

出版说明

近年来,随着微电子和计算机技术渗透到各个技术领域,人类正在步入一个技术迅猛发展的新时期。这个新时期的主要标志是计算机和信息处理的广泛应用。计算机在改造传统产业,实现管理自动化,促进新兴产业的发展等方面都起着重要作用,它在现代化建设中的战略地位愈来愈明显。计算机科学与其它学科的交叉又产生了许多新学科,推动着科学技术向更广阔的领域发展,正在对人类社会产生深远的影响。

科学技术是第一生产力。计算机科学技术是我国高科技领域的一个重要方面。为了推动我国计算机科学及产业的发展,促进学术交流,使科研成果尽快转化为生产力,清华大学出版社与广西科学技术出版社联合设立了“计算机学术著作基金”,旨在支持和鼓励科技人员,撰写高水平的学术著作,以反映和推广我国在这一领域的最新成果。

计算机学术著作出版基金资助出版的著作范围包括:有重要理论价值或重要应用价值的学术专著;计算机学科前沿探索的论著;推动计算机技术及产业发展的专著;与计算机有关的交叉学科的论著;有较大应用价值的工具书;世界名著的优透翻译作品。凡经作者本人申请,计算机学术著作出版基金评审委员会评审通过的著作,将由该基金资助出版,出版社将努力做好出版工作。

基金还支持两社列选的国家高科技重点图书和国家教委重点图书规划中计算机学科领域的学术著作的出版。为了做好选题工作,出版社特邀请“中国计算机学会”、“中国中文信息学会”帮助做好组织有关学术著作丛书的列选工作。

热诚希望得到广大计算机界同仁的支持和帮助。

清华大学出版社
广西科学技术出版社

计算机学术著作出版基金办公室

1992年4月

丛 书 序 言

计算机是当代发展最为迅猛的科学技术,其应用几乎已深入到人类社会活动和生活的一切领域,大大提高了社会生产力,引起了经济结构、社会结构和生活方式的深刻变化和变革,是最为活跃的生产力之一。计算机本身在国际范围内已成为年产值达 2500 亿美元的巨大产业,国际竞争异常剧烈,预计到本世纪末将发展为世界第一大产业。计算机科技具有极大的综合性质,与众多科学技术相交叉而反过来又渗入更多的科学技术,促进它们的发展。计算机科技内容十分丰富,学科分支生长尤为迅速,日新月异,层出不穷。因此在我国计算机科技尚比较落后的情况下,加强计算机科技的传播实为当务之急。

中国计算机学会一直把出版图书刊物作为学术活动的重要内容之一。我国计算机专家学者通过科学实践,做出了大量成果,积累了丰富经验与学识。他们有撰写著作的很大积极性,但相当时期以来计算机学术著作由于印数不多,出版往往遇到不少困难,专业性越强越有深度的著作,出版难度越大。最近清华大学出版社与广西科学技术出版社为促进我国计算机科学技术及产业的发展,推动计算机科技著作的出版工作,特设立“计算机学术著作出版基金”,以支持我国计算机科技工作者撰写高水平的学术著作,并将资助出版的著作列为中国计算机学会的学术著作丛书。我们十分重视这件事,并已把它列为学会本届理事会的工作要点之一。我们希望这一系列丛书能对传播学术成果、交流学术思想、促进科技转化为生产力起到良好作用,能对我国计算机科技发展具有有益的导向意义,也希望我国广大学会会员和计算机科技工作者,包括海外工作和学习的神州学人们能积极投稿,出好这一系列丛书。

中国计算机学会

1992 年 4 月 20 日

清华大学出版社 广西科学技术出版社
计算机学术著作出版基金

评 审 委 员 会

主 任 委 员 张效祥

副主任委员 汪成为 唐泽圣

委	员	王鼎兴	杨芙清	李三立	施伯乐	徐家福
		夏培肃	董韫美	黄 健	焦金生	

前 言

智能控制是在人工智能及自动控制等多学科基础上发展起来的新兴的交叉学科,目前尚未建立起一套较完整的智能控制的理论体系,关于它所包含的技术内容也还没有取得比较一致的共识。本书仅是根据作者的认识和体会,就智能控制的理论和技术作尽可能较为全面的介绍,以弥补这方面缺乏系统性资料的不足,期望达到抛砖引玉的作用。

本书在参考了国内外智能控制方面的重要文献基础上,对其主要内容加以系统总结和整理,同时也有部分内容是笔者研究工作的总结,如基于模糊状态模型的系统分析与设计、BP 网络学习算法的改进、模糊神经网络、具有再励学习的神经网络模糊 BOXES 控制、异步自学习控制、改进的遗传算法及利用遗传算法进行路径规划等。

本书的内容曾在清华大学研究生的“智能控制”课程中讲授过两遍,并曾印成讲义用作 1995 年全国智能控制高级研讨班的教材。这些教学实践为提高和改善本书的质量提供了重要的帮助。

本书第 1 章是绪论。简要介绍了智能控制的发展概况、研究对象、基本结构、主要特点、采用的数学工具及包含的主要理论等。

第 2 章介绍模糊逻辑控制。在简要介绍了模糊集合及模糊逻辑推理的基础上,重点介绍了模糊控制的基本原理、模糊控制系统的分析与设计以及自适应模糊控制等内容。

第 3 章介绍神经网络控制。重点介绍几种用于控制的神经网络模型,其中包括多层前馈网络、Hopfield 网络、CMAC、B 样条、RBF 及模糊神经网络等。在此基础上系统地介绍了基于神经网络的建模与控制。最后介绍了神经网络在机器人控制中的应用。

第 4 章介绍专家控制。主要介绍了专家控制的基本原理、典型结构和当前的研究课题,最后介绍了一种仿人智能控制。

第 5 章介绍学习控制。主要介绍了基于模式识别的学习控制、基于迭代和重复的学习控制以及联结主义的学习控制。

第 6 章介绍分层递阶控制。重点介绍 G. N. Saridis 等人提出的由组织级、协调级和执行级所组成的分层递阶的智能控制结构和原理。

第 7 章介绍遗传算法。重点介绍其工作原理、基本算法、算法改进及应用举例等。

本书第 1、2、6、7 章和第 3 章的第 1—5 节及第 8 节由孙增圻编写。第 3 章的第 6 和 7 节由邓志东编写。第 4、5 章由张再兴编写,邓志东为第 5 章提供了部分素材。全书由孙增圻统稿。

在本书编写过程中得到了许多人的支持帮助。其中:王鹏为第 2 章提供了部分素材,金亿创为第 3 章提供了部分素材,郭晨为第 4 章提供了部分素材。孙冬萍和钱宗华为本书的文稿整理作了大量的工作。在此,向以上提到的各位及其他为本书提供帮助的人们一并表示感谢。

由于笔者的水平所限,书中尚存在一些不足和错误之外,欢迎读者批评指正。

编著者

1996 年 4 月于清华大学

目 录

前言	VI
第 1 章 绪论	1
1.1 智能控制的基本概念	1
1.1.1 智能控制的研究对象	1
1.1.2 智能控制系统	1
1.1.3 智能控制系统的基本结构	2
1.1.4 智能控制系统的主要功能特点	3
1.1.5 智能控制研究的数学工具	4
1.2 智能控制的发展概况	4
1.3 智能控制理论	10
参考文献	15
第 2 章 模糊逻辑控制	16
2.1 概述	16
2.1.1 模糊控制与智能控制	16
2.1.2 模糊集合与模糊数学的概念	16
2.1.3 模糊控制的发展和应用概况	17
2.2 模糊集合及其运算	19
2.2.1 模糊集合的定义及表示方法	19
2.2.2 模糊集合的基本运算	22
2.2.3 模糊集合运算的基本性质	23
2.2.4 模糊集合的其它类型运算	24
2.3 模糊关系	24
2.3.1 模糊关系的定义及表示	24
2.3.2 模糊关系的合成	26
2.4 模糊逻辑与近似推理	27
2.4.1 语言变量	27
2.4.2 模糊蕴含关系	28
2.4.3 近似推理	30
2.4.4 模糊蕴含关系运算方法的比较和选择	35
2.4.5 合成运算方法的选择	40
2.4.6 句子连接关系的逻辑运算	41
2.5 基于控制规则库的模糊推理	42
2.5.1 模糊推理的基本方法	42

2.5.2	模糊推理的性质	45
2.5.3	模糊控制中几种常见的模糊推理类型	49
2.6	模糊控制的基本原理	52
2.6.1	模糊控制器的基本结构和组成	52
2.6.2	模糊化运算	54
2.6.3	数据库	55
2.6.4	规则库	58
2.6.5	模糊推理与清晰化计算	60
2.6.6	论域为离散时模糊控制的离线计算	62
2.7	模糊控制系统的分析和设计	67
2.7.1	模糊模型表示	67
2.7.2	模糊系统分析	68
2.7.3	模糊系统设计	74
2.7.4	基于 Takagi-Sugeno 模型的稳定性分析和设计	81
2.7.5	基于模糊状态方程模型的系统分析和设计	90
2.8	自适应模糊控制	107
2.8.1	基于性能反馈的直接自适应模糊控制	107
2.8.2	基于模糊模型求逆的间接自适应模糊控制	114
	参考文献	123
第3章	神经网络控制	125
3.1	概述	125
3.1.1	神经元模型	125
3.1.2	人工神经网络	127
3.1.3	生物神经网络系统与计算机处理信息的比较	128
3.1.4	神经网络的发展概况	128
3.2	前馈神经网络	129
3.2.1	感知器网络	130
3.2.2	BP 网络	133
3.2.3	BP 网络学习算法的改进	135
3.2.4	神经网络的训练	136
3.3	反馈神经网络	140
3.3.1	离散 Hopfield 网络	140
3.3.2	连续 Hopfield 网络	150
3.3.3	Boltzmann 机	154
3.4	局部逼近神经网络	158
3.4.1	CMAC 神经网络	158
3.4.2	B 样条神经网络	163
3.4.3	径向基函数神经网络	168

3.5	模糊神经网络	169
3.5.1	基于标准模型的模糊神经网络	170
3.5.2	基于 Takagi-Sugeno 模型的模糊神经网络	177
3.6	基于神经网络的系统建模与辨识	181
3.6.1	概述	181
3.6.2	逼近理论与网络建模	183
3.6.3	利用多层静态网络的系统辨识	187
3.6.4	利用动态网络的系统辨识	190
3.7	神经网络控制	194
3.7.1	概述	194
3.7.2	神经网络控制结构	195
3.7.3	基于全局逼近神经网络的控制	201
3.7.4	基于局部逼近神经网络的控制	205
3.7.5	模糊神经网络控制	209
3.7.6	有待解决的问题	216
3.8	神经网络在机器人控制中的应用	217
3.8.1	神经网络运动学控制	217
3.8.2	神经网络动力学控制	222
3.8.3	神经网络路径规划	227
	参考文献	237
第4章	专家控制	240
4.1	概述	240
4.1.1	专家控制的由来	240
4.1.2	专家系统	240
4.1.3	专家控制的研究状况和分类	244
4.2	专家控制的基本原理	246
4.2.1	专家控制的功能目标	246
4.2.2	控制作用的实现	247
4.2.3	设计规范和运行机制	219
4.3	专家控制系统的典型结构	250
4.3.1	系统结构	250
4.3.2	系统实现	256
4.4	专家控制的例示	258
4.4.1	自动调整过程	258
4.4.2	自动调整过程的实现	263
4.5	专家控制技术的研究课题	264
4.5.1	实时推理	264
4.5.2	知识获取	266

4.5.3 专家控制系统的稳定性分析·····	270
4.6 一种仿人智能控制·····	274
4.6.1 概念和定义·····	274
4.6.2 原理和结构·····	276
4.6.3 仿人智能控制的特点·····	278
参考文献·····	279
第5章 学习控制 ·····	282
5.1 概述·····	282
5.1.1 学习控制问题的提出·····	282
5.1.2 学习控制的表述·····	283
5.1.3 学习控制和自适应控制·····	284
5.1.4 学习控制的研究状况和分类·····	284
5.2 基于模式识别的学习控制·····	286
5.2.1 学习控制系统的一般形式·····	286
5.2.2 模式分类·····	288
5.2.3 可训练控制器·····	290
5.2.4 线性再励学习控制·····	292
5.2.5 Bayes 学习控制·····	292
5.2.6 基于模式识别的其他学习控制方法·····	294
5.2.7 研究课题·····	297
5.3 基于迭代和重复的学习控制·····	297
5.3.1 迭代和重复自学习控制的基本原理·····	298
5.3.2 异步自学习控制·····	301
5.3.3 异步自学习控制时域法·····	304
5.3.4 异步自学习控制频域法·····	309
5.4 联结主义学习控制·····	313
5.4.1 基本思想·····	313
5.4.2 联结主义学习系统的实现原理·····	315
5.4.3 联结主义学习控制系统的结构·····	322
5.4.4 研究课题·····	324
参考文献·····	325
第6章 分层递阶智能控制 ·····	328
6.1 一般结构原理·····	328
6.2 组织级·····	330
6.3 协调级·····	333
6.3.1 协调级的原理结构·····	333
6.3.2 Petri 网翻译器·····	336
6.3.3 协调级的 Petri 网结构·····	337

6.3.4 协调级结构的决策和学习.....	339
6.4 执行级	343
参考文献.....	344
第7章 遗传算法	345
7.1 概述	345
7.2 遗传算法的工作原理及操作步骤	347
7.2.1 遗传算法的基本操作.....	347
7.2.2 遗传算法的模式理论.....	350
7.3 遗传算法的实现及改进	353
7.3.1 遗传算法的实现.....	353
7.3.2 遗传算法的改进.....	356
7.3.3 改进的遗传算法举例.....	357
7.4 遗传算法应用举例	361
7.4.1 遗传算法在模糊逻辑控制中的应用.....	361
7.4.2 遗传算法在神经网络控制中的应用.....	365
7.4.3 用遗传算法进行路径规划.....	369
参考文献.....	373

第 1 章 绪 论

1.1 智能控制的基本概念

智能控制是一个新兴的学科领域,目前有关智能控制的定义、理论、结构等尚无统一的系统描述。下面仅就它的研究对象、智能控制系统的主要特征以及研究的数学工具等问题作简要的介绍。

1.1.1 智能控制的研究对象

智能控制是控制理论发展的高级阶段。它主要用来解决那些用传统方法难以解决的复杂系统的控制问题。其中包括智能机器人系统、计算机集成制造系统(CIMS)、复杂的工业过程控制系统、航天航空控制系统、社会经济管理系统、交通运输系统、环保及能源系统等。具体地说,智能控制的研究对象具备以下一些特点:

1. 不确定性的模型

传统的控制是基于模型的控制,这里的模型包括控制对象和干扰模型。对于传统控制通常认为模型已知或者经过辨识可以得到。而智能控制的对象通常存在严重的不确定性。这里所说的模型不确定性包含两层意思:一是模型未知或知之甚少;二是模型的结构和参数可能在很大范围内变化。无论哪种情况,传统方法都难于对它们进行控制,而这正是智能控制所要研究解决的问题。

2. 高度的非线性

在传统的控制理论中,线性系统理论比较成熟。对于具有高度非线性的控制对象,虽然也有一些非线性控制方法,但总的说来,非线性控制理论还很不成熟,而且方法比较复杂。采用智能控制的方法往往可以较好地解决非线性系统的控制问题。

3. 复杂的任务要求

在传统的控制系统中,控制的任务或者是要求输出量为定值(调节系统),或者是要求输出量跟随期望的运动轨迹(跟踪系统)。因此控制任务的要求比较单一。对于智能控制系统,任务的要求往往比较复杂。例如,在智能机器人系统中,它要求系统对一个复杂的任务具有自行规划和决策的能力,有自动躲避障碍运动到期望目标位置的能力。再如,在复杂的工业过程控制系统中,它除了要求对各被控物理量实现定值调节外,还要求能实现整个系统的自动启停、故障的自动诊断以及紧急情况的自动处理等功能。

1.1.2 智能控制系统

智能控制系统是实现某种控制任务的一种智能系统。所谓智能系统是指具备一定智能行为的系统。具体地说,若对于一个问题的激励输入,系统具备一定的智能行为,它能够产生合适的求解问题的响应,这样的系统便称为智能系统。例如,对于智能控制系

统，激励输入是任务要求及反馈的传感信息等，产生的响应则是合适的决策和控制作用。

从系统的角度，智能行为也是一种从输入到输出的映射关系，这种映射关系并不能用数学的方法精确地加以描述，因此它可看成是一种不依赖于模型的自适应估计。例如，一个钢琴家弹奏一支优美的乐曲，这是一种高级的智能行为，其输入是乐谱，输出是手指的动作和力度。输入和输出之间存在某种映射关系，这种映射关系可以定性地进行说明，但不可能用数学的方法来精确地加以描述，因此也不可能由别的人来精确地加以再现。

G. N. 萨里迪斯(Saridis)给出了另一种定义，通过驱动自主智能机来实现其目标而无需操作人员参与的系统称为智能控制系统。这里所说的智能机指的是能够在结构化或非结构化、熟悉或不熟悉的环境中，自主地或有人参与地执行拟人任务的机器。

上面的定义仍然比较抽象，下面给出一个通俗但并不严格的定义：在一个控制系统中，如果控制器完成了分不清是机器还是人完成的任务，称这样的系统为智能控制系统。

1.1.3 智能控制系统的基本结构

智能控制系统的结构如图 1.1 所示。

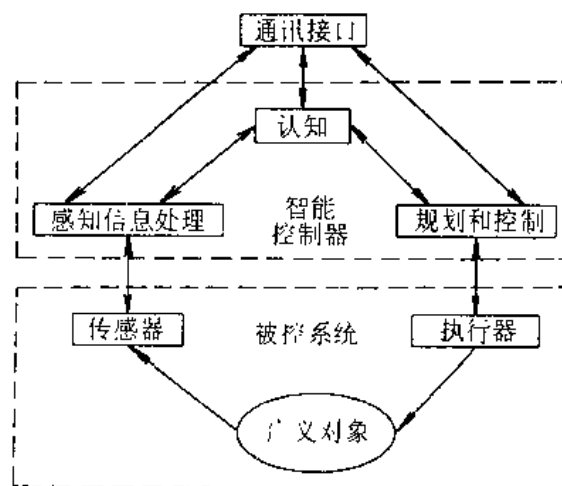


图 1.1 智能控制系统的典型结构

在该系统中，广义对象包括通常意义下的控制对象和所处的外部环境。例如对于智能机器人系统来说，机器人手臂、被操作物体及其所处环境统称为广义对象。传感器则包括关节位置的传感器、力传感器，还可能包括触觉传感器、滑觉传感器或视觉传感器等。感知信息处理将传感器得到的原始信息加以处理。例如视觉信息便要经过很复杂的处理才能获得有用信息。认知部分主要接收和储存知识、经验和数据，并对它们进行分析、推理，作出行动的决策，送至规划和控制部分。通讯接口除建立人-机之间的联系外，也建立系统中各模块之间的联系。规划和控制是整个系统的核心，它根据给定的任务要求、反馈的信息及经验知识，进行自动搜索、推理决策、动作规划，最终产生具体的控制作用。

经执行部件作用于控制对象。

对于不同用途的智能控制系统，以上各部分的形式和功能可能存在较大的差异。

G. N. 萨里迪斯提出了智能控制系统的分层递阶的组成结构形式，如图 1.2 所示。其中执行级一般需要比较准确的模型，以实现具有一定精度要求的控制任务；协调级用来协调执行级的动作，它不需要精确的模型，但需具备学习功能以便在再现的控制环境中改善性能，并能接受上一级的模糊指令和符号语言；组织级将操作员的自然语言翻译成机器语言，组织决策、规划任务、并直接干预低层的操作。在执行级中，识别的功能在于获得不确定的参数值或监督系统参数的变化。协调级中，识别的功能在于根据执行级送来的测量数据和组织级送来的指令产生合适的协调作用。在组织级中，识别的功能在于翻译定性的命令和其它的输入。该分层递阶的智能控制有两个明显的特点：

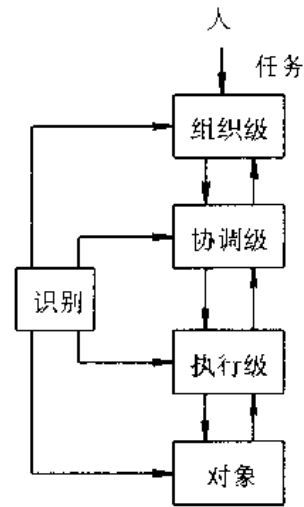


图 1.2 分层递阶智能控制结构

(1) 对控制来讲，自上而下控制精度愈来愈高；

(2) 对识别来讲，自下而上信息回馈愈来愈粗略。这种分层递阶的结构形式已被成功地应用于机器人的智能控制、交通系统的智能控制及管理。

以上虽然给出了智能控制系统的几种定义，但是它并没有提出一个明确的界限，什么样的系统才算是智能控制系统。同时，即使是智能控制系统，其智能程度也有高有低。因此，下面只能给出智能控制系统大致应具备的一些功能特点。

1.1.4 智能控制系统的主要功能特点

1. 学习功能

关于什么是学习，人们尚有许多争议，下面给出 G. N. 萨里迪斯给出的一个定义：一个系统，如果能对一个过程或其环境的未知特征所固有的信息进行学习，并将得到的经验用于进一步的估计、分类、决策或控制，从而使系统的性能得到改善，那么就称该系统为学习系统。

具有学习功能的控制系统也称为学习控制系统，它主要强调其具备学习功能的特点。学习控制系统可看成是智能控制系统的一种。智能控制系统的学习功能可能有低有高，低层次的学习功能主要包括对控制对象参数的学习，高层次的学习则包括知识的更新和遗忘。

2. 适应功能

这里所说的适应功能比传统的自适应控制中的适应功能具有更广泛的含义。它包括更高层次的适应性。正如前面已经提到的，智能控制系统中的智能行为实质上是一种从输入到输出之间的映射关系。它可看成是不依赖模型的自适应估计，因此它具有很好的适应性能。当系统的输入不是已经学习过的例子时，由于它具有插补功能，从而可给出

合适的输出。甚至当系统中某些部分出现故障时，系统也能够正常的工作。如果系统具有更高层次的智能，它还能自动找出故障甚至具备自修复的功能，从而体现了更强的适应性。

3. 组织功能

它指的是对于复杂的任务和分散的传感信息具有自行组织和协调的功能。该组织功能也表现为系统具有相应的主动性和灵活性，即智能控制器可以在任务要求的范围内自行决策、主动地采取行动；而当出现多目标冲突时，在一定的限制下，控制器可有权自行裁决。

1.1.5 智能控制研究的数学工具

传统的控制理论主要采用微分方程、状态方程以及各种变换等作为研究的数学工具，它本质上是数值计算方法。而人工智能则主要采用符号处理，一阶谓词逻辑等作为研究的数学工具。两者有着根本的区别。智能控制研究的数学工具则是上述两个方面的交叉和结合，它主要有以下几种形式：

1. 符号推理与数值计算的结合

例如专家控制，它的上层是专家系统，采用人工智能中的符号推理方法。下层是传统的控制系统，采用的仍是数值计算方法。因此整个智能控制系统的数学研究工具是这两种方法的结合。

2. 离散事件系统与连续时间系统分析的结合

计算机集成制造系统(CIMS)和智能机器人便属于这样的情况，它们是典型的智能控制系统。例如在CIMS中，上层任务的分配和调度、零件的加工和传输等均可用离散事件系统理论来进行分析和设计；下层的控制，如机床及机器人的控制，则采用常规的连续时间系统分析方法。

3. 介于两者之间的方法

(1) 神经网络。它通过许多简单的关系来实现复杂的函数关系。它本质上是非线性的动力学系统，但它并不依赖于模型。

(2) 模糊集合论。它形式上是利用规则进行逻辑推理，但其逻辑取值可在0与1之间连续变化，采用数值的方法而非符号的方法进行处理。

以上两种方法，即神经网络和模糊集合论，在某些方面如逻辑关系、不依赖于模型等类似于人工智能的方法；而其他方面如连续取值和非线性动力学特性等则类似于通常的数值方法，即传统控制理论的数学工具。因而它们是介于二者之间的数学工具，且可能是进行智能控制研究的主要的数学工具。

1.2 智能控制的发展概况

到了60年代，自动控制理论和技术的发展已渐趋成熟，而人工智能还只是个诞生不久的新兴技术。1966年J. M. 门德尔(Mendel)首先主张将人工智能用于飞船控制系统的设计^[4]。1971年著名学者傅京逊(K. S. Fu)从发展学习控制的角度首次正式提出智能控

制这个新兴的学科领域。他的文章题目是：“学习控制系统和智能控制系统：人工智能与自动控制的交叉”^[5]。他列举了以下三种智能控制系统的例子：

1. 人作为控制器的控制系统。图 1.3 所示为操纵驾驶杆以瞄准目标的手动控制系统。

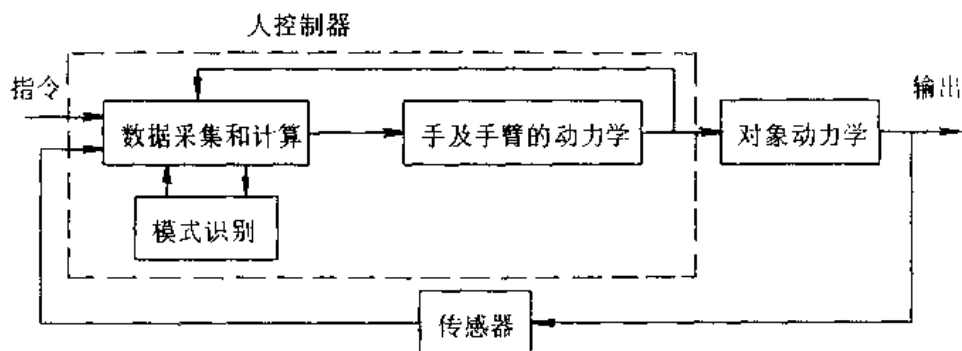


图 1.3 人作为控制器的控制系统

这里，人作为控制器包含在闭环控制回路内。由于人具有识别、决策、控制等功能，因此对于不同的控制任务及不同的对象和环境情况，它具有自学习、自适应和自组织的功能，自动采用不同的控制策略以适应不同的情况。显然，这样的控制系统属于智能控制系统。

2. 人-机结合作为控制器的控制系统。在这样的控制系统中，机器(主要是计算机)完成那些连续进行的需要快速计算的常规控制任务。人则主要完成任务分配、决策、监控等任务。图 1.4 表示了一个由人-机结合作为控制器的遥控操作系统典型结构，它是另外一种类型的智能控制系统。

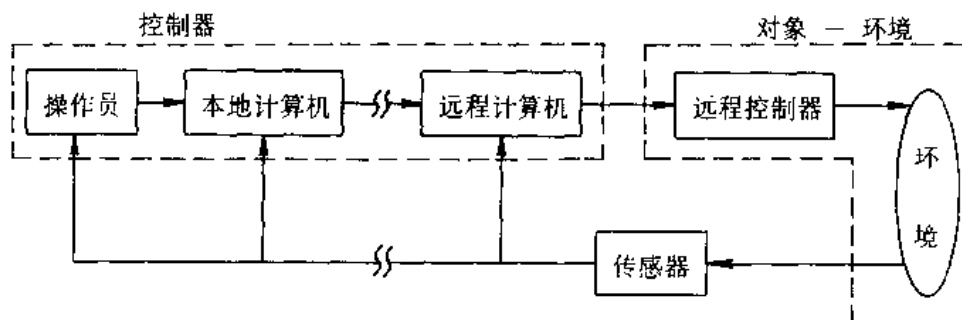


图 1.4 人-机结合作为控制器的遥控操作系统

3. 无人参与的智能控制系统。以上两种类型的智能控制系统均是有人参与的，许多智能控制的任务是由人完成的。我们更感兴趣的是如何将前面由人完成的那些功能变为由机器来完成，从而设计出无人参与的智能控制系统。一个最典型的例子是自主机器人。图 1.5 表示了斯坦福研究所(SRI)机器人系统的结构图。

在该控制系统中，控制器主要完成以下功能：问题求解和规划、环境建模、传感信息

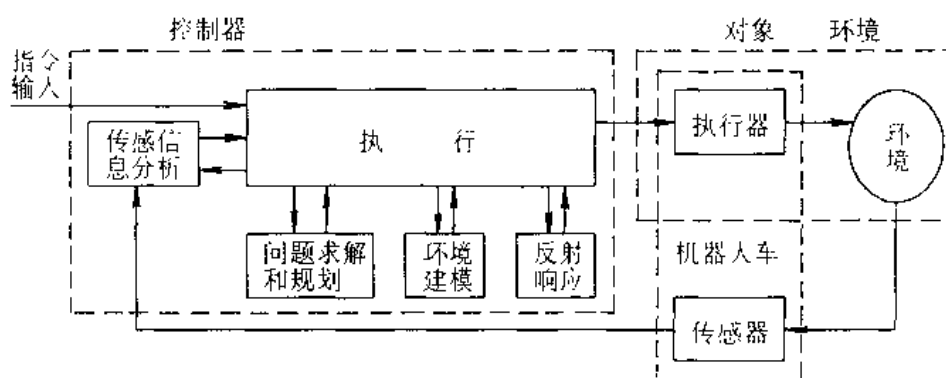


图 1.5 SRI 机器人系统

分析和反射响应。反射响应类似于常规控制器，它主要完成简单情况下的控制。该机器人本体由两个步进电机分别独立地驱动左右两个轮子，从而达到控制速度和方向的目的。在它前面装有摄像头和光学测距装置。本体与控制器之间通过无线电方式进行通讯。该系统所要完成的典型任务是在它运行的环境中重新排列一些简单的物体。为了完成这个特定的任务，机器人系统必须在动作之前首先进行问题求解和规划，以获得一组基本动作序列，这些基本动作包括轮子运动、摄像头读数等。为了获得该动作序列以实现特定的任务，必须知道环境模型的知识，即执行一个基本动作后，环境的状态将如何变化。因此，通过分析使得在执行一系列的基本动作后，被控制的过程能够最终达到所需要的状态。随着动作的执行，环境模型也随之发生改变。所以必须随时记录和更新环境模型的信息，这实际上便是学习的过程。为了获得环境的模型并能对它不断更新，必须有相应的传感器和信息处理系统。这里视觉传感系统是有关模型信息的主要来源。对于简单的任务和环境模型，可以采用动态规划的方法来获得最优解。但是对于复杂的环境，必须用模式识别的方法来进行分析，这时必须采用启发式的问题求解步骤来确定可行的动作序列，这个动作序列不一定是最优的，它可通过学习过程来不断地加以改进。

傅京逊的这篇论文中列举了三种智能控制系统的典型情况。第三种情况即是人们所希望的无人参与的智能控制系统。这里被控过程是一个复杂和不确定性的环境，这时要建立被控过程的准确的数学模型，并据此采用常规的控制方法是十分困难或几乎不可能的。对于复杂的环境和复杂的任务，如何将人工智能技术中较少依赖模型的问题求解方法与常规的控制方法相结合，这正是智能控制所要解决的问题。

G. N. 萨里迪斯对智能控制的发展作出了重要贡献，他在 1977 年出版了“随机系统的自组织控制”一书^[3]，1979 年发表了综述文章“朝向智能控制的实现”^[6]。在这两篇著作中他从控制理论发展的观点，论述了从通常的反馈控制到最优控制、随机控制，再到自适应控制、自学习控制、自组织控制，并最终向智能控制这个更高阶段发展的过程。他首次提出了分层递阶的智能控制结构形式。图 1.6 所示为他及其同事所研制的一个由大脑来监督和指挥的具有 P 个自由度的仿生手臂的系统结构图。该仿生手臂可用作假肢，也可用在危险环境中以完成一些拟人的操作。当用作假肢时，它通过肌电信号直接从神经系统接受定性命令。当用于危险环境作业时，操作人员可根据对环境的观察给出定性命令。该系统

可采用如图 1.6 所示的分层递阶的控制结构,其控制精度由下往上逐级地递减,智能程度递阶地增加。整个控制结构分成如下三个层次:

- (1) 语言组织级
- (2) 模糊自动机作为协调级
- (3) 一组自组织控制器作为控制级

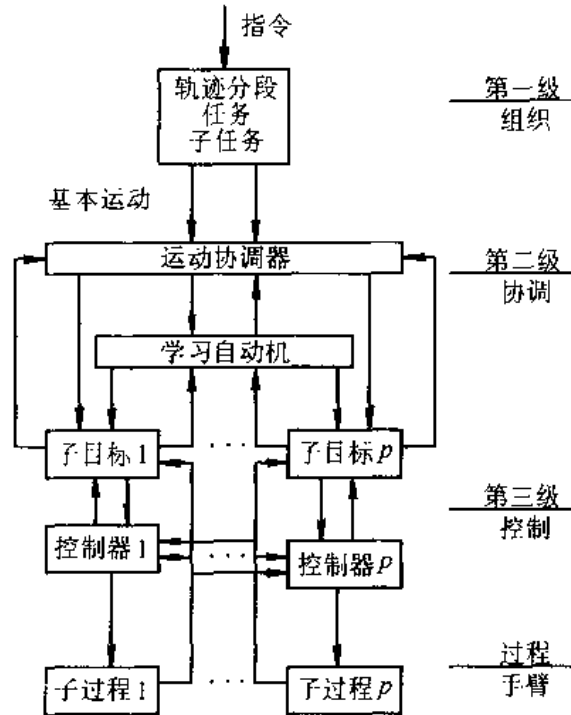


图 1.6 假肢手臂的分层递阶智能控制

对于自组织控制级,取性能指标函数为

$$J = \sum_{i=1}^p \mu_i J_i(\alpha_i)$$

其中 $J_i(\alpha_i)$ 是子过程的性能指标, α_i 是反映子过程响应速度的调整系数, μ_i 是协调各子过程运动的调整系数。对每个子过程构造反馈控制

$$u_i(z) = K_i \varphi(z_i) + C_i \Psi(z)$$

其中第一项是不考虑耦合时子系统的最优控制,第二项表示与其它子系统非线性耦合有关的控制项。对于每个子过程采用扩展子区间算法来求出渐近最优系数 K_i 和 C_i 。可见自组织控制级主要用来实现手的精确控制,基本上不具备智能。

协调级由模糊自动机来实现,模糊自动机可定义为一个 6 元组 $[Z, Q, U, F, H, \xi]$, 其中 Z 为模糊输入的有限集合, Q 是状态的有限集合, U 是输出的有限集合, F 是状态转移函数, H 是输出函数, ξ 是赋给状态的隶属度向量。对于每一个模糊输入 z_i , 相应地有一个隶属度转移矩阵 Z_i 来修改隶属度函数 ξ , 这样的模糊自动机可以用来协调手臂的运动。它的输入是上层送下来的模糊输入指令, 输出是基本运动的组合。根据这样的运动组合, 手臂可以从起始状态运动到要求的终端状态。

按照上面的讨论,当给定模糊指令和终态时,模糊自动机可经过训练给出手臂的恰当的复合运动。它比自组织控制具有较高的智能。但是,操作人员大脑产生的指令可能是更为综合的任务形式,例如要求拿一杯水来喝,这时需要有更高智能的一级来作为大脑和模糊自动机间的接口,它能将整个任务分段,对每段给定适当的终态 $x_d(T)$,并能处理由大脑传来的传感器反馈信息,评价任务的完成情况、在线地产生运动的组合、分段及方向改变等信息,从而减轻操作人员的负担。采用语言方法可以实现这种类型的信息处理,该方法用逻辑指令按一些预定的语法和句法,以类似于自然语言的方式来处理词汇链,从而可以根据更高层的复合指令产生出适当的指令串送到模糊自动机,最终产生出所需要的复合运动。

萨里迪斯等后来在分层递阶智能控制的理论和实践方面又做了大量的工作^[11]。对智能控制系统的三级结构作了明确的分工和定义。讨论了每一级的实现方法,建造了一个智能机器人的实验系统。在理论上的一个重要贡献是定义了熵作为整个智能控制系统的性能度量。对每一级定义了熵的计算方法,证明了在执行级的最优控制等价于使某种熵最小的控制方法。对于组织级和协调级的实现,又在原有的基础上进行了改进。在最新的工作中采用神经网络中的 Boltzmann 机来实现组织级的功能,利用 Petri 网作为工具来实现协调级的功能。在萨里迪斯等人的倡议下,1985年8月在美国纽约州的 Troy 召开了第一次智能控制学术讨论会,然后不久在 IEEE 的控制系统学会中成立了智能控制技术委员会,首任主席是萨里迪斯教授,从1987年起每年召开一次智能控制的国际学术会议。总之萨里迪斯等人为智能控制学科的建立和发展作出了重要贡献,尤其是他在分层递阶智能控制的理论和实践方面坚持不懈地作了大量的工作,分层递价智能控制理论已成为智能控制理论中一个相对比较成熟的重要分支。

在智能控制的发展过程中,另一个值得一提的著名学者是 K. J. 奥斯特洛姆 (Astrom),他在1986年发表的“专家控制”的著名文章中^[12],将人工智能中的专家系统技术引入到控制系统中,组成了另外一种类型的智能控制系统。在实际的控制系统中,核心的控制算法只是其中的一部分,它还需要许多其它的逻辑控制。例如对于一个 PID 调节器来说,需要考虑操作员接口、手动与自动的平滑切换、参数突然改变所引起的过渡过程、执行部件的非线性影响、积分项引起的大摆动现象、上下限报警等问题。采用启发逻辑可用来解决这些问题。在控制软件中,这部分的程序要远远大于控制算法的程序量。即使在控制算法部分,也可针对不同的情况采用不同的控制算法来获得更为满意的控制性能,这也需要启发逻辑来实现这样的转换。

为了说明上面的概念,奥斯特洛姆的文章给出了一个简单的工业过程控制的例子来加以说明。设控制对象的差分算子模型为

$$Ay(t) = Bu(t) + Ce(t)$$

其中 u 是控制变量, y 是输出量, e 是白噪声, A, B, C 是算子多项式。最小方差控制为

$$Ru(t) = -Sy(t)$$

多项式 R 和 S 满足

$$z^{d-1}CB = AR + BS$$

其中 $d = \deg A - \deg B$ 。由于最小方差控制器要抵消控制对象的零点,因此要求控制对象

为最小相位,即无零极点在单位圆外,而且即使所有的零点均在单位圆内,若很靠近单位圆周,则系统的响应中也将会出现纹波现象,因此在系统中需设置纹波监测器。当监测到有严重纹波现象时,可通过增大 d 来消除纹波现象。为了监测系统是否处于最小方差控制的状态,可以通过计算系统输出的相关函数来判别。因为这时系统的输出应为

$$y(t) = \lambda[e(t) + f_1 e(t-h) + \cdots + f_{d-1} e(t-dh+h)]$$

其中 $F=R/B$, h 是采样周期。由于假定 $e(t)$ 是白噪声,所以 $y(t)$ 在不同时刻是不相关的,为此,可设置最小方差监测器来检测系统是否处于最小方差控制状态。

如果不能得到控制对象的精确模型,则可采用自校正调节器,在一定条件下它可收敛到最小方差控制器。这时必须引入参数估计器。为了获得准确的参数估计,必须有足够的激励。为此,需引入激励监测器。参数估计只适合于系统参数为常数或缓慢变化的情况。因此还需包括一个参数跳变检测器。当没有充足的激励时,便不能获得正确的参数估计,为此需提供一扰动信号发生器。自校正控制需要已知如下的先验知识:采样周期 h 、延迟拍数 d 、多项式 R 的阶数 n_R 、 S 的阶数 n_S 、遗忘因子 λ 、初始估计 θ_0 、初始方差 P_0 以及控制量的高低限。参数 d 和 h 十分关键,若估计太小可使系统不稳定。为此,可引入稳定性监视器来进行检测,通过计算协方差函数 $r_{yy}(\tau)$ 和 $r_{yu}(\tau)$ 可用来确定 n_R 和 n_S 是否足够大,于是系统中还需包含一个阶次监测器。

时间延时 dh 的估计准确与否对控制器的性能有很大影响。因此系统中还应包含 K_c-t_c 估计器。 K_c 是系统临界振荡时的增益, t_c 是临界振荡周期,则 dh 近似为 $t_c/2$ 。根据 K_c 和 t_c 还可很容易确定PID控制器的参数,因此可将PID控制作为备用控制。

对于以上所说到的各种情况,可采用专家系统来进行统一的管理和决策。正常情况下系统采用最小方差控制。若检测到纹波现象较严重,则增大 d 。若通过最小方差监测器监测到系统已不是处于最小方差控制状态,说明系统的模型不准确或参数发生了改变,则系统转入自校正调节状态,应启动参数估计器。同时启动激励监测器,看是否有足够的激励。若激励不足,启动扰动信号发生器。同时根据稳定性监视器来检测 dh 的估值的正确性。若系统稳定性能差,则启动 K_c-t_c 估计器,重估 dh 。根据最小方差监测器可判断自校正控制是否已收敛到最小方差控制,若已收敛,则转入到最小方差控制。这一系列过程均可根据专家系统中的规则搜索来自动地完成。

奥斯特洛姆所提出的专家控制将人工智能中的专家系统技术与传统的控制方法相结合,并吸取了这两者的长处,在实际中取得了明显的效果。事实上,自那以后已经有很多采用这种方法在实际中成功应用的报道。虽然,专家控制在理论上并没有新的发展和突破,但是,它作为智能控制的一种形式,在实际上有着很广阔的应用前景。

近年来,神经网络的研究得到了越来越多的关注和重视。它在控制中的应用也是其中的一个主要方面,由于神经网络在许多方面试图模拟人脑的功能,因此神经网络控制并不依赖于精确的数学模型,而显示出具有自适应和自学习的功能,因此它也是智能控制的一种典型形式。目前利用神经网络组成自适应控制以及它在机器人中的应用研究方面均取得了很多成果,显示出了广泛的应用前景。

模糊控制是又一类智能控制的形式。现代计算机虽然有着极高的计算速度和极大的存储能力,但却不能完成一些人看起来十分简单的任务。一个很重要的原因是人具有模糊

决策和推理的功能,模糊控制正是试图模仿人的这种功能。1965 年,L. A. 扎德(Zadeh)首先提出了模糊集理论,为模糊控制奠定了基础。在其后的 20 年中已有很多模糊控制在实际中获得应用成功的例子。

在我国,重庆大学周其鉴等人从 80 年代初便开始仿人智能控制的研究,他们也为智能控制的发展作出了贡献。

智能控制作为一门新兴学科,现在还只是处于它的发展初期,还没有形成完整的理论体系。本书只就已经发展起来的几个智能控制分支加以介绍,以反映智能控制的发展现状。

1.3 智能控制理论

智能控制是一多学科交叉。傅京逊在 1971 年的文章中称它是人工智能与自动控制的交叉^[5]。后来萨里迪斯加进了运筹学,认为智能控制是人工智能、运筹学和自动控制三者的交叉,并用图 1.7 所示来形象的说明这一点。

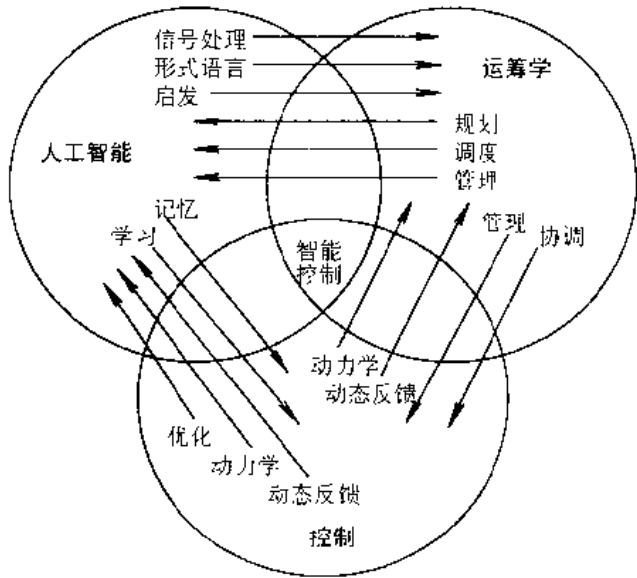


图 1.7 智能控制的多学科交叉

关于智能控制究竟应该是哪些学科的交叉,不同的人有不同的理解。有的人主张还应包括系统论、信息论、计算机科学、人类工程学等。但不管怎样,它至少说明智能控制是一个多学科的交叉,图 1.7 所示可能是目前多数人所接受的一种共识。

图 1.7 主要是针对分层递阶智能控制的情况。对于其它类型的智能控制,如专家控制、神经网络控制、模糊控制等,它所涵盖的学科领域不尽相同。如前所述,智能控制迄今尚未建立起完整的理论体系,因此要系统地讨论其理论内容为时尚早,下面仅就上面提到的几种类型的智能控制系统所包含的理论内容作一扼要介绍。

1. 自适应、自组织和自学习控制

自适应、自组织和自学习控制是传统控制向纵深发展的高级阶段。如前面已经讨论过的,适应功能、学习功能和组织功能是智能控制系统所具有的几个最主要的功能特点。因此,自适应自组织和自学习控制系统可看成为较初级的智能控制系统。同时它们也可构成分层递阶智能控制系统的下面一层的控制级。

自适应控制和自组织控制本质上并没有什么差别。自适应控制主要描述系统的行为,自组织控制主要描述系统的内部结构。根据萨里迪斯的定义^[3],所谓自组织控制是指在系统运行过程中,通过观测过程的输入和输出所获得的信息,能够逐渐减小系统的先验不确定性,而达到对系统的有效控制。自组织控制可由两种方法来实现:一种是给出明显的辨识来减小对象动力学所固有的不确定性;另一种则是设法减小与改进系统性能直接相关的不确定性。这后一种情形,可以认为隐含地进行着系统辨识,因为所积累的关于对象的信息,可由控制器直接予以应用而不经中间的模型。这两类自组织控制代表了两种不同的设计方法。

如果通过观测过程的输入和输出所获得的信息,能够减小过程参数的先验不确定性,则称该自组织控制为参数自适应自组织控制;若减小的是与改进系统性能直接相关的不确定性,则称之为品质自适应自组织控制。图 1.8 和图 1.9 分别表示了这两种控制方法的结构。在现有的许多文献中,分别称它们为自校正控制系统和模型参考自适应控制系统。

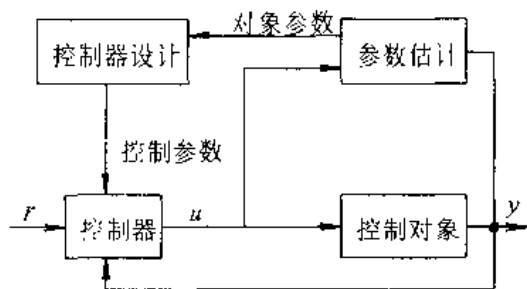


图 1.8 参数自适应自组织控制

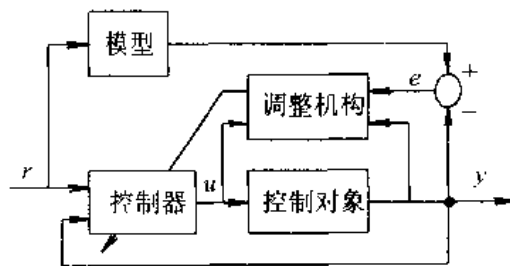


图 1.9 品质自适应自组织控制

所谓学习控制系统是指:一个系统,如果能对一个过程或其环境的未知特征所固有的信息进行学习,并将其学得的信息用来控制一个具有未知特征的过程,称之为学习控制系统。图 1.10 表示了学习控制系统的一种结构。研究学习控制的最常用的方法有以下几种:(1) 模式分类;(2) 再励学习;(3) 贝叶斯估计;(4) 随机逼近;(5) 随机自动机模型;(6) 模糊自动机模型;(7) 语言学方法。

还有一种主要基于人工智能方法的学习控制系统,它主要借助于人工智能的学习原理和方法,通过不断获取新的知识来逐步改变系统的性能,其典型结构如图 1.11 所示。其中知识库主要用来存储知识,并具有知识更新的功能。学习环节具有采集环境信息、接受监督指导、进行学习推理和修改知识库功能。执行环节主要利用知识库的知识,进行识别、决策并采取相应行动。监督环节主要进行性能评价、监督学习环节及控制选例环节。选例环节的作用主要是从环境中选取有典型意义的事例或样本,作为系统的训练集或学习对象。

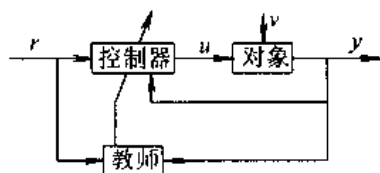


图 1.10 学习控制系统

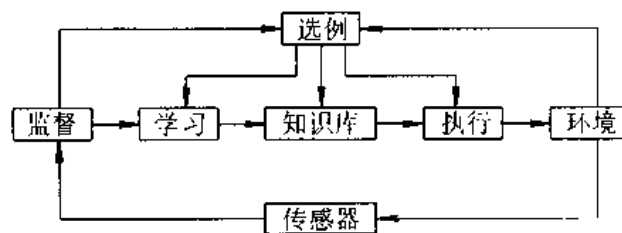


图 1.11 人工智能学习控制系统

另外还有一种基于重复性的学习控制,其典型结构如图 1.12 所示。其中 u_k 表示第 k 次运动的控制量, y_k 是实际输出, y_d 是期望的输出。采用学习控制算法 $u_{k+1} = u_k + f(e_k)$, 使得经过多次重复后,在 u_k 的作用下系统能够产生期望的输出。这里的主要问题是学习控制算法的收敛性问题。

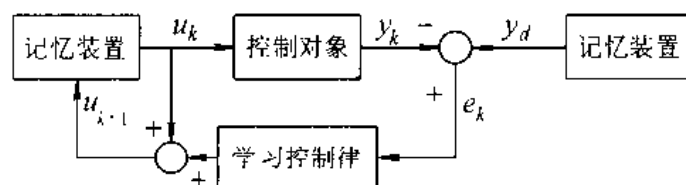


图 1.12 重复性学习控制系统

2. 知识工程

作为智能控制系统的一个重要分支的专家控制系统以及上面讨论的人工智能学习控制系统均离不开知识的表示、知识的运用、知识的获取和更新等问题。这些正是知识工程的主要问题。因此知识工程也是智能控制理论中的一项重要内容。

广义地讲,设计控制系统便是有效地组织和运用知识的过程。控制器则是运用知识进行推理决策、产生控制作用的装置,它一般由计算机来完成。对于传统的控制,控制对象模型及性能要求可以看成是用数值表示的知识。控制算法则是运用知识进行决策计算,以产生所需的控制作用。在智能控制系统中,有一部分是数值类型的知识,更主要的知识是一些经验、规则,它们是用符号形式来表示的。在这种情况下,设计控制器便是如何获取知识,如何运用知识进行推理、决策以产生有效的控制。在学习控制系统中,还有一个如何更新知识以实现学习的功能。

3. 信息熵

在分层递阶智能控制系统中,G. N. 萨里迪斯提出用熵作为整个系统的一个性能测度。因为在不同的层次以不同的形式包含了运动的不确定性。而熵正是采用概率模型时不确定性的一个度量。分层递阶智能控制系统的设计问题可以看成是如下的过程:在自上而下精度渐增、智能递减的分层递阶系统中,寻求正确的决策和控制序列以使整个系统的总熵极小。可见信息熵在分层递阶智能控制系统的分析和设计中起着十分重要的作用。

4. Petri 网

Petri 网是新近发展起来的一种既是图形也是数学的建模工具。它主要用来描述和研

究信息处理系统, 这些系统往往具有以下特点: 并发性、异步性、分布性和不确定性等。Petri 网的应用领域很广, 它可用于性能评价、通讯协议、柔性制造系统、离散事件系统、形式语言、多处理机系统、决策模型等。因此, Petri 网非常适用于在分层递阶智能控制系统中作为协调级的解析模型, 利用该模型可以比较容易地将协调级中各模块之间的连接关系描述清楚。它也可以比较容易地处理在协调过程中所碰到的并发活动和冲突仲裁等问题。同时利用该模型既可以作定性的也可作定量的分析, 这也是其它方法所难以做到的。

5. 人-机系统理论

人-机结合的控制系統也是一种智能控制系统。研究系统中人-机交互作用主要有三个目的: 一是研究人作为系统中的一个部件的特性, 并进而研究整个系统的行为; 二是在系统中如何构造仿人的特性, 从而实现无人参与的仿人智能控制; 三是研究人-机各自特性, 将人的高层决策能力与计算机的快速响应能力相结合, 充分发挥各自的优越性, 有效地构造出人-机结合的智能控制系统。

6. 形式语言与自动机

利用形式语言与自动机作为工具可以实现分层递阶智能控制系统中组织级和协调级的功能。在萨里迪斯的早期工作中, 组织级是由一种语言翻译器来实现的, 它将输入的定性指令映射为下层协调级可以执行的另外一种语言。协调级则采用随机自动机来实现, 这样的自动机也等价于一种形式语言。

形式语言和自动机理论作为处理符号指令的工具, 在设计智能控制系统的上层结构时是常常需要用到的。

7. 大系统理论

智能控制系统中的分层递阶的控制思想是与大系统理论中的分层递阶和分解协调的思想一脉相承的。虽然大系统理论是传统控制理论在广度方面的发展, 智能控制是传统控制理论在纵深方向的发展, 但二者仍有许多方面是相通的。因此可以将大系统控制理论的某些思想应用到智能控制系统的设计中。

8. 神经网络理论

神经网络的研究已经有 30 多年的历史, 它的发展过程并不是一帆风顺的, 有高潮也有低潮, 有成绩也有挫折。近几年来研究表明, 神经网络在很多领域具有广阔的应用前景。它在智能控制中的应用是其中的一个重要方面。

正如前面已经提到的, 神经网络是介于符号推理与数值计算之间的一种数学工具。它具有很好的适应能力和学习能力, 因此它适合于用作智能控制的工具。从本质上看, 神经网络是一种不依赖模型的自适应函数估计器。给定一个输入, 可以得到一个输出, 但它并不依赖于模型, 即它并不需要知道输出和输入之间存在着怎样的数学关系。而通常的函数估计器是依赖于数学模型的, 这是它与神经网络的一个根本区别。当给定的输入并不是原来训练的输入时, 神经网络也能给出合适的输出, 即它具有插值功能或适应功能。人工智能专家系统在一定意义上也可将其看作为不依赖模型的估计器, 它将条件映射为相应的动作, 它也不依赖于模型。在这一点上它与神经网络有共同之处, 但是它采用的是符号处理方法, 它不适用于数值的方法, 也不能用硬件方法来实现。符号系统虽然也是随时间改变的, 但是它没有导数, 它不是一个动力学系统。当输入不是预先设计的情况时, 它不能

给出合适的输出。因而它不具备适应功能。在专家系统中,知识明显地表示为规则,而在神经网络中,知识是通过学习例子而分布地存储在网络中。正是由于这一点,神经网络具有很好的容错能力,当个别处理单元损坏时,对神经网络的整体行为只有很小的影响,而不会影响整个系统的正常工作。神经网络还特别适合于用来进行模式分类,因而它可用于基于模式分类的学习控制。

神经网络也是一种可以训练的非线性动力学系统,因而它呈现非线性动力学系统的许多特性,如李雅普诺夫稳定性、平衡点、极限环、平衡吸引子、混沌现象等。这些也都是在用神经网络组成智能控制系统时必须研究的问题。

9. 模糊集合论

自 1965 年 L. A. 扎德(Zadeh)提出了模糊集合的概念以来,模糊集合理论发展十分迅速,并在许多领域中获得了应用。它在控制中的应用尤为引人注目。模糊系统不仅在工业控制中获得了广泛应用,而且也已扩展到其它领域,如地铁的自动化、照相镜头的自动聚焦、彩色电视的自动调节、冰箱的除霜、空调、洗衣机、吸尘器、交通信号灯及电梯的控制等。

由于模糊控制主要是模仿人的控制经验而不是依赖于控制对象的模型。因此模糊控制器实现了人的某些智能,因而它也是一种智能控制。

正如前面提到的,模糊集合理论是介于逻辑计算与数值计算之间的一种数学工具。它形式上是利用规则进行逻辑推理,但其逻辑取值可在 0 与 1 之间连续变化,采用数值的方法而非符号的方法进行处理。符号处理方法可以直截了当地用规则来表示结构性的知识,但是它却不能直接使用数值计算的工具,因而也不能用大规模集成电路来实现一个 AI 系统。而模糊系统可以兼具两者的优点,它可用数值的方法来表示结构性知识,从而用数值的方法来处理。因而可以用大规模集成电路来实现模糊系统。

与神经网络一样,模糊系统也可看成是一种不依赖于模型的自适应估计器。给定一个输入,便可得到一个合适的输出。它主要依赖于模糊规则和模糊变量的隶属度函数,而无需知道输出和输入之间的数学依存关系。

模糊系统也是一种可以训练的非线性动力学系统,因而也存在诸如稳定性等问题需要加以研究。

10. 优化理论

在学习控制系统中常常通过对系统性能的评判和优化来修改系统的结构和参数。在神经网络控制中也常常是根据使某种代价函数极小来选择网络的连接权系数。在分层递阶控制系统中,也是通过使系统的总熵最小来实现系统的优化设计。因此优化理论也是智能控制理论的一个主要内容。

在优化理论中新近发展了一种遗传算法(Genetic Algorithm——简称 GA 算法)它是一种全局随机寻优算法。它模仿生物进化的过程,来逐步达到最好的结果。这种优化算法也将在智能控制中发挥重要的作用。

参 考 文 献

- [1] Kosko B. Neural Networks and Fuzzy System—A Dynamical Systems Approach to Machine Intelligence. Prentice-Hall, 1992
- [2] Saridis G N. Knowledge Implementation; Structure of Intelligent Control Systems. IEEE International Symposium on Intelligent Control, 1987
- [3] Saridis G N. Self-Organizing Control of Stochastic Systems, Marcel Dekker, Inc., 1977
- [4] Mendel J M. Application of Artificial Intelligence Techniques to a Spacecraft Control Problem. In Self-Organizing Control Systems, Vol. 5 Rept. DAC-59328 Douglas Aircraft Co., 1966
- [5] Fu K S. Learning Control Systems and Intelligent Control Systems; An Intersection of Artificial Intelligence and Automatic Control. IEEE Trans. on AC, Feb. 1971
- [6] Saridis G N. Toward the Realization of Intelligent Controls. Proc. of the IEEE, 1979, 67(8)
- [7] Saridis G N, Graham J H. Linguistic Decision Schemata for Intelligent Robots, Automatica, 1984, 20(1)
- [8] Saridis G N. Intelligent Robotic Control. IEEE Trans. on Automatic Control, 1983, 28(5)
- [9] Valavanis K P, Saridis G N. Information Theoretic Modeling of Intelligent Robot Systems. IEEE Trans, on System, Man, and Cybernetics, 1988, 18(6)
- [10] Saridis G N. Entropy Formulation of Optimal and Adaptive Control. IEEE Trans. on Automatic Control, 1988, 33(8)
- [11] Wang F Y, Saridis G N. A Coordination Theory for Intelligent Machines. Automatica, 1990, 26(5)
- [12] Å Ström K J, Anton J J, Arzen K E. Expert Control. Automatica, 1986, 22(3)
- [13] Zhou Q J, Bai J K. An Intelligent Controller of Novel Design. Proc. of Multinational Instrumentation Conference, 1983
- [14] White D A, Sofge D A ed. Handbook of Intelligent Control. Van Nostrand Reinhold, 1992
- [15] 孙增圻,张再兴. 智能控制的理论与技术. 控制与决策, 1996, 11(1)
- [16] 张再兴,孙增圻. 关于专家控制. 信息与控制, 1995, 24(3)

第 2 章 模糊逻辑控制

2.1 概 述

2.1.1 模糊控制与智能控制

如第 1 章所述,由人作为控制器的控制系统是典型的智能控制系统,其中包含了人的高级智能活动。模糊控制在一定程度上模仿了人的控制,它不需要有准确的控制对象模型。因此它是一种智能控制的方法。

例如,一个操作员通过观察仪表显示对过程进行控制,仪表显示反映了过程的输出量。当操作员通过仪表观察到输出量发生变化时,他根据所积累的知识和操作经验,作出决策,并采取相应的控制动作。这是一个从过程变化到控制行动之间的映射关系。这个映射是通过操作员的决策来实现的,这个决策过程并不是通过精确的定量计算,而是依靠定性或模糊的知识。例如,若控制的过程是水箱中的水温,检测仪表给出的是精确量,譬如 80°C , 操作员将这个精确量转化为头脑中的概念量,比方说“温度偏高”,他使用这个概念与头脑中已有的控制经验和模式相匹配,得到“温度偏高应该加入较多冷却水”的推断,进而操作员需将“加入较多冷却水”这个模糊概念给出定量解释,譬如说加入冷却水的流量应为 $10\text{m}^3/\text{h}$, 然后按此定量值控制执行装置,从而完成了整个控制过程的一个循环。这里人采用了一种模糊的控制方法,其中包含了人的智能行为。显然人并不是按照某种控制算法加以精确的计算,而且人也不可能有这样的记忆和计算能力,能在极短时间内完成较为复杂的计算。

本章所要介绍的模糊控制即是模仿上述人的控制过程,其中包含了人的控制经验和知识。因而从这个意义上说,模糊控制也是一种智能控制。模糊控制方法既可用于简单的控制对象,也可用于复杂的过程。

2.1.2 模糊集合与模糊数学的概念

上面所举的人进行控制的例子中,“温度偏高”中的“偏高”、“加入较多冷却水”中的“较多”等都是些模糊的概念,而利用这些模糊概念最终却能实现稳定的控制。如何描述这些模糊的概念,并对它们进行分析、推理,这正是模糊集合与模糊数学所要解决的问题。

模糊集合理论的产生和发展到现在不过是 30 年的历史,但它已经逐步地渗入到自然科学和社会科学的各个领域,并且取得了引人注目的成果。笼统地说,模糊集合是一种特别定义的集合,它可用来描述模糊现象。有关模糊集合、模糊逻辑等的数学理论,称之为模糊数学。

模糊性也是一种不确定性,但它不同于随机性,所以模糊理论不同于概率论。模糊性通常是指对概念的定义以及语言意义的理解上的不确定性。例如,“老人”、“温度高”、“数量大”等所含的不确定性即为模糊性。可见,模糊性主要是人为的主观理解上的不确定性,

而随机性则主要反映的是客观上的自然的不确定性,或者是事件发生的偶然性。偶然性与模糊性具有本质上的不同,它们是不同的情况下的不确定性。例如,“明天有雨”的不确定性,是由今天的预测产生的,时间过去了,到明天就变成确定的了。再有“掷一下骰子是4点”的不确定性是根据掷之前推测发生的,实际做一下掷骰子实验,它就是确定的事件了。但是“老人”、“气温高”等的不确定性,即使时间过去了,即使做了实验,它仍然是不确定的,这是由语言意义模糊性的本质所确定的。

模糊集合是一种特别定义的集合,它与普通集合既有联系也有分别。对于普通集合来说,任何一个元素要么属于该集合,要么不属于,非此即彼,界限分明,决无模棱两可。而对于模糊集合来说,一个元素可以是既属于又不属于,亦此亦彼,界限模糊。例如,一个人到苹果园去摘苹果,如果规定比2两重的苹果算作“大”苹果,这是普通集合的概念。因此,若摘到一个2.5两的苹果,可以毫不犹豫地说这是一个“大”苹果,若摘到一个1.9两的苹果,也可以毫不犹豫地说它不是“大”苹果。这就是关于“大”的两值逻辑,是用精确的量作为边界来划分属于还是不属于该集合。如果规定差不多比2两重的苹果为“大”苹果,这是一个模糊集合的概念。这时若摘到一个2.5两的苹果,可以不加思索地算作“大”苹果。那么对于1.9两的苹果呢,这就需要人为的决定了。你可以说它不够“大”,但若这个苹果园中“大”的苹果不够多的话,将它勉强算作“大”苹果也不为过。这就是关于“大”的连续值逻辑,是用人为的量作为边界来划分属于还是不属于该集合。

从上述例子可以看到,模糊性是人们在社会交往和生产实践中经常使用的,它提供了定性与定量、主观与客观、模糊与清晰之间的一个人为折衷。它既不同于确定性,也不同于偶然性和随机性。概率论是研究随机现象的,模糊数学则是研究模糊现象的,两者都属于不确定性数学。应当特别注意的一点是,不可认为模糊数学是模糊的概念,它是完全精确的,它是借助定量的方法研究模糊现象的工具。

2.1.3 模糊控制的发展和应用概况

美国教授查德(L. A. Zadeh) 1965年首先提出了模糊集合的概念,由此开创了模糊数学及其应用的新纪元。模糊控制是模糊集合理论应用的一个重要方面。1974年英国教授马丹尼(E. H. Mamdani) 首先将模糊集合理论应用于加热器的控制,其后产生了许多应用的例子。其中比较典型的有:热交换过程的控制,暖水工厂的控制,污水处理过程控制,交通路口控制,水泥窑控制,飞船飞行控制,机器人控制,模型小车的停靠和转弯控制,汽车速度控制,水质净化控制,电梯控制,电流和核反应堆的控制,并且生产出了专用的模糊芯片和模糊计算机。在模糊控制的应用方面,日本走在了前列。日本在国内建立了专门的模糊控制研究所,日本仙台的一条地铁的控制系统采用了模糊控制的方法取得了很好的效果。日本还率先将模糊控制应用到家用电器产品的控制,如照相机、吸尘器、洗衣机等,模糊控制的应用在日本已经相当普及。

下面具体介绍几个典型的应用。

1. Sugeno 的模糊小车

M. Sugeno 设计了一个模型小车的模糊控制。这是模糊控制应用的一个非常令人感兴趣的例子。利用模糊控制的方法可使该小车沿着预先设定的弯曲的轨道停靠到车库指

定位置。模糊控制的规则是通过向有经验的驾驶员学习而获得的。

2. 模糊自动火车运行系统

这是日本日立公司所开发的系统。自 1987 年 2 月,该系统已成功地在日本仙台的都市地铁系统应用。在该系统中,目标评估模糊控制器对每一条可能的控制命令的性能加以预测,然后基于熟练的操作员的经验选择其中可能最好的一条命令加以执行。该模糊系统包含了两种规则库。一种是关于恒速控制,它要求火车启动后维持恒速前进,另一种是火车自动停站的控制,它调节火车的速度以准确地停靠在车站的指定位置。这些规则是基于对运行的安全性、乘坐的舒适性、停放的准确性、能耗及运行时间等综合性能的评价。每一种规则库均包含 12 条规则,规则的前件是上述的综合性能评价,其后件根据性能的满意程度来决定所采取的控制行动。每 100ms 控制一次,运行结果表明,该模糊控制系统在乘坐的舒适性、停靠的准确性、能耗、运行时间及鲁棒性等方面均优于常规的 PID 控制。

3. 模糊自动集装箱吊车操纵系统

在该系统中,主要的性能指标为安全性、停放精度、集装箱摆动及吊运时间等。其中包含了两个主要的操作:台车操作和线绳操作。每一操作包含两个功能级:决策级和动作级。该模糊自动操纵系统在日本的北九州港进行了实地试验。试验结果表明,由一个不熟练的操作员来操纵的该模糊系统其货物处理能力超过每小时 30 个集装箱,其操作性能、安全性、精确度等均可与非常有经验的操作员相媲美。

4. 模糊逻辑芯片和模糊计算机

第一个模糊逻辑芯片是 1985 年由 AT&T 贝尔实验室的 Togai 和 Watanabe 设计的。该模糊推理芯片可以并行地处理 16 条规则。它由四部分组成:规则集存储器、推理处理单元、控制器、输入输出电路。规则集存储器是用静态随机存储器实现的,从而可动态地改变规则集。该芯片在 16MHz 的时钟下的执行速度可达 250K FLIPS。现在正在开发基于该芯片的模糊逻辑加速器(FLA)。1989 年 3 月,美国北卡罗利纳微电子中心的 Watanabe 设计制造了世界上最快的模糊逻辑芯片,该芯片包含有 688000 个晶体管,速度达到 580K FLIPS。

日本的 Yamaskawa 设计了一个模糊计算机的雏形。该模糊计算机由一个模糊存储器、一个推理机、一个 MAX 块、一个清晰化块以及一个控制单元组成。模糊存储器存储语言模糊信息,推理机实现 MAX 和 MIN 操作。该模糊计算机能以高达 10M FLIPS 的速度处理模糊信息,它朝着真正的工业应用以及常识知识处理方向迈出了十分重要的一步。

模糊控制无论从理论和应用方面均已取得了很大的进展,但与常规控制理论相比,仍然显得很不成熟。当已知系统的模型时,已有比较成熟的常规控制理论和方法来分析和设计系统。但是目前尚未建立起有效的方法来分析和设计模糊系统,它还主要依靠经验和试凑。因此现在有许多人正在进行研究,试图把许多常规控制的理论和概念推广到模糊控制系统,如能控性、稳定性等。近来的另外一个研究方向则是如何使模糊控制器具有学习能力。在这方面,模糊逻辑与神经网络相结合是一个值得注意的动向,两者的结合既可以模拟人的控制功能,又可以如人那样具有较强的对环境变化的适应能力和学习能力。这是一个很有前途的发展方向。

2.2 模糊集合及其运算

2.2.1 模糊集合的定义及表示方法

上一节介绍了模糊性的概念。例如到苹果园去摘“大苹果”，这里“大苹果”便是一个模糊的概念。如果将“大苹果”看作是一个集合，则“大苹果”便是一个模糊集合。如前所述，若认为差不多比 2 两重的苹果称之为“大苹果”，那么，2.5 两的苹果应毫无疑问地属于“大苹果”，如对此加以量化，则可设其属于的程度为 1。2.1 两苹果属于“大苹果”的程度譬如说为 0.7，2 两苹果属于的程度为 0.5，1.9 两的苹果属于的程度为 0.3 等等。以后称属于的程度为隶属度函数，其值可在 0~1 之间连续变化。可见，隶属度函数反映了模糊集合中的元素属于该集合的程度。若模糊集合“大苹果”用大写字母 A 表示，隶属度函数用 μ 表示。 A 中的元素用 x 表示，则 $\mu_A(x)$ 便表示 x 属于 A 的隶属度，对上面的数值例子可写成

$$\mu_A(2.5) = 1, \mu_A(2.1) = 0.7, \mu_A(2.0) = 0.5, \mu_A(1.9) = 0.3, \dots$$

隶属度函数也可用图形来描述，如图 2.1 所示。

若将“大苹果”看成普通集合，即硬性规定凡比 2 两重的都算“大苹果”小于 2 两都不算“大苹果”，则其隶属度函数如图 2.1 中的虚线所示。它仅取两个值 $\{0, 1\}$ 。可见，普通集合是模糊集合的一个特例。由此，不难给出如下的关于模糊集合的定义。

定义：给定论域 X ， $A = \{x\}$ 是 X 中的模糊集合的含义是

$$\mu_A: X \rightarrow [0, 1]$$

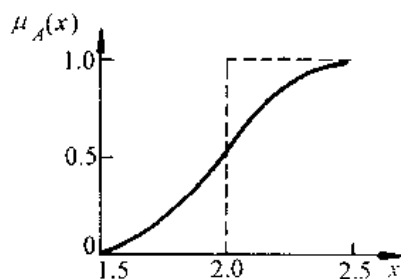


图 2.1 模糊集合“大苹果”的隶属度函数

这样的隶属度函数表示其特征的集合。若 $\mu_A(x)$ 接近 1，表示 x 属于 A 的程度高， $\mu_A(x)$ 接近 0，表示 x 属于 A 的程度低。

在上面的定义中，论域 X 指的是所讨论的事物的全体。如在摘苹果的例子中，论域 X 指的是重量 $X > 0$ 的全体。

模糊集合有很多种表示方法，最根本是要将它所包含的元素及相应的隶属度函数表示出来。因此它可用如下的序偶形式来表示：

$$A = \{(x, \mu_A(x)) | x \in X\}$$

也可表示成如下更紧凑的形式

$$A = \begin{cases} \int_X \frac{\mu_A(x)}{x} & X \text{ 连续} \\ \sum_{i=1}^n \frac{\mu_A(x_i)}{x_i} & X \text{ 离散} \end{cases}$$

下面再举两个模糊集合的例子。

例 2.1 在整数 $1, 2, \dots, 10$ 组成的论域中，即论域 $X = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ ，

设 A 表示模糊集合“几个”。并设各元素的隶属度函数依次为 $\mu_A(X) = \{0, 0, 0.3, 0.7, 1, 1, 0.7, 0.3, 0, 0\}$, 这里论域 X 是离散的, 则 A 可表示为

$$A = \{(x, \mu_A(x)) | x \in X\} \\ = \{(1, 0), (2, 0), (3, 0.3), (4, 0.7), (5, 1), (6, 1), (7, 0.7), (8, 0.3), (9, 0), (10, 0)\}$$

或者

$$A = \sum_{i=1}^{10} \frac{\mu_A(x_i)}{x_i} \\ = \frac{0}{1} + \frac{0}{2} + \frac{0.3}{3} + \frac{0.7}{4} + \frac{1}{5} + \frac{1}{6} + \frac{0.7}{7} + \frac{0.3}{8} + \frac{0}{9} + \frac{0}{10}$$

例 2.2 若以年龄为论域, 并设 $X = [0, 200]$ 。设 O 表示模糊集合“年老”, Y 表示模糊集合“年青”。已知“年老”和“年青”的隶属度函数分别为

$$\mu_O(x) = \begin{cases} 0 & 0 \leq x \leq 50 \\ \frac{1}{1 + \left(\frac{5}{x-50}\right)^2} & 50 < x \leq 200 \end{cases}$$

$$\mu_Y(x) = \begin{cases} 1 & 0 \leq x \leq 25 \\ \frac{1}{1 + \left(\frac{x-25}{5}\right)^2} & 25 < x \leq 200 \end{cases}$$

其隶属度函数曲线如图 2.2 所示。

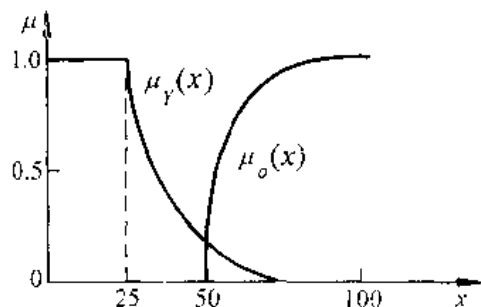


图 2.2 “年青”和“年老”的隶属度函数

这里论域 X 是连续的, 因而模糊集合 O 可表示为

$$O = \{(x, 0) | 0 \leq x \leq 50\} + \left\{ \left[x, \frac{1}{1 + \left(\frac{5}{x-50}\right)^2} \right] \middle| 50 < x \leq 200 \right\}$$

或者

$$O = \int_{0 \leq x \leq 50} \frac{0}{x} + \int_{50 < x \leq 200} \frac{\left[1 + \left(\frac{5}{x-50}\right)^2 \right]^{-1}}{x}$$

模糊集合 Y 可以表示为

$$Y = \{(x, 1) | 0 \leq x \leq 25\} + \left\{ \left(x, \frac{1}{1 + \left(\frac{x-25}{5} \right)^2} \right) \middle| 25 < x \leq 200 \right\}$$

或者

$$Y = \int_{0 \leq x \leq 25} \frac{1}{x} + \int_{25 < x \leq 200} \frac{\left[1 + \left(\frac{x-25}{5} \right)^2 \right]^{-1}}{x}$$

为了以后叙述方便,下面再介绍几个有关名词术语。

1. **台(support)集合**。它定义为

$$A_s = \{x | \mu_A(x) > 0\}$$

其意义为论域 X 中所有使 $\mu_A(x) > 0$ 的 x 的全体。如在例 2.1 中,模糊集合“几个”的台集合为

$$A_s = \{3, 4, 5, 6, 7, 8\}$$

在例 2.2 中,台集合为

$$A_s = \{x | 50 < x \leq 200\}$$

显然,台集合为普通集合,即

$$\mu_{A_s} = \begin{cases} 1 & x \in A_s \\ 0 & x \notin A_s \end{cases}$$

今后,模糊集合可只在它的台集合上加以表示。如例 2.1 中的“几个”可表示为

$$A = \text{“几个”} = \frac{0.3}{3} + \frac{0.7}{4} + \frac{1}{5} + \frac{1}{6} + \frac{0.7}{7} + \frac{0.3}{8}$$

例 2.2 中的“年老”可表示为

$$O = \text{“年老”} = \int_{50 < x \leq 200} \frac{\left[1 + \left(\frac{x-50}{5} \right)^2 \right]^{-1}}{x}$$

2. **α 截集**。它定义为

$$A_\alpha = \{x | \mu_A(x) > \alpha\} \quad \alpha \in [0, 1)$$

$$A_\alpha = \{x | \mu_A(x) \geq \alpha\} \quad \alpha \in (0, 1]$$

A_α 和 A_α 分别称为模糊集合 A 的强 α 截集和弱 α 截集。显然, α 截集也是普通集合,且

$$A_s = A_\alpha |_{\alpha=0}$$

3. **正则(normal)模糊集合**。如果

$$\max_{x \in X} \mu_A(x) = 1$$

则称 A 为正则模糊集合。

4. **凸(convex)模糊集合**。如果

$$\mu_A(\lambda x_1 + (1 - \lambda)x_2) \geq \min(\mu_A(x_1), \mu_A(x_2))$$

$$x_1, x_2 \in X, \lambda \in [0, 1]$$

则称 A 为凸模糊集合。

5. **分界点(crossover point)**。使得 $\mu_A(x) = 0.5$ 的点 x 称为模糊集合 A 的分界点。

6. **单点模糊集合(singleton)**。在论域 X 中,若模糊集合的台集合仅为一个点,且在该

点的隶属度函数 $\mu_A=1$, 则称 A 为单点模糊集合。

2.2.2 模糊集合的基本运算

1. 模糊集合的相等

若有两个模糊集合 A 和 B , 对于所有的 $x \in X$, 均有 $\mu_A(x) = \mu_B(x)$, 则称模糊集合 A 与模糊集合 B 相等, 记作 $A=B$ 。

2. 模糊集合的包含关系

若有两个模糊集合 A 和 B , 对于所有的 $x \in X$, 均有 $\mu_A(x) \leq \mu_B(x)$, 则称 A 包含于 B 或 A 是 B 的子集, 记作 $A \subseteq B$ 。

3. 模糊空集

若对所有 $x \in X$, 均有 $\mu_A(x) = 0$, 则称 A 为模糊空集, 记作 $A = \emptyset$ 。

4. 模糊集合的并集

若有三个模糊集合 A, B 和 C , 对于所有的 $x \in X$, 均有

$$\mu_C(x) = \mu_A(x) \vee \mu_B(x) = \max[\mu_A(x), \mu_B(x)]$$

则称 C 为 A 与 B 的并集, 记为 $C = A \cup B$ 。

5. 模糊集合的交集

若有三个模糊集合 A, B 和 C , 对于所有的 $x \in X$, 均有

$$\mu_C(x) = \mu_A(x) \wedge \mu_B(x) = \min[\mu_A(x), \mu_B(x)]$$

则称 C 为 A 与 B 的交集, 记为 $C = A \cap B$ 。

6. 模糊集合的补集

若有两个模糊集合 A 与 B , 对于所有的 $x \in X$, 均有

$$\mu_B(x) = 1 - \mu_A(x)$$

则称 B 为 A 的补集, 记为 $B = \bar{A}$ 。

7. 模糊集合的直积(Cartesian product)。

若有两个模糊集合 A 和 B , 其论域分别为 X 和 Y , 则定义在积空间 $X \times Y$ 上的模糊集合 $A \times B$ 为 A 和 B 的直积, 其隶属度函数为

$$\mu_{A \times B}(x, y) = \min[\mu_A(x), \mu_B(y)]$$

或者

$$\mu_{A \times B}(x, y) = \mu_A(x) \mu_B(y)$$

两个模糊集合直积的概念可以很容易推广到多个集合。

例 2.3 设论域 $X = \{x_1, x_2, x_3, x_4\}$ 以及模糊集合

$$A = \frac{1}{x_1} + \frac{0.8}{x_2} + \frac{0.4}{x_3} + \frac{0.5}{x_4}$$

$$B = \frac{0.9}{x_1} + \frac{0.4}{x_2} + \frac{0.7}{x_4}$$

求 $A \cup B, A \cap B, \bar{A}$ 和 \bar{B} 。

根据上面的定义, 容易求得

$$A \cup B = \frac{1 \vee 0.9}{x_1} + \frac{0.8 \vee 0.4}{x_2} + \frac{0.4 \vee 0}{x_3} + \frac{0.5 \vee 0.7}{x_4} = \frac{1}{x_1} + \frac{0.8}{x_2} + \frac{0.4}{x_3} + \frac{0.7}{x_4}$$

$$A \cap B = \frac{1 \wedge 0.9}{x_1} + \frac{0.8 \wedge 0.4}{x_2} + \frac{0.4 \wedge 0}{x_3} + \frac{0.5 \wedge 0.7}{x_4} = \frac{0.9}{x_1} + \frac{0.4}{x_2} + \frac{0.5}{x_4}$$

$$\bar{A} = \frac{1-1}{x_1} + \frac{1-0.8}{x_2} + \frac{1-0.4}{x_3} + \frac{1-0.5}{x_4} = \frac{0.2}{x_2} + \frac{0.6}{x_3} + \frac{0.5}{x_4}$$

$$\bar{B} = \frac{1-0.9}{x_1} + \frac{1-0.4}{x_2} + \frac{1-0}{x_3} + \frac{1-0.7}{x_4} = \frac{0.1}{x_1} + \frac{0.6}{x_2} + \frac{1}{x_3} + \frac{0.3}{x_4}$$

2.2.3 模糊集合运算的基本性质

1. 分配律

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

2. 结合律

$$(A \cap B) \cap C = A \cap (B \cap C)$$

$$(A \cup B) \cup C = A \cup (B \cup C)$$

3. 交换律

$$A \cup B = B \cup A$$

$$A \cap B = B \cap A$$

4. 吸收律

$$(A \cap B) \cup A = A$$

$$(A \cup B) \cap A = A$$

5. 幂等律

$$A \cup A = A \quad A \cap A = A$$

6. 同一律

$A \cup X = X, A \cap X = A, A \cup \emptyset = A, A \cap \emptyset = \emptyset$, 其中 X 表示论域全集, \emptyset 表示空集。

7. 达·摩根律

$$\overline{(A \cup B)} = \bar{A} \cap \bar{B} \quad \overline{(A \cap B)} = \bar{A} \cup \bar{B}$$

8. 双重否定律

$$\bar{\bar{A}} = A$$

以上运算性质与普通集合的运算性质完全相同,但是在普通集合中成立的排中律和矛盾律对于模糊集合不再成立,即

$$A \cup \bar{A} \neq X \quad A \cap \bar{A} \neq \emptyset$$

9. α 截集到模糊集合的转换

$$A = \bigcup_{\alpha \in [0,1)} \alpha A_{\alpha} = \bigcup_{\alpha \in (0,1]} \alpha A_{\alpha}$$

即

$$\mu_A(x) = \sup_{\alpha \in [0,1)} [\alpha \wedge \mu_{A_{\alpha}}(x)] = \sup_{\alpha \in (0,1]} [\alpha \wedge \mu_{A_{\alpha}}(x)]$$

2.2.4 模糊集合的其它类型运算

在模糊集合的运算中,还常常用到其它类型的运算,下面列出主要的几种。

1. 代数和

若有三个模糊集合 A, B 和 C , 对于所有的 $x \in X$, 均有

$$\mu_C(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x)$$

则称 C 为 A 与 B 的代数和, 记为 $C = A \dot{+} B$. 上述说明也可简单表达为

$$A \dot{+} B \leftrightarrow \mu_{A \dot{+} B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x)$$

2. 代数积

$$A \cdot B \leftrightarrow \mu_{A \cdot B}(x) = \mu_A(x)\mu_B(x)$$

3. 有界和

$$A \oplus B \leftrightarrow \mu_{A \oplus B}(x) = \min\{1, \mu_A(x) + \mu_B(x)\}$$

4. 有界差

$$A \ominus B \leftrightarrow \mu_{A \ominus B}(x) = \max\{0, \mu_A(x) - \mu_B(x)\}$$

5. 有界积

$$A \odot B \leftrightarrow \mu_{A \odot B}(x) = \max\{0, \mu_A(x) + \mu_B(x) - 1\}$$

6. 强制和(drastic sum)

$$A \cup B \leftrightarrow \mu_{A \cup B}(x) = \begin{cases} \mu_A(x), & \mu_B(x) = 0 \\ \mu_B(x), & \mu_A(x) = 0 \\ 1, & \mu_A(x), \mu_B(x) > 0 \end{cases}$$

7. 强制积(drastic product)

$$A \cap B \leftrightarrow \mu_{A \cap B}(x) = \begin{cases} \mu_A(x), & \mu_B(x) = 1 \\ \mu_B(x), & \mu_A(x) = 1 \\ 0, & \mu_A(x), \mu_B(x) < 1 \end{cases}$$

2.3 模糊关系

在日常生活中经常听到诸如“ A 与 B 很相似”、“ X 比 Y 大很多”等描述模糊关系的语句。借助于模糊集合理论,可以定量地来描述这些模糊关系。

2.3.1 模糊关系的定义及表示

定义: n 元模糊关系 R 是定义在直积 $X_1 \times X_2 \times \cdots \times X_n$ 上的模糊集合, 它可表示为

$$\begin{aligned} R_{X_1 \times X_2 \times \cdots \times X_n} &= \{((x_1, x_2, \cdots, x_n), \mu_R(x_1, x_2, \cdots, x_n)) | (x_1, x_2, \cdots, x_n) \\ &\quad \in X_1 \times X_2 \times \cdots \times X_n\} \\ &= \int_{X_1 \times X_2 \times \cdots \times X_n} \mu_R(x_1, x_2, \cdots, x_n) / (x_1, x_2, \cdots, x_n) \end{aligned}$$

以后用得较多的是 $n=2$ 时的模糊关系。

例 2.4 设 X 是实数集合, 并 $x, y \in X$, 对于“ y 比 x 大得多”的模糊关系 R , 其隶属度函数可以表示为

$$\mu_R(x, y) = \begin{cases} 0 & , x \geq y \\ \frac{1}{1 + \left(\frac{10}{y-x}\right)^2} & , x < y \end{cases}$$

而对于“ x 和 y 大致相等”这样的模糊关系 R , 其隶属度函数可表示为

$$\mu_R(x, y) = e^{-\alpha |x-y|} \quad \alpha > 0$$

因为模糊关系也是模糊集合, 所以它可用如上所述的表示模糊集合的方法来表示。此外, 有些情况下, 它还可以用矩阵和图的形式来更形象地加以描述。

当 $X = \{x_1, x_2, \dots, x_n\}$, $Y = \{y_1, y_2, \dots, y_m\}$ 是有限集合时, 定义在 $X \times Y$ 上的模糊关系 R 可用如下的 $n \times m$ 阶矩阵来表示。

$$R = \begin{bmatrix} \mu_R(x_1, y_1) & \mu_R(x_1, y_2) & \cdots & \mu_R(x_1, y_m) \\ \mu_R(x_2, y_1) & \mu_R(x_2, y_2) & \cdots & \mu_R(x_2, y_m) \\ \vdots & \vdots & & \vdots \\ \mu_R(x_n, y_1) & \mu_R(x_n, y_2) & \cdots & \mu_R(x_n, y_m) \end{bmatrix}$$

这样的矩阵称为模糊矩阵, 由于其元素均为隶属度函数, 因此它们均在 $[0, 1]$ 中取值。

若用图来表示模糊关系时, 则将 x_i, y_j 作为节点, 在 x_i 到 y_j 的连线上标上 $\mu_R(x_i, y_j)$ 的值, 这样的图便称为模糊图。

例 2.5 设 X 为家庭中的儿子和女儿, Y 为家庭成员中的父亲和母亲, 对于“子女与父母长得相似”的模糊关系 R , 可以用如下的模糊矩阵和如图 2.3 所示的模糊图来表示。

$$R = \begin{matrix} & \begin{matrix} \text{父} & \text{母} \end{matrix} \\ \begin{matrix} \text{子} \\ \text{女} \end{matrix} & \begin{bmatrix} 0.8 & 0.3 \\ 0.3 & 0.6 \end{bmatrix} \end{matrix}$$

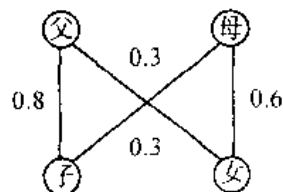


图 2.3 例 2.5 的模糊图

下面再给出几个特殊的模糊关系及其矩阵表示。

• 逆模糊关系

$$R^C \leftrightarrow \mu_{R^C}(y, x) = \mu_R(x, y)$$

• 恒等关系

$$I \leftrightarrow \mu_I(x, y) = \begin{cases} 1 & x = y \\ 0 & x \neq y \end{cases}$$

• 零关系

$$O \leftrightarrow \mu_O(x, y) = 0$$

• 全称关系 (universe relation)

$$E \leftrightarrow \mu_E(x, y) = 1$$

若以上几个模糊关系均用模糊矩阵表示, 则

$$R^C = R^T \quad (R \text{ 的转置})$$

$$I = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}_{n \times n}, O = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{n \times m}, E = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}_{n \times m}$$

2.3.2 模糊关系的合成

如前所述,模糊关系是定义在直积空间上的模糊集合,所以它也遵从一般模糊集合的运算规则,例如

包含: $R \subseteq S \leftrightarrow \mu_R(x, y) \leq \mu_S(x, y)$

并: $R \cup S \leftrightarrow \mu_{R \cup S}(x, y) = \mu_R(x, y) \vee \mu_S(x, y)$

交: $R \cap S \leftrightarrow \mu_{R \cap S}(x, y) = \mu_R(x, y) \wedge \mu_S(x, y)$

补: $\bar{R} \leftrightarrow \mu_{\bar{R}}(x, y) = 1 - \mu_R(x, y)$

下面介绍模糊关系的合成运算,它在模糊控制中有很重要的应用。

设 X, Y, Z 是论域, R 是 X 到 Y 的一个模糊关系, S 是 Y 到 Z 的一个模糊关系, 则 R 到 S 的合成 T 也是一个模糊关系, 记为 $T = R \circ S$, 它具有隶属度

$$\mu_{R \circ S}(x, z) = \bigvee_{y \in Y} (\mu_R(x, y) * \mu_S(y, z))$$

其中 \vee 是并的符号, 它表示对所有 y 取极大值或上界值, “ $*$ ”是二项积的符号, 因此上面的合成称为最大-星合成(max-star composition)。其中二项积算子“ $*$ ”可以定义为以下几种运算, 其中 $x, y \in [0, 1]$

交 $x \wedge y = \min\{x, y\}$

代数积 $x \cdot y = xy$

有界积 $x \odot y = \max\{0, x + y - 1\}$

强制积 $x \odot y = \begin{cases} x & y = 1 \\ y & x = 1 \\ 0 & x, y < 1 \end{cases}$

若二项积采用求交运算, 则

$$R \circ S \leftrightarrow \mu_{R \circ S}(x, z) = \bigvee_{y \in Y} (\mu_R(x, y) \wedge \mu_S(y, z))$$

这时称为最大-最小合成(max-min composition), 这是最常用的一种合成方法。

当论域 X, Y, Z 为有限时, 模糊关系的合成可用模糊矩阵的合成来表示。设

$$R = (r_{ij})_{n \times m} \quad S = (s_{jk})_{m \times l} \quad T = (t_{ik})_{n \times l}$$

则

$$t_{ik} = \bigvee_{j=1}^m (r_{ij} \wedge s_{jk})$$

例 2.6 已知子女与父母的相似关系模糊矩阵为

$$R = \begin{matrix} & \begin{matrix} \text{父} & \text{母} \end{matrix} \\ \begin{matrix} \text{子} \\ \text{女} \end{matrix} & \begin{bmatrix} 0.8 & 0.3 \\ 0.3 & 0.6 \end{bmatrix} \end{matrix}$$

父母与祖父母的相似关系的模糊矩阵。

$$S = \begin{matrix} & \begin{matrix} \text{父} & \text{母} \end{matrix} \\ \begin{matrix} \text{祖父} & \text{祖母} \end{matrix} & \begin{bmatrix} 0.7 & 0.5 \\ 0.1 & 0.1 \end{bmatrix} \end{matrix}$$

要求子女与祖父母的相似关系模糊矩阵。

这是一个典型的模糊关系合成的问题。按最大-最小合成规则

$$\begin{aligned} T = R \circ S &= \begin{bmatrix} 0.8 & 0.3 \\ 0.3 & 0.6 \end{bmatrix} \circ \begin{bmatrix} 0.7 & 0.5 \\ 0.1 & 0.1 \end{bmatrix} \\ &= \begin{bmatrix} \max(0.8 \wedge 0.7, 0.3 \wedge 0.1) & \max(0.8 \wedge 0.5, 0.3 \wedge 0.1) \\ \max(0.3 \wedge 0.7, 0.6 \wedge 0.1) & \max(0.3 \wedge 0.5, 0.6 \wedge 0.1) \end{bmatrix} \\ &= \begin{matrix} & \begin{matrix} \text{祖父} & \text{祖母} \end{matrix} \\ \begin{matrix} \text{父} & \text{母} \end{matrix} & \begin{bmatrix} 0.7 & 0.5 \\ 0.3 & 0.3 \end{bmatrix} \end{matrix} \end{aligned}$$

下面列出一些从合成关系得出的一些基本性质。

- $R \circ I = I \circ R = R$
- $R \circ O = O \circ R = O$
- 一般情况 $R \circ S \neq S \circ R$
- $(R \circ S) \circ T = R \circ (S \circ T)$
- $R^{m+1} = R^m \circ R$
- $R^{m+n} = R^m \circ R^n$
- $(R^m)^n = R^{mn}$
- $R \circ (S \cup T) = (R \circ S) \cup (R \circ T)$
- $R \circ (S \cap T) = (R \circ S) \cap (R \circ T)$
- $S \subseteq T \rightarrow R \circ S \subseteq R \circ T$

2.4 模糊逻辑与近似推理

2.4.1 语言变量

语言是人们进行思维和信息交流的重要工具。语言可分为两种：自然语言和形式语言。人们日常所用的语言属自然语言。自然语言的特点是语义丰富、灵活，同时具有模糊性，如“这朵花很美丽”、“他很年轻”、“小张的个子很高”等。通常的计算机语言是形式语言。形式语言有严格的语法规则和语义，不存在任何的模糊性和歧义。带模糊性的语言称为模糊语言，如长、短、大、小、高、矮、年轻、年老、较老、很老、极老等。在模糊控制中，关于误差的模糊语言常见的有：正大、正中、正小、正零、负零、负小、负中、负大等。

语言变量是自然语言中的词或句，它的取值不是通常的数，而是用模糊语言表示的模糊集合。例如，若“年龄”看成是一个模糊语言变量，则它的取值不是具体岁数，而是诸如“年幼”、“年轻”、“年老”等用模糊语言表示的模糊集合。

L. A. 查德(Zadah)为语言变量给出了如下的定义：

语言变量由一个 5 元组 $(x, T(x), U, G, M)$ 来表征。其中 x 是变量的名称, U 是 X 的论域, $T(x)$ 是语言变量值的集合, 每个语言变量值是定义在论域 U 上一个模糊集合, G 是语法规则, 用以产生语言变量 x 的值的名称, M 是语义规则, 用于产生模糊集合的隶属度函数。

例如, 若定义“速度”为语言变量, 则 $T(\text{速度})$ 可能为

$$T(\text{速度}) = \{\text{慢, 适中, 快, 很慢, 稍快} \dots\}$$

上述每个模糊语言如慢、适中等是定义在论域 U 上的一个模糊集合。设论域 $U = [0, 160]$, 则可认为大致低于 60km/h 为“慢”, 80km/h 左右为“适中”, 大于 100km/h 以上为“快”, ……。这些模糊集合可以用图 2.4 所示的隶属度函数图来描述。

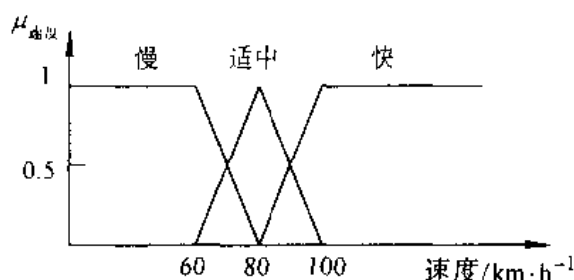


图 2.4 模糊语言变量“速度”的隶属度函数

如上所述, 每个模糊语言相当于一个模糊集合, 通常在模糊语言前面加上“极”、“非”、“相当”、“比较”、“略”、“稍微”的修饰词。其结果改变了该模糊语言的含义, 相应地隶属度函数也要改变。例如, 设原来的模糊语言为 A , 其隶属度函数为 μ_A , 则通常有

$$\begin{aligned} \mu_{\text{极}A} &= \mu_A^4, & \mu_{\text{非常}A} &= \mu_A^2, & \mu_{\text{相当}A} &= \mu_A^{1.25} \\ \mu_{\text{比较}A} &= \mu_A^{0.75}, & \mu_{\text{略}A} &= \mu_A^{0.5}, & \mu_{\text{稍微}A} &= \mu_A^{0.25} \end{aligned}$$

2.4.2 模糊蕴含关系

1. 模糊蕴含关系的直觉判据

在模糊控制中, 模糊控制规则实质上是模糊蕴含关系。在模糊逻辑中有很多种定义模糊蕴含的方法。我们必须针对控制的目的选择符合直觉判据的定义方法。

在近似推理中有两类最主要的模糊蕴含推理方式: 一类是广义的肯定式推理方式, 另一类是广义的否定式推理方式。

肯定式:

前提 1: x 是 A'

前提 2: 如果 x 是 A 则 y 是 B

结论: y 是 B'

否定式:

前提 1: y 是 B'

前提 2: 如果 x 是 A 则 y 是 B

结论: x 是 A'

其中 A, A', B, B' 均为模糊语言。横线上方是前提或条件,横线下是结论。表 2.1 和表 2.2 归纳了模糊蕴含应满足的直觉判据。

表 2.1 肯定式的直觉判据

	x 是 A' (前提 1)	y 是 B' (结论)
判据 1	x 是 A	y 是 B
判据 2-1	x 是非常 A	y 是非常 B
判据 2-2	x 是非常 A	y 是 B
判据 3-1	x 是略 A	y 是略 B
判据 3-2	x 是略 A	y 是 B
判据 4-1	x 是非 A	y 不定
判据 4-2	x 是非 A	y 是非 B

表 2.2 否定式的直觉判据

	y 是 B' (前提)	x 是 A' (结论)
判据 5	y 是非 B	x 是非 A
判据 6	y 是非(非常 B)	x 是非(非常 A)
判据 7	y 是非(略 B)	x 是非(略 A)
判据 8-1	y 是 B	x 不定
判据 8-2	y 是 B	x 是 A

从表中我们注意到,在模糊蕴含中“ x 是 A ”与“ y 是 B ”之间的因果关系并不要求非常严格,即直觉判据 2-2 和 3-2 还是可以接受的。判据 4-2 相当于:如果 x 是 A 则 y 是 B , 否则 y 是非 B 。虽然在形式逻辑中这样的关系是不适用的,但在日常推理中我们常常希望有这样的因果关系。判据 8-2 也是同样的情况。

2. 模糊蕴含关系的运算方法。

在上述两类模糊蕴含推理方法中,模糊前题 2:“如果 x 是 A 则 y 是 B ”表示了 A 与 B 之间的模糊蕴含关系,记为 $A \rightarrow B$ 。在普通的形式逻辑中 $A \rightarrow B$ 有严格的定义。但在模糊逻辑中 $A \rightarrow B$ 不是普通逻辑的简单推广。很多人对此进行了研究,并提出了许多定义的方法,在模糊逻辑控制中,常常见到如下几种模糊蕴含关系的运算方法。

(1) 模糊蕴含最小运算(Mamdani)

$$R_C = A \rightarrow B = A \times B = \int_{X \times Y} \mu_A(x) \wedge \mu_B(y) / (x, y)$$

(2) 模糊蕴含积运算(Larsen)

$$R_P = A \rightarrow B = A \times B = \int_{X \times Y} \mu_A(x) \mu_B(y) / (x, y)$$

(3) 模糊蕴含算术运算(Zadeh)

$$R_z = A \rightarrow B = (\bar{A} \times Y) \oplus (X \times B)$$

$$= \int_{X \times Y} 1 \wedge (1 - \mu_A(x) + \mu_B(y)) / (x, y)$$

(4) 模糊蕴含的最大最小运算(Zadeh)

$$R_m = A \rightarrow B = (A \times B) \cup (\bar{A} \times Y)$$

$$= \int_{X \times Y} (\mu_A(x) \wedge \mu_B(y)) \vee (1 - \mu_A(x)) / (x, y)$$

(5) 模糊蕴含的布尔运算

$$R_b = A \rightarrow B = (\bar{A} \times Y) \cup (X \times B)$$

$$= \int_{X \times Y} (1 - \mu_A(x)) \vee \mu_B(y) / (x, y)$$

(6) 模糊蕴含的标准法运算(1)

$$R_s = A \rightarrow B = A \times Y \rightarrow X \times B$$

$$= \int_{X \times Y} (\mu_A(x) > \mu_B(y)) / (x, y)$$

其中

$$\mu_A(x) > \mu_B(y) = \begin{cases} 1 & \mu_A(x) \leq \mu_B(y) \\ 0 & \mu_A(x) > \mu_B(y) \end{cases}$$

(7) 模糊蕴含的标准法运算(2)

$$R_d = A \rightarrow B = A \times Y \rightarrow X \times B$$

$$= \int_{X \times Y} (\mu_A(x) >> \mu_B(y)) / (x, y)$$

其中

$$\mu_A(x) >> \mu_B(y) = \begin{cases} 1 & \mu_A(x) \leq \mu_B(y) \\ \frac{\mu_B(y)}{\mu_A(x)} & \mu_A(x) > \mu_B(y) \end{cases}$$

2.4.3 近似推理

上面列举了7种模糊蕴含关系的运算方法,它们均可以应用于广义肯定式和广义否定式的模糊推理中。模糊推理也即近似推理,这两个术语将不加区分地混用。

对于广义肯定式推理,结论 B' 是根据模糊集合 A' 和模糊蕴含关系 $A \rightarrow B$ 的合成推出来的,因此可得如下的近似推理关系

$$B' = A' \circ (A \rightarrow B) = A' \circ R$$

其中 R 为模糊蕴含关系,它可采用上面所列举的任何一种运算方法,“ \circ ”是合成运算符。假定模糊集合 A' 具有如下形式

$$A = \int_X \mu_A(x) / x$$

$$\text{非常 } A = A^2 = \int_X \mu_A^2(x) / x$$

$$\text{略 } A = A^{0.5} = \int_X \mu_A^{0.5}(x) / x$$

$$\text{非 } A = \overline{A} = \int_X (1 - \mu_A(x)) / x$$

根据上述 A' 的各种表示方式, 利用近似推理公式可以推出相应的 B' 。

类似地, 对于广义否定式推理, 有如下的近似推理公式:

$$A' = (A \rightarrow B) \circ B' = R \circ B'$$

其中 R 为模糊蕴含关系, “ \circ ”是合成运算符, B' 是模糊集合, 它具有如下形式:

$$\text{非 } B = \overline{B} = \int_Y (1 - \mu_B(y)) / y$$

$$\text{非(非常 } B) = \overline{\text{非常 } B} = \int_Y (1 - \mu_B^2(y)) / y$$

$$\text{非(略 } B) = \overline{\text{略 } B} = \int_Y (1 - \mu_B^{0.5}(y)) / y$$

$$B = \int_Y \mu_B(y) / y$$

同样, 根据上述各种情况的 B' 可推出相应的 A' 。

下面以模糊蕴含积运算规则 R_p 为例进行模糊推理计算。

1. 广义肯定式推理

设 $A' = A^\alpha$ ($\alpha > 0$), 相应的推理结果 B' 可以推得如下:

$$B' = A' \circ R_p = \int_X \mu_A^\alpha(x) / x \circ \int_{X \times Y} \mu_A(x) \mu_B(y) / (x, y)$$

假定采用最大-最小合成法, 可求得

$$\begin{aligned} \mu_{B'}(y) &= \bigvee_{x \in X} (\mu_A^\alpha(x) \wedge \mu_A(x) \mu_B(y)) = \sup_{x \in X} \min \{ \mu_A^\alpha(x), \mu_A(x) \mu_B(y) \} \\ &= \sup_{x \in X} S_p(\mu_A^\alpha(x)) \end{aligned}$$

其中

$$S_p(\mu_A^\alpha(x)) = \min \{ \mu_A^\alpha(x), \mu_A(x) \mu_B(y) \}$$

下面分别讨论 A' 为不同值的情况。

(1) $A' = A$

$$\begin{aligned} \mu_{B'}(y) &= \sup_{x \in X} \min \{ \mu_A(x), \mu_A(x) \mu_B(y) \} \\ &= \sup_{x \in X} \mu_A(x) \mu_B(y) \\ &= \mu_B(y) \end{aligned}$$

(2) $A' = A^2$

$$\begin{aligned} \mu_{B'}(y) &= \sup_{x \in X} \min \{ \mu_A^2(x), \mu_A(x) \mu_B(y) \} \\ &= \sup_{x \in X} S_p(\mu_A^2(x)) \end{aligned}$$

图 2.5 画出了当 $\mu_B(y)$ 为参变量时的 $S_p(\mu_A^2(x))$ 的曲线图, 图中同时画出了 $\mu_A^2(x)$ 及 $\mu_A(x) \mu_B(y)$ 的曲线, 进而可求得

$$\mu_{B'}(y) = \sup_{x \in X} S_p(\mu_A^2(x)) = \mu_B(y)$$

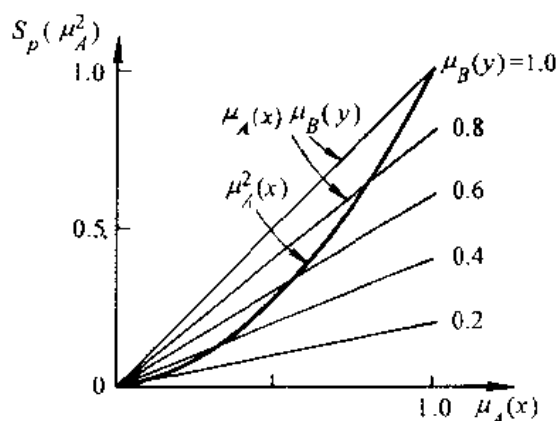


图 2.5 利用模糊蕴含积运算的广义肯定式推理: $A' = A^2$

(3) $A' = A^{0.5}$

$$\begin{aligned}\mu_{B'}(y) &= \sup_{x \in X} \min \{ \mu_A^{0.5}(x), \mu_A(x) \mu_B(y) \} \\ &= \sup_{x \in X} \mu_A(x) \mu_B(y) \\ &= \mu_B(y)\end{aligned}$$

(4) $A' = \text{非 } A = \bar{A}$

$$\begin{aligned}\mu_{B'}(y) &= \sup_{x \in X} \min \{ 1 - \mu_A(x), \mu_A(x) \mu_B(y) \} \\ &= \sup_{x \in X} S_p(1 - \mu_A(x))\end{aligned}$$

图 2.6 画出了当 $\mu_B(y)$ 为参变量时的 $S_p(1 - \mu_A(x))$ 的曲线图。

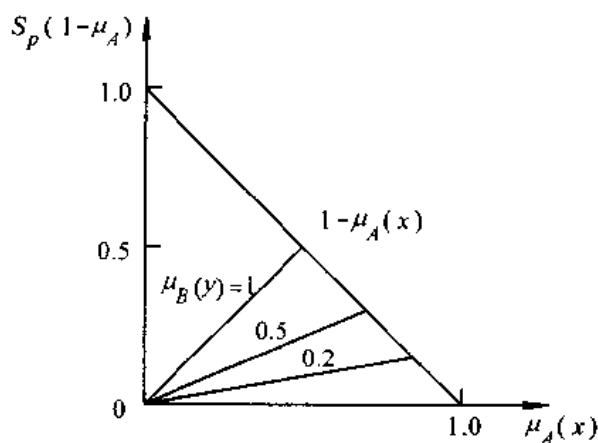


图 2.6 利用模糊蕴含积运算的广义肯定式推理: $A' = \text{非 } A$

根据图 2.6, 通过图解可以求得

$$\mu_{B'}(y) = \sup_{x \in X} S_p(1 - \mu_A(x)) = \frac{\mu_B(y)}{1 + \mu_B(y)}$$

2. 广义否定式推理

设 $B' = \text{非 } B^a (a > 0)$, 相应的推理结果 A' 可以推得如下:

$$A' = R_P \circ (\text{非 } B^*) = \int_{x \times Y} \mu_A(x) \mu_B(y) / (x, y) \circ \int_Y (1 - \mu_B^*(y)) / y$$

假定采用最大-最小合成法,可求得

$$\begin{aligned} \mu_{A'}(x) &= \sup_{y \in Y} \min \{ \mu_A(x) \mu_B(y), 1 - \mu_B^*(y) \} \\ &= \sup_{y \in Y} S_i(1 - \mu_B^*(y)) \end{aligned}$$

其中 $S_i(1 - \mu_B^*(y)) = \min \{ \mu_A(x) \mu_B(y), 1 - \mu_B^*(y) \}$

下面分别讨论 B' 为不同值的情况。

(1) $B' = \text{非 } B$

$$\begin{aligned} \mu_{A'}(x) &= \sup_{y \in Y} \min \{ \mu_A(x) \mu_B(y), 1 - \mu_B(y) \} \\ &= \sup_{y \in Y} S_i(1 - \mu_B(y)) \end{aligned}$$

图 2.7 画出了当 $\mu_A(x)$ 为参变量时的 $S_i(1 - \mu_B(y))$ 的曲线。根据图解可以求得

$$\mu_{A'}(x) = \sup_{y \in Y} S_i(1 - \mu_B(y)) = \frac{\mu_A(x)}{1 + \mu_A(x)}$$

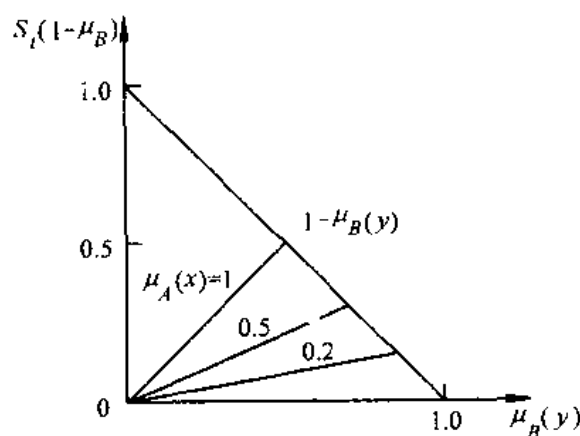


图 2.7 利用模糊蕴含积运算的广义否定式推理: $B' = \text{非 } B$

(2) $B' = \text{非 } B^2$

画出 $\mu_A(x)$ 为参变量时的 $S_i(1 - \mu_B^2(y))$ 的曲线图如图 2.8 所示,进而求得

$$\begin{aligned} \mu_{A'}(x) &= \sup_{y \in Y} \min \{ \mu_A(x) \mu_B(y), 1 - \mu_B^2(y) \} \\ &= \sup_{y \in Y} S_i(1 - \mu_B^2(y)) \\ &= \frac{\mu_A(x) \sqrt{\mu_A^2(x) + 4} - \mu_A^2(x)}{2} \end{aligned}$$

(3) $B' = \text{非 } B^{0.5}$

画出 $\mu_A(x)$ 为参变量时的 $S_i(1 - \mu_B^{0.5}(y))$ 的曲线图如图 2.9 所示,进而求得

$$\begin{aligned} \mu_{A'}(x) &= \sup_{y \in Y} \min \{ \mu_A(x) \mu_B(y), 1 - \mu_B^{0.5}(y) \} \\ &= \sup_{y \in Y} S_i(1 - \mu_B^{0.5}(y)) \end{aligned}$$

$$= \frac{2\mu_A(x) + 1 - \sqrt{4\mu_A(x) + 1}}{2\mu_A(x)}$$

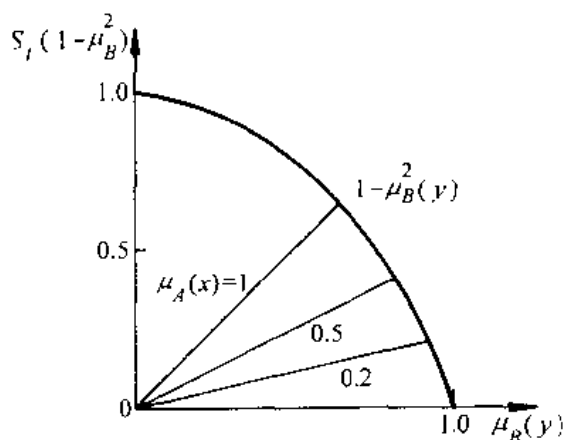


图 2.8 利用模糊蕴含积运算的广义否定式推理: $B' = \text{非 } B^2$

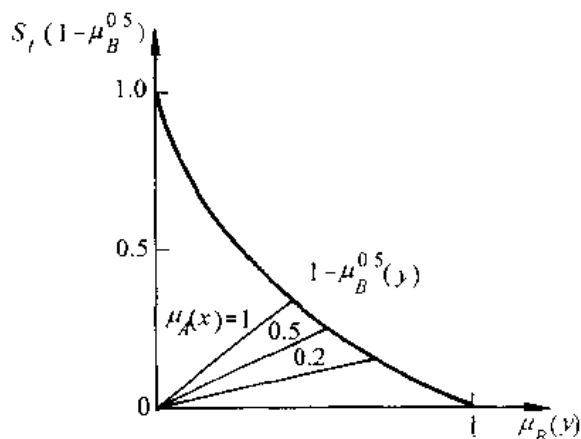


图 2.9 利用模糊蕴含积运算的广义否定式推理: $B' = \text{非 } B^{0.5}$

(4) $B' = B$

这时容易求得

$$\begin{aligned}\mu_{A'}(x) &= \sup_{y \in Y} \min \{ \mu_A(x) \mu_B(y), \mu_B(y) \} \\ &= \sup_{y \in Y} \mu_A(x) \mu_B(y) \\ &= \mu_A(x)\end{aligned}$$

上面以 R_p 为例进行了推理计算, 仿照类似的方法可以对 $R_a, R_i, R_m, R_r, R_b, R_\Delta$ 也进行同样的推理计算。表 2.3 和表 2.4 给出了所有的推理计算结果。

表 2.3 广义肯定式的推理结果

	A	非常 A	略 A	非 A
R_c	μ_B	μ_B	μ_B	$0.5 \wedge \mu_B$
R_p	μ_B	μ_B	μ_B	$\frac{\mu_B}{1 + \mu_B}$
R_a	$\frac{1 + \mu_B}{2}$	$\frac{3 + 2\mu_B - \sqrt{5 + 4\mu_B}}{2}$	$\frac{\sqrt{5 + 4\mu_B} - 1}{2}$	1
R_m	$0.5 \vee \mu_B$	$\frac{3 - \sqrt{5}}{2} \vee \mu_B$	$\frac{\sqrt{5} - 1}{2} \vee \mu_B$	1
R_b	$0.5 \vee \mu_B$	$\frac{3 - \sqrt{5}}{2} \vee \mu_B$	$\frac{\sqrt{5} - 1}{2} \vee \mu_B$	1
R_i	μ_B	μ_B^2	$\sqrt{\mu_B}$	1
R_Δ	$\sqrt{\mu_B}$	$\mu_B^{\frac{2}{3}}$	$\mu_B^{\frac{1}{3}}$	1

表 2.4 广义否定式的推理结果

	B	非常 \bar{B}	略 \bar{B}	B
R_c	$0.5 \wedge \mu_A$	$\frac{\sqrt{5}-1}{2} \wedge \mu_A$	$\frac{3-\sqrt{5}}{2} \wedge \mu_A$	μ_A
R_p	$\frac{\mu_A}{1+\mu_A}$	$\frac{\mu_A \sqrt{\mu_A^2+4} - \mu_A^2}{2}$	$\frac{1+2\mu_A - \sqrt{1+4\mu_A}}{2\mu_A}$	μ_A
R_a	$1 - \frac{\mu_A}{2}$	$\frac{1-2\mu_A + \sqrt{1+4\mu_A}}{2}$	$\frac{3-\sqrt{1+\mu_A}}{2}$	1
R_m	$0.5 \vee (1-\mu_A)$	$(1-\mu_A) \vee \left(\frac{\sqrt{5}-1}{2} \wedge \mu_A \right)$	$\frac{3-\sqrt{5}}{2} \vee (1-\mu_A)$	$\mu_A \vee (1-\mu_A)$
R_b	$0.5 \vee (1-\mu_A)$	$\frac{\sqrt{5}-1}{2} \vee (1-\mu_A)$	$\frac{3-\sqrt{5}}{2} \vee (1-\mu_A)$	1
R_i	$1 - \mu_A$	$1 - \mu_A^2$	$1 - \sqrt{\mu_A}$	1
R_Δ	$\frac{1}{1+\mu_A}$	$\frac{\sqrt{4\mu_A^2+1}-1}{2\mu_A^2}$	$\frac{2+\mu_A - \sqrt{\mu_A^2+4\mu_A}}{2}$	1

将表 2.3 和表 2.4 的推理结果用表 2.1 和表 2.2 的直觉判据加以检验,检验的结果列于表 2.5,其中“○”表示推理结果满足直觉判据,“×”表示不满足直觉判据。

表 2.5 利用直觉判据对各种蕴含操作检验的结果

	R_c	R_p	R_a	R_m	R_b	R_i	R_Δ
判据 1	○	○	×	×	×	○	×
判据 2—1	×	×	×	×	×	○	×
判据 2—2	○	○	×	×	×	×	×
判据 3—1	×	×	×	×	×	○	×
判据 3—2	○	○	×	×	×	×	×
判据 4—1	×	×	○	○	○	○	○
判据 4—2	×	×	×	×	×	×	×
判据 5	×	×	×	×	×	○	×
判据 6	×	×	×	×	×	○	×
判据 7	×	×	×	×	×	○	×
判据 8—1	×	×	○	×	○	○	○
判据 8—2	○	○	×	×	×	×	×

2.4.4 模糊蕴含关系运算方法的比较和选择

在模糊逻辑控制中,主要是根据输入和控制规则来决定控制作用,而且也不需要将一个规则的结论用作另一规则的前提。因此,它是单级数据驱动推理,即肯定式方式推理。因

而在模糊逻辑控制中无需用到反向目标驱动推理(否定式)及链式推理的机制。

从表 2.5 可以看出,虽然 R_L 和 R_p 不太符合常规的逻辑结构,但它们都比较适用于近似推理,尤其适用广义的假言推理。 R_A 、 R_b 和 R_m 虽然比较符合常规的逻辑结构,但是推理的结果不太符合直觉的判据。 R_d 推理结果也不太满足直觉判据。另外 R_c 也给出了较为满意的结果,因此它也很适用于近似推理。

下面通过一个具体例子来说明不同的模糊蕴含关系运算方法,并具体比较各自的推理结果。

例 2.7 若人工调节炉温,有如下的经验规则:“如果炉温低,则应施加高电压”。试问当炉温为“低”、“非常低”、“略低”、“不低”时,应施加怎样的电压?

解: 这是典型的近似推理问题,设 x 和 y 分别表示模糊语言变量“炉温”和“电压”,并设 x 和 y 论域为

$$X = Y = \{1, 2, 3, 4, 5\}$$

设 A 表示炉温低的模糊集合,且有

$$A = \text{“炉温低”} = \frac{1}{1} + \frac{0.8}{2} + \frac{0.6}{3} + \frac{0.4}{4} + \frac{0.2}{5}$$

设 B 表示高电压的模糊集合,且有

$$B = \text{“高电压”} = \frac{0.2}{1} + \frac{0.4}{2} + \frac{0.6}{3} + \frac{0.8}{4} + \frac{1}{5}$$

从而模糊规则可表述为:“如果 x 是 A 则 y 是 B ”。设 A' 分别表示 A 、非常 A 、略 A 和非 A , 则上述问题便变为如果 x 是 A' , 则 B' 应是什么。这便是上面讨论过的典型的广义肯定式推理。下面分别用不同的模糊蕴含关系运算法来进行推理。

1. 模糊蕴含最小运算法 R_c

为了进行近似推理,首先需求模糊蕴含关系 $R_c = A \rightarrow B$ 。根据前面的定义

$$R_c \leftrightarrow \mu_{A \rightarrow B}(x, y) = \mu_A(x) \wedge \mu_B(y) = \min\{\mu_A(x), \mu_B(y)\}$$

该例中 x 和 y 的论域均是离散的,因而模糊蕴含关系 R_c 可用模糊矩阵来表示。为了运算方便,对于离散的模糊集合,也可用相应的模糊向量来表示。在该例中,模糊集合 A 和 B 可表示成如下的模糊向量:

$$A = [1 \ 0.8 \ 0.6 \ 0.4 \ 0.2]$$

$$B = [0.2 \ 0.4 \ 0.6 \ 0.8 \ 1]$$

这样模糊蕴含关系矩阵 R 可以采用如下的方法运算

$$R = A^T \circ B$$

其中 \circ 表示模糊蕴含关系运算符。若采用最小运算法,则有

$$R_c = A^T \circ B = \begin{bmatrix} 1 \\ 0.8 \\ 0.6 \\ 0.4 \\ 0.2 \end{bmatrix} \wedge [0.2 \ 0.4 \ 0.6 \ 0.8 \ 1]$$

$$\begin{aligned}
&= \begin{bmatrix} 1 \wedge 0.2 & 1 \wedge 0.4 & 1 \wedge 0.6 & 1 \wedge 0.8 & 1 \wedge 1 \\ 0.8 \wedge 0.2 & 0.8 \wedge 0.4 & 0.8 \wedge 0.6 & 0.8 \wedge 0.8 & 0.8 \wedge 1 \\ 0.6 \wedge 0.2 & 0.6 \wedge 0.4 & 0.6 \wedge 0.6 & 0.6 \wedge 0.8 & 0.6 \wedge 1 \\ 0.4 \wedge 0.2 & 0.4 \wedge 0.4 & 0.4 \wedge 0.6 & 0.4 \wedge 0.8 & 0.4 \wedge 1 \\ 0.2 \wedge 0.2 & 0.2 \wedge 0.4 & 0.2 \wedge 0.6 & 0.2 \wedge 0.8 & 0.2 \wedge 1 \end{bmatrix} \\
&= \begin{bmatrix} 0.2 & 0.4 & 0.6 & 0.8 & 1 \\ 0.2 & 0.4 & 0.6 & 0.8 & 0.8 \\ 0.2 & 0.4 & 0.6 & 0.6 & 0.6 \\ 0.2 & 0.4 & 0.4 & 0.4 & 0.4 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \end{bmatrix}
\end{aligned}$$

下面是 A' 取不同值时的推理结果。

(1) $A' = A$

$$B' = A' \circ R_c$$

$$\begin{aligned}
&= [1 \quad 0.8 \quad 0.6 \quad 0.4 \quad 0.2] \circ \begin{bmatrix} 0.2 & 0.4 & 0.6 & 0.8 & 1 \\ 0.2 & 0.4 & 0.6 & 0.8 & 0.8 \\ 0.2 & 0.4 & 0.6 & 0.6 & 0.6 \\ 0.2 & 0.4 & 0.4 & 0.4 & 0.4 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \end{bmatrix} \\
&= [0.2 \quad 0.4 \quad 0.6 \quad 0.8 \quad 1]
\end{aligned}$$

其中每个元素是按最大-最小的合成规则计算出来的。例如上式中的第一个元素是这样计算的：

$$\begin{aligned}
&(1 \wedge 0.2) \vee (0.8 \wedge 0.2) \vee (0.6 \wedge 0.2) \vee (0.4 \wedge 0.2) \vee (0.2 \wedge 0.2) \\
&= 0.2 \vee 0.2 \vee 0.2 \vee 0.2 \vee 0.2 = 0.2
\end{aligned}$$

分析近似推理的结果知

$$B' = \frac{0.2}{1} + \frac{0.4}{2} + \frac{0.6}{3} + \frac{0.8}{4} + \frac{1}{5} = \text{“高电压”}$$

显然推理结果满足直觉判据。

(2) $A' = A^2$

$$B' = A' \circ R_c = A^2 \circ R_c$$

$$\begin{aligned}
&= [1 \quad 0.64 \quad 0.36 \quad 0.16 \quad 0.04] \circ \begin{bmatrix} 0.2 & 0.4 & 0.6 & 0.8 & 1 \\ 0.2 & 0.4 & 0.6 & 0.8 & 0.8 \\ 0.2 & 0.4 & 0.6 & 0.6 & 0.6 \\ 0.2 & 0.4 & 0.4 & 0.4 & 0.4 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \end{bmatrix} \\
&= [0.2 \quad 0.4 \quad 0.6 \quad 0.8 \quad 1]
\end{aligned}$$

这时推理结果 B' 仍为“高电压”，它满足直觉判据 2-2(见表 2.5)，尚属满足要求。

(3) $A' = A^{0.5}$

$$B' = A' \circ R_c = A^{0.5} \circ R_c$$

$$\begin{aligned}
&= [1 \quad 0.89 \quad 0.77 \quad 0.63 \quad 0.45] \circ \begin{bmatrix} 0.2 & 0.4 & 0.6 & 0.8 & 1 \\ 0.2 & 0.4 & 0.6 & 0.8 & 0.8 \\ 0.2 & 0.4 & 0.6 & 0.6 & 0.6 \\ 0.2 & 0.4 & 0.4 & 0.4 & 0.4 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \end{bmatrix} \\
&= [0.2 \quad 0.4 \quad 0.6 \quad 0.8 \quad 1]
\end{aligned}$$

这时推理结果 B' 仍为“高电压”，它满足直觉判据 3-2 (见表 2.5)，尚属满足要求。

(4) $A' = \text{非 } A = \bar{A}$

$$B' = A' \circ R_c = \bar{A} \circ R_c$$

$$\begin{aligned}
&= [0 \quad 0.2 \quad 0.4 \quad 0.6 \quad 0.8] \circ \begin{bmatrix} 0.2 & 0.4 & 0.6 & 0.8 & 1 \\ 0.2 & 0.4 & 0.6 & 0.8 & 0.8 \\ 0.2 & 0.4 & 0.6 & 0.6 & 0.6 \\ 0.2 & 0.4 & 0.4 & 0.4 & 0.4 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \end{bmatrix} \\
&= [0.2 \quad 0.4 \quad 0.4 \quad 0.4 \quad 0.4]
\end{aligned}$$

根据直觉推理，此时应有 $B' = \text{非 } B = \bar{B} = [0.8 \quad 0.5 \quad 0.4 \quad 0.2 \quad 0]$ (判据 4-2)，可见此时的推理结果与直觉判据有较大出入。

2. 模糊蕴含积运算法 R_p

根据 $R_p \leftrightarrow \mu_{A \rightarrow B}(x, y) = \mu_A(x) \mu_B(y)$ ，可以求得

$$R_p = A^1 \bigcirc \rightarrow B$$

$$\begin{aligned}
&= \begin{bmatrix} 1 \\ 0.8 \\ 0.6 \\ 0.4 \\ 0.2 \end{bmatrix} [0.2 \quad 0.4 \quad 0.6 \quad 0.8 \quad 1] = \begin{bmatrix} 0.2 & 0.4 & 0.6 & 0.8 & 1 \\ 0.16 & 0.32 & 0.48 & 0.64 & 0.8 \\ 0.12 & 0.24 & 0.36 & 0.48 & 0.6 \\ 0.08 & 0.16 & 0.24 & 0.32 & 0.4 \\ 0.04 & 0.08 & 0.12 & 0.16 & 0.2 \end{bmatrix}
\end{aligned}$$

(1) $A' = A$

$$B' = A' \circ R_p = A \circ R_p = [0.2 \quad 0.4 \quad 0.6 \quad 0.8 \quad 1]$$

(2) $A' = A^2$

$$B' = A' \circ R_p = A^2 \circ R_p = [0.2 \quad 0.4 \quad 0.6 \quad 0.8 \quad 1]$$

(3) $A' = A^{0.5}$

$$B' = A' \circ R_p = A^{0.5} \circ R_p = [0.2 \quad 0.4 \quad 0.6 \quad 0.8 \quad 1]$$

(4) $A' = \text{非 } A = \bar{A}$

$$B' = A' \circ R_p = \bar{A} \circ R_p = [0.16 \quad 0.24 \quad 0.36 \quad 0.4 \quad 0.4]$$

由上面结果看出，前 3 种情况 ($A' = A, A^2, A^{0.5}$) 均满足直觉判据，只有第 4 种情况 ($A' = \bar{A}$) 不满足。其结论显然与表 2.5 给出的相同。

3. 模糊蕴含算术运算法 R_a

$$R_a \leftrightarrow \mu_{A \rightarrow B}(x, y) = 1 \wedge [1 - \mu_A(x) + \mu_B(y)]$$

据此求得

$$R_p = A^T \rightarrow B$$

$$= \begin{bmatrix} 1 \\ 0.8 \\ 0.6 \\ 0.4 \\ 0.2 \end{bmatrix} \rightarrow [0.2 \quad 0.4 \quad 0.6 \quad 0.8 \quad 1] = \begin{bmatrix} 0.2 & 0.4 & 0.6 & 0.8 & 1 \\ 0.4 & 0.6 & 0.8 & 1 & 1 \\ 0.6 & 0.8 & 1 & 1 & 1 \\ 0.8 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

进而求得

$$B' = A' \circ R_p = \begin{cases} [0.6 & 0.6 & 0.8 & 0.8 & 1] & A' = A \\ [0.4 & 0.6 & 0.64 & 0.8 & 1] & A' = A^2 \\ [0.63 & 0.77 & 0.8 & 0.89 & 1] & A' = A^{0.5} \\ [0.8 & 0.8 & 0.8 & 0.8 & 0.8] & A' = \bar{A} \end{cases}$$

结果表明,采用这种方法虽然形式上具有与普通集合类似的推理结构,但是应用到模糊推理基本上均不满足直觉判据。所以在模糊逻辑控制中很少被采用。

4. 模糊蕴含最大最小运算法 R_m

$$R_m \leftrightarrow \mu_{A \rightarrow B}(x, y) = [\mu_A(x) \wedge \mu_B(y)] \vee [1 - \mu_A(x)]$$

$$R_m = A^T \rightarrow B$$

$$= \begin{bmatrix} 1 \\ 0.8 \\ 0.6 \\ 0.4 \\ 0.2 \end{bmatrix} \rightarrow [0.2 \quad 0.4 \quad 0.6 \quad 0.8 \quad 1] = \begin{bmatrix} 0.2 & 0.4 & 0.6 & 0.8 & 1 \\ 0.2 & 0.4 & 0.6 & 0.8 & 0.8 \\ 0.4 & 0.4 & 0.6 & 0.6 & 0.6 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \end{bmatrix}$$

$$B' = A' \circ R_m = \begin{cases} [0.4 & 0.4 & 0.6 & 0.8 & 1] & A' = A \\ [0.36 & 0.4 & 0.6 & 0.8 & 1] & A' = A^2 \\ [0.6 & 0.6 & 0.6 & 0.8 & 1] & A' = A^{0.5} \\ [0.8 & 0.8 & 0.8 & 0.8 & 0.8] & A' = \bar{A} \end{cases}$$

可见这种方法也基本上不满足直觉判据。

5. 模糊蕴含布尔运算法 R_b

$$R_b \leftrightarrow \mu_{A \rightarrow B}(x, y) = [1 - \mu_A(x)] \vee \mu_B(y)$$

$$R_b = A^T \rightarrow B$$

$$= \begin{bmatrix} 1 \\ 0.8 \\ 0.6 \\ 0.4 \\ 0.2 \end{bmatrix} \rightarrow [0.2 \quad 0.4 \quad 0.6 \quad 0.8 \quad 1] = \begin{bmatrix} 0.2 & 0.4 & 0.6 & 0.8 & 1 \\ 0.2 & 0.4 & 0.6 & 0.8 & 1 \\ 0.4 & 0.4 & 0.6 & 0.8 & 1 \\ 0.6 & 0.6 & 0.6 & 0.8 & 1 \\ 0.8 & 0.8 & 0.8 & 0.8 & 1 \end{bmatrix}$$

$$B' = A' \circ R_s = \begin{cases} [0.4 & 0.4 & 0.6 & 0.8 & 1] & A' = A \\ [0.36 & 0.4 & 0.6 & 0.8 & 1] & A' = A^2 \\ [0.6 & 0.6 & 0.6 & 0.8 & 1] & A' = A^{0.5} \\ [0.8 & 0.8 & 0.8 & 0.8 & 0.8] & A' = \bar{A} \end{cases}$$

这里采用 R_s 得到了与 R_m 完全相同的结果,但基本上都不满足直觉判据。

6. 模糊蕴含的标准运算法(1)

$$R_{s \leftrightarrow \mu_{A \rightarrow B}}(x, y) = \begin{cases} 1 & \mu_A(x) \leq \mu_B(y) \\ 0 & \mu_A(x) > \mu_B(y) \end{cases}$$

$$R_s = A^T \bigcirc B$$

$$= \begin{bmatrix} 1 \\ 0.8 \\ 0.6 \\ 0.4 \\ 0.2 \end{bmatrix} \bigcirc [0.2 \quad 0.4 \quad 0.6 \quad 0.8 \quad 1] = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$B' = A' \circ R_s = \begin{cases} [0.2 & 0.4 & 0.6 & 0.8 & 1] & A' = A \\ [0.04 & 0.16 & 0.36 & 0.64 & 1] & A' = A^2 \\ [0.45 & 0.63 & 0.77 & 0.89 & 1] & A' = A^{0.5} \\ [0.8 & 0.8 & 0.8 & 0.8 & 0.8] & A' = \bar{A} \end{cases}$$

可见,这种方法与直觉判据具有很好的符合程度。

7. 模糊蕴含的标准运算法(2)

$$R_{\Delta \leftrightarrow \mu_{A \rightarrow B}}(x, y) = \begin{cases} 1 & \mu_A(x) \leq \mu_B(y) \\ \frac{\mu_B(y)}{\mu_A(x)} & \mu_A(x) > \mu_B(y) \end{cases}$$

$$R_{\Delta} = A^T \bigcirc B$$

$$= \begin{bmatrix} 1 \\ 0.8 \\ 0.6 \\ 0.4 \\ 0.2 \end{bmatrix} \bigcirc [0.2 \quad 0.4 \quad 0.6 \quad 0.8 \quad 1] = \begin{bmatrix} 0.2 & 0.4 & 0.6 & 0.8 & 1 \\ 0.25 & 0.5 & 0.75 & 1 & 1 \\ 0.33 & 0.67 & 1 & 1 & 1 \\ 0.5 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$B' = A' \circ R_{\Delta} = \begin{cases} [0.4 & 0.6 & 0.75 & 0.8 & 1] & A' = A \\ [0.33 & 0.5 & 0.64 & 0.8 & 1] & A' = A^2 \\ [0.5 & 0.63 & 0.77 & 0.89 & 1] & A' = A^{0.5} \\ [0.8 & 0.8 & 0.8 & 0.8 & 0.8] & A' = \bar{A} \end{cases}$$

可见,这种方法也基本上不满足直观判据的要求。

2.4.5 合成运算方法的选择

对于 $B' = A' \circ R$ 中所用到的合成运算,通常可以采用如下 4 种不同的方法。

1. 最大-最小合成法(Zadeh, 1973)

$$\mu_{B'}(y) = \bigvee_{x \in X} [\mu_{A'}(x) \wedge \mu_R(x, y)]$$

这是上面讨论中所采用的方法。

2. 最大-代数积合成法(Kaufmann, 1975)

$$\mu_{B'}(y) = \bigvee_{x \in X} \mu_{A'}(x) \mu_R(x, y)$$

该方法有时也简称为最大-积合成法。

3. 最大-有界积合成法(Mizumoto, 1981)

$$\begin{aligned} \mu_{B'}(y) &= \bigvee_{x \in X} \mu_{A'}(x) \odot \mu_R(x, y) \\ &= \bigvee_{x \in X} \max\{0, \mu_{A'}(x) + \mu_R(x, y) - 1\} \end{aligned}$$

4. 最大-强制积合成法(Mizumoto, 1981)

$$\mu_{B'}(y) = \bigvee_{x \in X} \mu_{A'}(x) \odot \mu_R(x, y)$$

其中

$$\mu_{A'}(x) \odot \mu_R(x, y) = \begin{cases} \mu_{A'}(x) & \mu_R(x, y) = 1 \\ \mu_R(x, y) & \mu_{A'}(x) = 1 \\ 0 & \mu_{A'}(x), \mu_R(x, y) < 1 \end{cases}$$

在模糊逻辑控制的应用中,最常用的是第1和第2两种方法,即最大-最小和最大-积合成法。原因是这两种方法计算比较简单。尤其是实时性要求很高的控制问题,这是一个首要考虑的因素。但据 Mizumoto 对不同的模糊运算方法所进行的研究考察表明,采用上述第2,3,4的合成方法要比应用第1种方法(Zadeh 的最大-最小合成法)效果更好。当然这个结论还需要进一步的研究和证实。

2.4.6 句子连接关系的逻辑运算

1. 句子连接词“and”

在模糊逻辑控制中,常常使用如下的广义肯定式推理方式:

前提1: x 是 A' and y 是 B'

前提2: 如果 x 是 A and y 是 B 则 z 是 C

结论: z 是 C'

与前面不同的是,这里模糊条件的假设部分是将模糊命题用“and”连接起来的。一般情况下可以有多个“and”将多个模糊命题连接在一起。

在前提2中的前提条件“ x 是 A and y 是 B ”可以看成是直积空间 $X \times Y$ 上的模糊集合,并记为 $A \times B$,其隶属度函数为

$$\mu_{A \times B}(x, y) = \min\{\mu_A(x), \mu_B(y)\}$$

或者

$$\mu_{A \times B}(x, y) = \mu_A(x) \mu_B(y)$$

这时的模糊蕴含关系可记为 $A \times B \rightarrow C$,其具体的运算方法也可如前面简单模糊蕴含关系那样有好多种,如

$$C = (A \times B) \circ R_C$$

$$\begin{aligned}
 R_c &= A \times B \rightarrow C = A \times B \times C = \int_{x, y, z} \mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z) / (x, y, z) \\
 R_o &= A \times B \rightarrow C = (\overline{A \times B \times C}) \oplus (X \times Y \times C) \\
 &= \int_{x, y, z} 1 \wedge [1 - (\mu_A(x) \wedge \mu_B(y)) + \mu_C(z)] / (x, y, z)
 \end{aligned}$$

其余类推。

对于结论 C' 可用如下的近似推理求出：

$$C' = (A' \times B') \circ R$$

其中 R 为模糊蕴含关系，它可用上面所定义的任何一种。“ \circ ”为合成运算符。

2. 句子连接词“also”

在模糊逻辑控制中，常常给出一系列的模糊控制规则，每一条规则都具有如下的形式：“如果 x 是 A_i and y 是 B_i 则 z 是 C_i ” ($i=1, 2, \dots, n$)，这些规则之间无先后次序之分。连接这些子句的连接词用“also”表示。这就要求对于“also”的运算具有能够任意交换和任意结合的性质。求并和求交运算均能满足这样的要求。Mizumoto 采用不同的模糊蕴含运算方法与不同的“also”运算相组合，进行了大量的研究和比较，最后得出结论认为，模糊蕴含运算采用 R_c 或 R_o 以及“also”采用求并运算给出了最好的控制结果。从实用的观点，这样的组合对于实现也是最简单的。模糊蕴含运算标准法 R_c 虽然对于近似推理也给出了很好的结果，但是它并不适于用在控制中。

2.5 基于控制规则库的模糊推理

2.5.1 模糊推理的基本方法

模糊控制中的规则通常来源于专家的知识，对于多输入多输出(MIMO)系统，其规则具有如下的形式：

$$R = \{R_{\text{MIMO}}^1, R_{\text{MIMO}}^2, \dots, R_{\text{MIMO}}^n\}$$

其中

$$R_{\text{MIMO}}^i: \text{如果}(x \text{ 是 } A_i \text{ and } \dots \text{and } y \text{ 是 } B_i) \text{ 则}(z_1 \text{ 是 } C_i, \dots, z_q \text{ 是 } D_i)$$

R_{MIMO}^i 的前提条件构成了在直积空间 $X \times \dots \times Y$ 上的模糊集合 $A_i \times \dots \times B_i$ ，结论是 q 个控制作用的并，它们之间是互相独立的。因此，第 i 条规则 R_{MIMO}^i 可以表示为如下的模糊蕴含关系：

$$R_{\text{MIMO}}^i: (A_i \times \dots \times B_i) \rightarrow (C_i + \dots + D_i)$$

于是规则 R 可以表示为

$$\begin{aligned}
 R &= \left\{ \bigcup_{i=1}^n R_{\text{MIMO}}^i \right\} \\
 &= \left\{ \bigcup_{i=1}^n [(A_i \times \dots \times B_i) \rightarrow (C_i + \dots + D_i)] \right\} \\
 &= \left\{ \bigcup_{i=1}^n [(A_i \times \dots \times B_i) \rightarrow C_i], \dots, \bigcup_{i=1}^n [(A_i \times \dots \times B_i) \rightarrow D_i] \right\} \\
 &= \{RB_{\text{MISO}}^1, \dots, RB_{\text{MISO}}^q\}
 \end{aligned}$$

可见规则库 R 可看成由 q 个子规则库所组成, 每一个子规则库由 n 个多输入单输出 (MISO) 的规则所组成。由于每个子规则是互相独立的, 因此下面只需考虑其中一个 MISO 子规则库的近似推理问题。

不失一般性, 考虑如下的两个输入一个输出的模糊系统:

输入: x 是 A' and y 是 B'
 R_1 : 如果 x 是 A_1 and y 是 B_1 则 z 是 C_1
 also R_2 : 如果 x 是 A_2 and y 是 B_2 则 z 是 C_2
 \vdots
 also R_n : 如果 x 是 A_n and y 是 B_n 则 z 是 C_n

输出: z 是 C'

其中 x, y 和 z 是代表系统状态和控制量的语言变量, A_i, B_i 和 C_i 分别是 x, y 和 z 的语言值。 x, y 和 z 的论域分别为 X, Y 和 Z 。

模糊控制规则“如果 x 是 A_i and y 是 B_i 则 z 是 C_i ”的模糊蕴含关系 R_i 定义为:

$$R_i = (A_i \text{ and } B_i) \rightarrow C_i$$

即

$$\begin{aligned}\mu_{R_i} &= \mu_{(A_i \text{ and } B_i \rightarrow C_i)}(x, y, z) \\ &= [\mu_{A_i}(x) \text{ and } \mu_{B_i}(y)] \rightarrow \mu_{C_i}(z)\end{aligned}$$

其中“ A_i and B_i ”是定义在 $X \times Y$ 上的模糊集合 $A_i \times B_i$, $R_i = (A_i \text{ and } B_i) \rightarrow C_i$ 是定义在 $X \times Y \times Z$ 上的模糊蕴含关系。考虑 n 条模糊控制规则的总的模糊蕴含关系为(取连接词“also”为求并运算)

$$R = \bigcup_{i=1}^n R_i$$

最后求得推理的结论为

$$C' = (A' \text{ and } B') \circ R$$

其中

$$\mu_{(A' \text{ and } B')}(x, y) = \mu_{A'}(x) \wedge \mu_{B'}(y)$$

或者

$$\mu_{(A' \text{ and } B')}(x, y) = \mu_{A'}(x) \mu_{B'}(y)$$

“ \circ ”是合成运算符, 通常采用最大-最小合成法。

例 2.8 已知一个双输入单输出的模糊系统, 其输入量为 x 和 y , 输出量为 z , 其输入输出关系可用如下两条模糊规则描述:

R_1 : 如果 x 是 A_1 and y 是 B_1 则 z 是 C_1

R_2 : 如果 x 是 A_2 and y 是 B_2 则 z 是 C_2

现已知输入为 x 是 A' and y 是 B' , 试求输出量 z 。这里 x, y, z 均为模糊语言变量, 且已知

$$\begin{aligned}A_1 &= \frac{1.0}{a_1} + \frac{0.5}{a_2} + \frac{0}{a_3} & B_1 &= \frac{1.0}{b_1} + \frac{0.6}{b_2} + \frac{0.2}{b_3} & C_1 &= \frac{1.0}{c_1} + \frac{0.4}{c_2} + \frac{0}{c_3} \\ A_2 &= \frac{0}{a_1} + \frac{0.5}{a_2} + \frac{1.0}{a_3} & B_2 &= \frac{0.2}{b_1} + \frac{0.6}{b_2} + \frac{1.0}{b_3} & C_2 &= \frac{0}{c_1} + \frac{0.4}{c_2} + \frac{1.0}{c_3}\end{aligned}$$

$$A' = \frac{0.5}{a_1} + \frac{1.0}{a_2} + \frac{0.5}{a_3} \quad B' = \frac{0.6}{b_1} + \frac{1.0}{b_2} + \frac{0.6}{b_3}$$

解: 由于这里所有模糊集合的元素均为离散量, 所以模糊集合可用模糊向量来描述, 模糊关系可用模糊矩阵来描述。

(1) 求每条规则的蕴含关系 $R_i = (A_i \text{ and } B_i) \rightarrow C_i \quad (i=1, 2)$

若此处 A_i and B_i 采用求交运算, 蕴含关系运算采用最小运算 R_c , 则

$$A_1 \text{ and } B_1 = A_1 \times B_1 = A_1^T \wedge B_1 = \begin{bmatrix} 1.0 \\ 0.5 \\ 0 \end{bmatrix} \wedge [1.0 \quad 0.6 \quad 0.2] = \begin{bmatrix} 1.0 & 0.6 & 0.2 \\ 0.5 & 0.5 & 0.2 \\ 0 & 0 & 0 \end{bmatrix}$$

为便于下面进一步的计算, 可将 $A_1 \times B_1$ 的模糊矩阵表示成如下的向量:

$$\bar{R}_{A_1 \times B_1} = [1.0 \quad 0.6 \quad 0.2 \quad 0.5 \quad 0.5 \quad 0.2 \quad 0 \quad 0 \quad 0]$$

则

$$\begin{aligned} R_1 &= (A_1 \text{ and } B_1) \rightarrow C_1 = \bar{R}_{A_1 \times B_1} \wedge C_1 \\ &= \begin{bmatrix} 1.0 \\ 0.6 \\ 0.2 \\ 0.5 \\ 0.5 \\ 0.2 \\ 0 \\ 0 \\ 0 \end{bmatrix} \wedge [1.0 \quad 0.4 \quad 0] = \begin{bmatrix} 1.0 & 0.4 & 0 \\ 0.6 & 0.4 & 0 \\ 0.2 & 0.2 & 0 \\ 0.5 & 0.4 & 0 \\ 0.5 & 0.4 & 0 \\ 0.2 & 0.2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{aligned}$$

仿照同样的步骤可以求得 R_2 为

$$R_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0.2 & 0.2 \\ 0 & 0.4 & 0.5 \\ 0 & 0.4 & 0.5 \\ 0 & 0.2 & 0.2 \\ 0 & 0.4 & 0.6 \\ 0 & 0.4 & 1.0 \end{bmatrix}$$

(2) 求总的模糊蕴含关系 R

$$R = R_1 \cup R_2 = \begin{bmatrix} 1.0 & 0.4 & 0 \\ 0.6 & 0.4 & 0 \\ 0.2 & 0.2 & 0 \\ 0.5 & 0.4 & 0.2 \\ 0.5 & 0.4 & 0.5 \\ 0.2 & 0.4 & 0.5 \\ 0 & 0.2 & 0.2 \\ 0 & 0.4 & 0.6 \\ 0 & 0.4 & 1.0 \end{bmatrix}$$

(3) 计算输入量的模糊集合 A' and B'

$$A' \text{ and } B' = A' \times B' = A'^T \wedge B' = \begin{bmatrix} 0.5 \\ 1.0 \\ 0.5 \end{bmatrix} \wedge [0.6 \quad 1.0 \quad 0.6] = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.6 & 1.0 & 0.6 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}$$

$$\bar{R}_{A' \times B'} = [0.5 \quad 0.5 \quad 0.5 \quad 0.6 \quad 1.0 \quad 0.6 \quad 0.5 \quad 0.5 \quad 0.5]$$

(4) 计算输出量的模糊集合

$$C' = (A' \text{ and } B') \circ R = \bar{R}_{A' \times B'} \circ R$$

$$= [0.5 \quad 0.5 \quad 0.5 \quad 0.6 \quad 1.0 \quad 0.6 \quad 0.5 \quad 0.5 \quad 0.5] \circ \begin{bmatrix} 1.0 & 0.4 & 0 \\ 0.6 & 0.4 & 0 \\ 0.2 & 0.2 & 0 \\ 0.5 & 0.4 & 0.2 \\ 0.5 & 0.4 & 0.5 \\ 0.2 & 0.4 & 0.5 \\ 0 & 0.2 & 0.2 \\ 0 & 0.4 & 0.6 \\ 0 & 0.4 & 1.0 \end{bmatrix}$$

$$= [0.5 \quad 0.4 \quad 0.5]$$

最后求得输出量 z 的模糊集合为

$$C' = \frac{0.5}{c_1} + \frac{0.4}{c_2} + \frac{0.5}{c_3}$$

2.5.2 模糊推理的性质

从上面的例题计算可以看出,当输入的维数较高,即有很多个模糊子句用“and”相连时,模糊推理的计算便较复杂。下面介绍模糊推理计算的一些有用的性质。

性质 1: 若合成运算“ \circ ”采用最大-最小法或最大-积法,连接词“also”采用求并法,则“ \circ ”和“also”的运算次序可以交换,即

$$(A' \text{ and } B') \circ \bigcup_{i=1}^n R_i = \bigcup_{i=1}^n (A' \text{ and } B') \circ R_i$$

证明: 先考虑“ \circ ”表示最大-最小合成法

$$C' = (A' \text{ and } B') \circ \bigcup_{i=1}^n R_i = (A' \text{ and } B') \circ \bigcup_{i=1}^n (A_i \text{ and } B_i \rightarrow C_i)$$

即

$$\begin{aligned} \mu_{C'}(z) &= [\mu_{A'}(x) \text{ and } \mu_{B'}(y)] \circ \max[\mu_{R_1}(x, y, z), \dots, \mu_{R_n}(x, y, z)] \\ &= \max_{x, y} \min\{[\mu_{A'}(x) \text{ and } \mu_{B'}(y)], \max[\mu_{R_1}(x, y, z), \dots, \mu_{R_n}(x, y, z)]\} \\ &= \max_{x, y} \max\{\min[(\mu_{A'}(x) \text{ and } \mu_{B'}(y)), \mu_{R_1}(x, y, z)], \dots, \\ &\quad \min[(\mu_{A'}(x) \text{ and } \mu_{B'}(y)), \mu_{R_n}(x, y, z)]\} \\ &= \max\{[(\mu_{A'}(x) \text{ and } \mu_{B'}(y)) \circ \mu_{R_1}(x, y, z)], \dots, [(\mu_{A'}(x) \text{ and } \mu_{B'}(y)) \circ \mu_{R_n}(x, y, z)]\} \end{aligned}$$

也就是说

$$\begin{aligned} C' &= [(A' \text{ and } B') \circ R_1] \cup \dots \cup [(A' \text{ and } B') \circ R_n] \\ &= \bigcup_{i=1}^n [(A' \text{ and } B') \circ R_i] \\ &= \bigcup_{i=1}^n [(A' \text{ and } B') \circ (A_i \text{ and } B_i \rightarrow C_i)] \\ &= \bigcup_{i=1}^n C'_i \end{aligned}$$

其中

$$C'_i = (A' \text{ and } B') \circ (A_i \text{ and } B_i \rightarrow C_i)$$

对于“ \circ ”表示最大-积合成法的情况,同样可以证得上述结论也成立。

例 2.9 利用性质 1 重新求解例 2.8

解:

$$\begin{aligned} C' &= C'_1 \cup C'_2 \\ &= [(A' \text{ and } B') \circ R_1] \cup [(A' \text{ and } B') \circ R_2] \\ &= [\bar{R}_{A' \times B'} \circ R_1] \cup [\bar{R}_{A' \times B'} \circ R_2] \end{aligned}$$

在例 2.8 中,合成运算符“ \circ ”采用的是最大-最小法,所以下面也采用同样的方法。

$$C'_1 = \bar{R}_{A' \times B'} \circ R_1$$

$$\begin{aligned} &= [0.5 \quad 0.5 \quad 0.5 \quad 0.6 \quad 1.0 \quad 0.6 \quad 0.5 \quad 0.5 \quad 0.5] \circ \begin{bmatrix} 1.0 & 0.4 & 0 \\ 0.6 & 0.4 & 0 \\ 0.2 & 0.2 & 0 \\ 0.5 & 0.4 & 0 \\ 0.5 & 0.4 & 0 \\ 0.2 & 0.2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ &= [0.5 \quad 0.4 \quad 0] \end{aligned}$$

$$C'_2 = \bar{R}_{A' \times B'} \circ R_2$$

$$= [0.5 \quad 0.5 \quad 0.5 \quad 0.6 \quad 1.0 \quad 0.6 \quad 0.5 \quad 0.5 \quad 0.5] \circ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0.2 & 0.2 \\ 0 & 0.4 & 0.5 \\ 0 & 0.4 & 0.5 \\ 0 & 0.2 & 0.2 \\ 0 & 0.4 & 0.6 \\ 0 & 0.4 & 1.0 \end{bmatrix}$$

$$= [0 \quad 0.4 \quad 0.5]$$

$$C' = C_1' \cup C_2' = [0.5 \quad 0.4 \quad 0.5]$$

可见所求结果与例 2.8 相同。

性质 2: 若模糊蕴含关系采用 R_c 和 R_p 时, 则有

$$(A' \text{ and } B') \circ (A_i \text{ and } B_i \rightarrow C_i) = [A' \circ (A_i \rightarrow C_i)] \cap [B' \circ (B_i \rightarrow C_i)]$$

证明: 这里假设模糊蕴含关系采用 R_c , 合成运算采用最大-最小法, and 运算采用求交法, 则

$$\begin{aligned} C_i' &= (A' \text{ and } B') \circ (A_i \text{ and } B_i \rightarrow C_i) \\ \mu_{C_i'} &= (\mu_{A'} \text{ and } \mu_{B'}) \circ (\mu_{A_i \times B_i} \rightarrow \mu_{C_i}) \\ &= (\mu_{A'} \text{ and } \mu_{B'}) \circ [\min(\mu_{A_i}, \mu_{B_i}) \rightarrow \mu_{C_i}] \\ &= (\mu_{A'} \text{ and } \mu_{B'}) \circ \min[(\mu_{A_i} \rightarrow \mu_{C_i}), (\mu_{B_i} \rightarrow \mu_{C_i})] \\ &= \max_{x,y} \min\{\min(\mu_{A'}, \mu_{B'}), \min[(\mu_{A_i} \rightarrow \mu_{C_i}), (\mu_{B_i} \rightarrow \mu_{C_i})]\} \\ &= \max_{x,y} \min\{\min[\mu_{A'}, (\mu_{A_i} \rightarrow \mu_{C_i})], \min[\mu_{B'}, (\mu_{B_i} \rightarrow \mu_{C_i})]\} \\ &= \min\{[\mu_{A'} \circ (\mu_{A_i} \rightarrow \mu_{C_i})], [\mu_{B'} \circ (\mu_{B_i} \rightarrow \mu_{C_i})]\} \end{aligned}$$

也即

$$C_i' = [A' \circ (A_i \rightarrow C_i)] \cap [B' \circ (B_i \rightarrow C_i)]$$

采用类似的步骤可以证得当模糊蕴含关系采用 R_p 时, 上面的关系也成立。

例 2.10 利用性质 1 和性质 2, 重新求解例 2.8

解

$$\begin{aligned} A_1 \rightarrow C_1 &= \begin{bmatrix} 1.0 \\ 0.5 \\ 0 \end{bmatrix} \wedge [1.0 \quad 0.4 \quad 0] = \begin{bmatrix} 1.0 & 0.4 & 0 \\ 0.5 & 0.4 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ A' \circ (A_1 \rightarrow C_1) &= [0.5 \quad 1.0 \quad 0.5] \circ \begin{bmatrix} 1.0 & 0.4 & 0 \\ 0.5 & 0.4 & 0 \\ 0 & 0 & 0 \end{bmatrix} = [0.5 \quad 0.4 \quad 0] \\ B_1 \rightarrow C_1 &= \begin{bmatrix} 1.0 \\ 0.6 \\ 0.2 \end{bmatrix} \wedge [1.0 \quad 0.4 \quad 0] = \begin{bmatrix} 1.0 & 0.4 & 0 \\ 0.6 & 0.4 & 0 \\ 0.2 & 0.2 & 0 \end{bmatrix} \end{aligned}$$

$$B' \circ (B_1 \rightarrow C_1) = [0.6 \quad 1.0 \quad 0.6] \circ \begin{bmatrix} 1.0 & 0.4 & 0 \\ 0.6 & 0.4 & 0 \\ 0.2 & 0.2 & 0 \end{bmatrix} = [0.6 \quad 0.4 \quad 0]$$

$$\begin{aligned} C_1' &= [A' \circ (A_1 \rightarrow C_1)] \cap [B' \circ (B_1 \rightarrow C_1)] \\ &= [0.5 \quad 0.4 \quad 0] \cap [0.6 \quad 0.4 \quad 0] = [0.5 \quad 0.4 \quad 0] \end{aligned}$$

同理可以求得

$$C_2' = [A' \circ (A_2 \rightarrow C_2)] \cap [B' \circ (B_2 \rightarrow C_2)] = [0 \quad 0.4 \quad 0.5]$$

根据性质 1 有

$$C' = C_1' \cup C_2' = [0.5 \quad 0.4 \quad 0] \cup [0 \quad 0.4 \quad 0.5] = [0.5 \quad 0.4 \quad 0.5]$$

可见所得结果与例 2.8 相同。

通过上例看出,利用性质 2 可使计算简单。当用 and 连接的模糊子句很多时,用例 2.8 的方法计算总的模糊蕴含关系 R 很复杂,模糊矩阵的维数将很高。而利用性质 2,每个子模糊蕴含关系都比较简单,模糊矩阵的维数也较低,并不随 and 连接的模糊子句的个数增加而增加。

性质 3: 对于 $C' = (A' \text{ and } B') \circ (A, \text{ and } B_i \rightarrow C_i)$ 的推理结果可以用如下简洁的形式来表示。

$$\mu_{C'}(z) = \alpha_i \wedge \mu_{C_i}(z) \quad \text{当模糊蕴含运算采用 } R_c$$

$$\mu_{C'}(z) = \alpha_i \mu_{C_i}(z) \quad \text{当模糊蕴含运算采用 } R_p$$

其中
$$\alpha_i = [\max_x (\mu_{A'}(x) \wedge \mu_{A_i}(x))] \wedge [\max_y (\mu_{B'}(y) \wedge \mu_{B_i}(y))]$$

证明: 设模糊运算采用 R_c , 合成运算采用最大-最小法, 则根据性质 2 有

$$\begin{aligned} C' &= (A' \text{ and } B') \circ (A, \text{ and } B_i \rightarrow C_i) = [A' \circ (A_i \rightarrow C_i)] \cap [B' \circ (B_i \rightarrow C_i)] \\ \mu_{C'}(z) &= \min \{ \max_x \min [\mu_{A'}(x), (\mu_{A_i}(x) \rightarrow \mu_{C_i}(z))], \max_y \min [\mu_{B'}(y), (\mu_{B_i}(y) \rightarrow \mu_{C_i}(z))] \} \\ &= \min \{ \max_x \min [\mu_{A'}(x), \mu_{A_i}(x) \wedge \mu_{C_i}(z)], \max_y \min [\mu_{B'}(y), \mu_{B_i}(y) \wedge \mu_{C_i}(z)] \} \\ &= \min \{ \max_x [\mu_{A'}(x) \wedge \mu_{A_i}(x) \wedge \mu_{C_i}(z)], \max_y [\mu_{B'}(y) \wedge \mu_{B_i}(y) \wedge \mu_{C_i}(z)] \} \\ &= [\max_x (\mu_{A'}(x) \wedge \mu_{A_i}(x) \wedge \mu_{C_i}(z))] \wedge [\max_y (\mu_{B'}(y) \wedge \mu_{B_i}(y) \wedge \mu_{C_i}(z))] \\ &= [\max_x (\mu_{A'}(x) \wedge \mu_{A_i}(x))] \wedge [\max_y (\mu_{B'}(y) \wedge \mu_{B_i}(y))] \wedge \mu_{C_i}(z) \\ &= \alpha_i \wedge \mu_{C_i}(z) \end{aligned}$$

设模糊蕴含运算采用 R_p , 则有

$$\begin{aligned} \mu_{C'}(z) &= \min \{ \max_x \min [\mu_{A'}(x), \mu_{A_i}(x) \mu_{C_i}(z)], \max_y \min [\mu_{B'}(y), \mu_{B_i}(y) \mu_{C_i}(z)] \} \\ &= \min \{ \max_x [\mu_{A'}(x) \wedge \mu_{A_i}(x) \mu_{C_i}(z)], \max_y [\mu_{B'}(y) \wedge \mu_{B_i}(y) \mu_{C_i}(z)] \} \\ &= [\max_x (\mu_{A'}(x) \wedge \mu_{A_i}(x) \mu_{C_i}(z))] \wedge [\max_y (\mu_{B'}(y) \wedge \mu_{B_i}(y) \mu_{C_i}(z))] \\ &\approx \{ [\max_x (\mu_{A'}(x) \wedge \mu_{A_i}(x))] \wedge [\max_y (\mu_{B'}(y) \wedge \mu_{B_i}(y))] \} \mu_{C_i}(z) \\ &= \alpha_i \mu_{C_i}(z) \end{aligned}$$

推论:如果输入量的模糊集合是模糊单点(singleton),即 $A' = \frac{1}{x_0}, B' = \frac{1}{y_0}$, 则有

$$R_C: \mu_{C_i}(z) = \alpha_i \wedge \mu_{C_i}(z)$$

$$R_P: \mu_{C_i}(z) = \alpha_i \mu_{C_i}(z)$$

其中

$$\alpha_i = \mu_{A_i}(x_0) \wedge \mu_{B_i}(y_0)$$

根据性质 3, 这个推论的结论是显然的。

结合性质 2 和性质 3, 可以得到

$$R_C: \mu_{C_i}(z) = \bigcup_{i=1}^n \alpha_i \wedge \mu_{C_i}(z)$$

$$R_P: \mu_{C_i}(z) = \bigcup_{i=1}^n \alpha_i \mu_{C_i}(z)$$

这里 α_i 可以看成是相应于第 i 条规则的加权因子, 它也看成是第 i 条规则的适用程度, 或者看成是第 i 条规则对模糊控制作用所产生的贡献的大小。这样的认识可以帮助我们对模糊控制机理的理解。

为了便于说明问题, 假设有如下的两条模糊控制规则

R_1 : 如果 x 是 A_1 and y 是 B_1 则 z 是 C_1

R_2 : 如果 x 是 A_2 and y 是 B_2 则 z 是 C_2

前面举例(例 2.8—例 2.10)说明了当 x, y, z 的论域为离散量且为有限时的推理计算方法。由于这时模糊集合可用模糊向量来表示, 模糊关系可用模糊矩阵来表示, 因此推理计算可表示成相应的向量和矩阵运算。

图 2.10 和图 2.11 表示了当论域为连续时模糊推理计算的方法, 图 2.10 相应于模糊蕴含计算采用 R_C 的情况, 图 2.11 相应于模糊蕴含计算用 R_P 的情况。图 2.12 和图 2.13 表示了输入为单点模糊集合时模糊推理的图示计算方法。

2.5.3 模糊控制中几种常见的模糊推理类型

1. 第一种模糊推理——模糊蕴含运算采用 Mamdani 的最小运算规则

这种模糊推理方法就是上面讨论得较多的方法, 即模糊蕴含运算采用 R_C , 从而有

$$\mu_{C_i}(z) = \alpha_i \wedge \mu_{C_i}(z)$$

$$\mu_{C'}(z) = \mu_{C_1}(z) \vee \mu_{C_2}(z) = [\alpha_1 \wedge \mu_{C_1}(z)] \vee [\alpha_2 \wedge \mu_{C_2}(z)]$$

α_1 和 α_2 可根据性质 3 及相应推论进行计算。图 2.10 和图 2.12 利用图形对这第一种模糊推理方法进行了解释。

2. 第二种模糊推理——模糊蕴含运算采用 Larsen 的积运算规则

$$\mu_{C_i}(z) = \alpha_i \mu_{C_i}(z)$$

$$\mu_{C'}(z) = \mu_{C_1}(z) \vee \mu_{C_2}(z) = [\alpha_1 \mu_{C_1}(z)] \vee [\alpha_2 \mu_{C_2}(z)]$$

图 2.11 和图 2.13 利用图形解释了该推理过程。

3. 第三种模糊推理——语言变量的隶属度函数为单调的 Tsukamoto 方法

这是 Tsukamoto 提出的方法, 它是第一种推理方法当 A_i, B_i 和 C_i 的隶属度函数为单

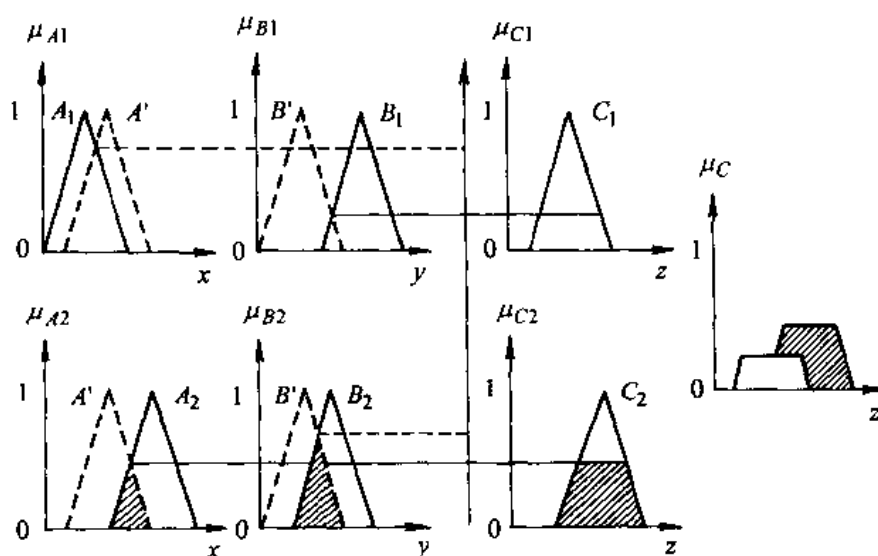


图 2.10 采用 R_c 时的模糊推理计算

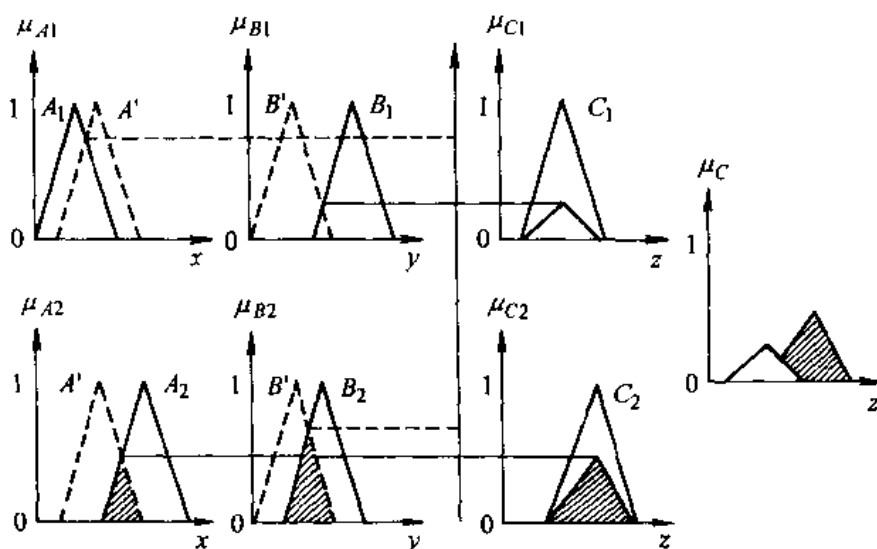


图 2.11 采用 R_p 时的模糊推理计算

调函数时的特例。实质上,对 A_i 和 B_i 的隶属度函数单调性的限制条件可以取消,只要求 C_i 的隶属度函数为单调即可。

对于该方法,首先根据第一条规则求出 α_1 ,根据 $\alpha_1 = C_1(z_1)$ 而求得 z_1 ;再根据第二条规则求出 α_2 ,根据 $\alpha_2 = C_2(z_2)$ 求得 z_2 ,准确的输出量可表示为 z_1 和 z_2 的加权组合(参见图 2.14),即

$$z_0 = \frac{\alpha_1 z_1 + \alpha_2 z_2}{\alpha_1 + \alpha_2}$$

4. 第四种模糊推理——规则的结论是输入语言变量的函数

在这种推理方式中,模糊控制规则具有如下形式

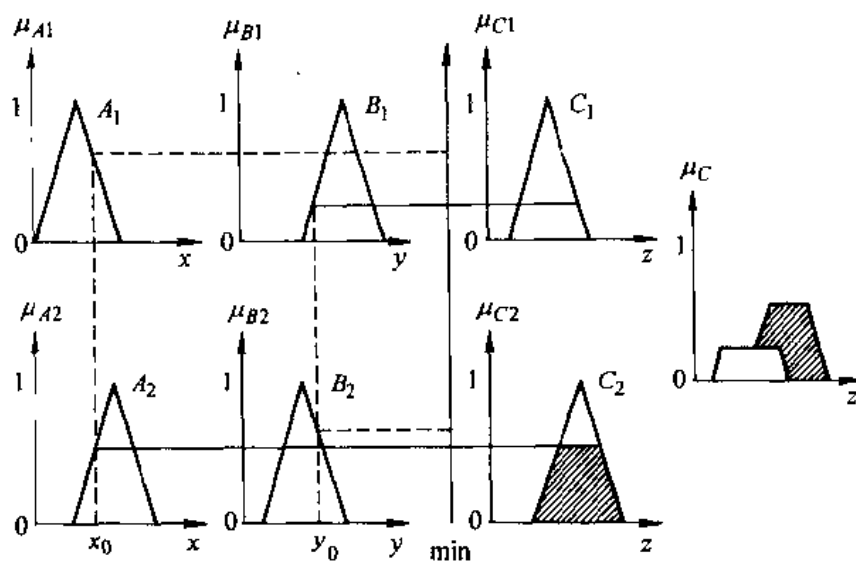


图 2.12 输入为模糊单点且采用 R_{\min} 时的模糊推理计算

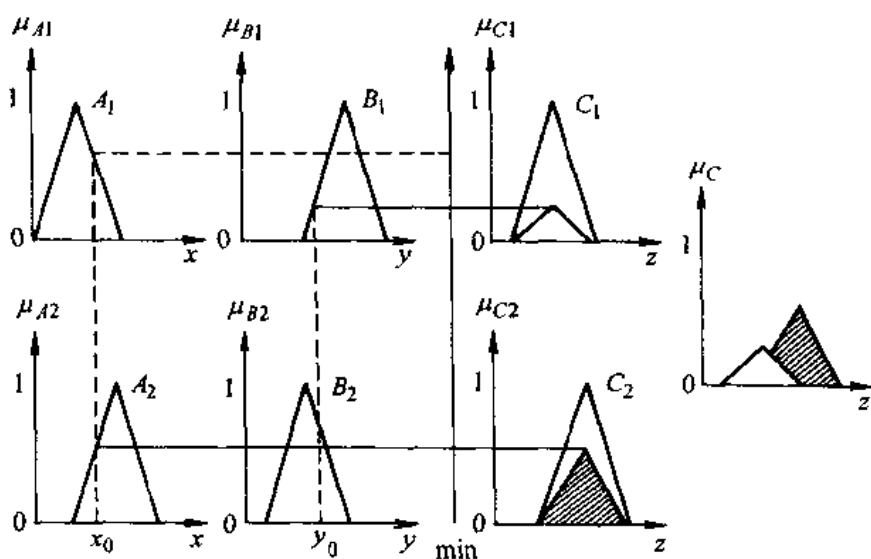


图 2.13 输入为模糊单点且采用 R_P 时的模糊推理计算

R_i : 如果 (x 是 A_i ... and y 是 B_i) 则 $z = f_i(x, \dots, y)$

其中 x, \dots, y 表示过程状态变量和控制变量, A_i, \dots, B_i 是语言变量 x, \dots, y 的语言变量值。

为简单起见,仍假定有如下两条控制规则

R_1 : 如果 x 是 A_1 and y 是 B_1 则 $z = f_1(x, y)$

R_2 : 如果 x 是 A_2 and y 是 B_2 则 $z = f_2(x, y)$

根据第一条规则,控制作用的推断值为 $\alpha_1 f_1(x_0, y_0)$,第二条规则的推断值为 $\alpha_2 f_2(x_0, y_0)$,总的控制作用是 $f_1(x_0, y_0)$ 和 $f_2(x_0, y_0)$ 的加权组合,即

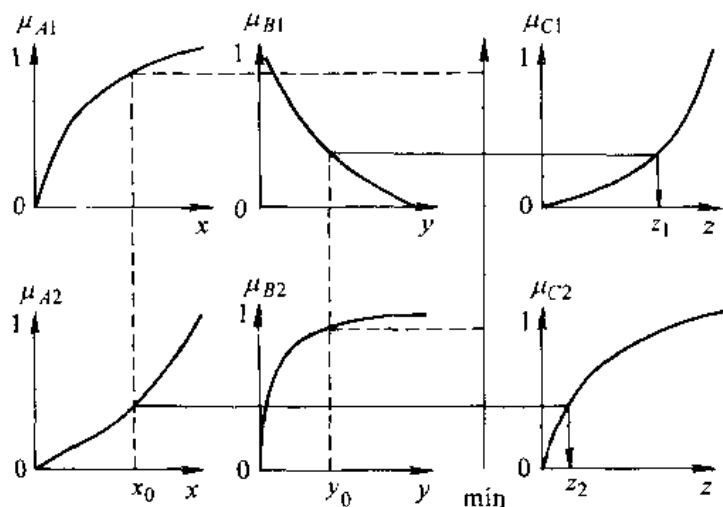


图 2.14 采用第三种方法的模糊推理

$$z_0 = \frac{\alpha_1 f_1(x_0, y_0) + \alpha_2 f_2(x_0, y_0)}{\alpha_1 + \alpha_2}$$

该方法是由 Takagi 和 Sugeno 提出来的,而且已用来控制一个模型小车沿着弯曲的轨道运动,并控制汽车停靠到车库中。利用该方法取得了满意的结果。

2.6 模糊控制的基本原理

2.6.1 模糊控制器的基本结构和组成

图 2.15 表示了模糊控制器的基本结构

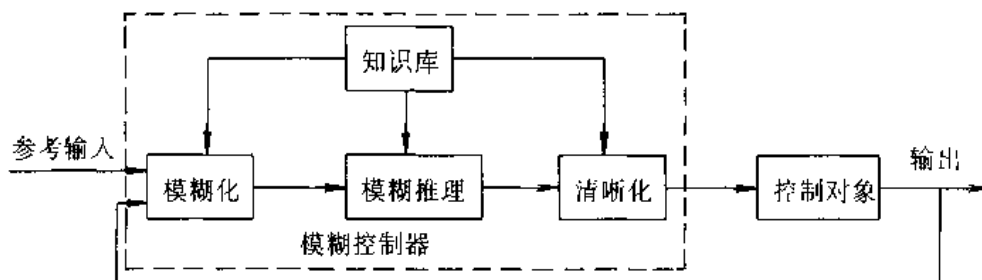


图 2.15 模糊控制器的结构图

1. 模糊控制器的组成

模糊控制器主要由以下 4 部分组成

(1) 模糊化

这部分的作用是将输入的精确量转换成模糊化量。其中输入量包括外界的参考输入、系统的输出或状态等。模糊化的具体过程如下:

① 首先对这些输入量进行处理以变成模糊控制器要求的输入量。例如,常见的情况

是计算 $e = r - y$ 和 $\dot{e} = de/dt$, 其中 r 表示参考输入, y 表示系统输出, e 表示误差。有时为了减小噪声的影响, 常常对 \dot{e} 进行滤波后再使用, 例如可取 $\dot{e} = [s/(Ts+1)]e$ 。

② 将上述已经处理过的输入量进行尺度变换, 使其变换到各自的论域范围。

③ 将已经变换到论域范围的输入量进行模糊处理, 使原先精确的输入量变成模糊量, 并用相应的模糊集合来表示。

(2) 知识库

知识库中包含了具体应用领域中的知识和要求的控制目标。它通常由数据库和模糊控制规则库两部分组成。

① 数据库主要包括各语言变量的隶属度函数, 尺度变换因子以及模糊空间的分级数等。

② 规则库包括了用模糊语言变量表示的一系列控制规则。它们反映了控制专家的经验 and 知识。

(3) 模糊推理

模糊推理是模糊控制器的核心, 它具有模拟人的基于模糊概念的推理能力, 该推理过程是基于模糊逻辑中的蕴含关系及推理规则来进行的。

(4) 清晰化

清晰化的作用是将模糊推理得到的控制量(模糊量)变换为实际用于控制的清晰量。它包含以下两部分内容:

① 将模糊的控制量经清晰化变换变成表示在论域范围的清晰量。

② 将表示在论域范围的清晰量经尺度变换变成实际的控制量。

2. 模糊条件句与模糊控制规则

正如前面所说, 模糊控制是模仿人的一种控制方法。在模糊控制中, 通过用一组语言描述的规则来表示专家的知识, 专家知识通常具有如下的形式:

IF(满足一组条件) THEN(可以推出一组结论)

在 IF-THEN 规则中的前提和结论均是模糊的概念。如“若温度偏高, 则加入较多的冷却水”, 其中“偏高”和“较多”均为模糊量。常常称这样的 IF-THEN 规则为模糊条件句。因此在模糊控制中, 模糊控制规则也就是模糊条件句。其中前提为具体应用领域中的条件, 结论为要采取的控制行动。IF-THEN 的模糊控制规则为表示控制领域的专家知识提供了方便的工具。对于多输入多输出(MIMO)模糊系统, 则有多个前提和多个结论。例如, 对于两输入单输出(MISO)系统, 模糊控制规则具有如下的形式:

R_1 : 如果 x 是 A_1 and y 是 B_1 则 z 是 C_1

R_2 : 如果 x 是 A_2 and y 是 B_2 则 z 是 C_2 ;

\vdots

R_n : 如果 x 是 A_n and y 是 B_n 则 z 是 C_n

其中 x, y 和 z 均为语言变量, x 和 y 为输入量, z 为控制量。 A_i, B_i 和 $C_i (i=1, 2, \dots, n)$ 分别是语言变量 x, y, z 在其论域 X, Y, Z 上的语言变量值, 所有规则组合在一起构成了规则库。对于其中的一条规则

R_i : 如果 x 是 A_i and y 是 B_i , 则 z 是 C_i .

其模糊蕴含关系定义为

$$\begin{aligned}\mu_{R_i} &= \mu_{(A_i \text{ and } B_i \rightarrow C_i)}(x, y, z) \\ &= [\mu_{A_i}(x) \text{ and } \mu_{B_i}(y)] \rightarrow \mu_{C_i}(z)\end{aligned}$$

其中“ A_i and B_i ”是定义在 $X \times Y$ 上的模糊集合 $A_i \times B_i$, $R_i = (A_i \text{ and } B_i) \rightarrow C_i$ 是定义在 $X \times Y \times Z$ 上的模糊蕴含关系。 R_i 的具体运算方式已如上节所述。

3. 模糊控制中的几个基本运算操作

(1) 模糊化运算

$$x = \text{fz}(x_0)$$

其中 x_0 是输入的清晰量, x 是模糊集合, fz 表示模糊化运算符(fuzzifier)。

(2) 句子连接运算

$$R = \text{also}(R_1, R_2, \dots, R_n)$$

其中 $R_i (i = 1, 2, \dots, n)$ 是第 i 条规则所表示的模糊蕴含关系。 R 是 n 个模糊关系的组合, 组合运算用符号 also 表示。具体运算方法已在上节进行了讨论。

(3) 合成运算

$$z = (x \text{ and } y) \circ R$$

其中 x 和 y 是输入模糊量, z 是输出模糊量, and 是句子连接运算符, “ \circ ”是合成运算符, and 和“ \circ ”的具体运算已在上节进行了讨论。

(4) 清晰化运算

以上推理过程得到的输出量 z 仍是模糊量, 而实际的控制必须为清晰量。因此要进行如下的清晰化运算

$$z_0 = \text{df}(z)$$

其中 z_0 为控制输出的清晰化量, df 表示清晰化运算符(defuzzifier)。

2.6.2 模糊化运算

模糊化运算是将输入空间的观测量映射为输入论域上的模糊集合。模糊化在处理不确定信息方面具有重要的作用。在模糊控制中, 观测到的数据常常是清晰量。由于模糊控制器对数据进行处理是基于模糊集合的方法。因此对输入数据进行模糊化是必不可少的一步。在进行模糊化运算之前, 首先需要对输入量进行尺度变换, 使其变换到相应的论域范围(后面将对此进行专门讨论)。下面所讨论的模糊化运算中的输入量均假定为已经过尺度变换的量。

在模糊控制中主要采用以下两种模糊化方法。

1. 单点模糊集合

如果输入量数据 x_0 是准确的, 则通常将其模糊化为单点模糊集合。设该模糊集合用 A 表示, 则有

$$\mu_A(x) = \begin{cases} 1 & x = x_0 \\ 0 & x \neq x_0 \end{cases}$$

其隶属度函数如图 2.16 所示。

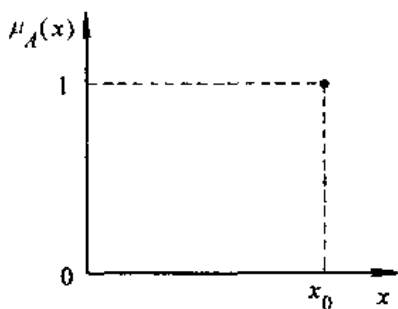


图 2.16 单点模糊集合的隶属度函数

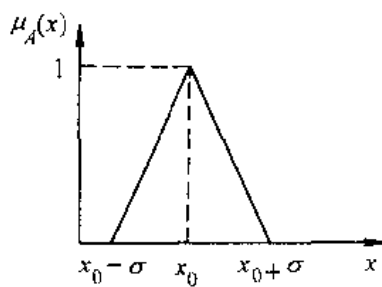


图 2.17 三角形模糊集合的隶属度函数

这种模糊化方法只是形式上将清晰量转变成了模糊量,而实质上它表示的仍是准确量。在模糊控制中,当测量数据准确时,采用这样的模糊化方法是十分自然和合理的。

2. 三角形模糊集合

如果输入量数据存在随机测量噪声,这时模糊化运算相当于将随机量变换为模糊量。对于这种情况,可以取模糊量的隶属度函数为等腰三角形,如图 2.17 所示。三角形的顶点相应于该随机数的均值,底边的长度等于 2σ , σ 表示该随机数据的标准差。隶属度函数取为三角形主要是考虑其表示方便,计算简单。另一种常用的方法是取隶属度函数为铃形函数,即

$$\mu_A(x) = e^{-\frac{(x-x_0)^2}{2\sigma^2}}$$

它也就是正态分布的函数。

2.6.3 数据库

如前所述,模糊控制器中的知识库有两部分组成:数据库和模糊控制规则库。本节首先讨论数据库。数据库中包含了与模糊控制规则及模糊数据处理有关的各种参数,其中包括尺度变换参数、模糊空间分割和隶属度函数的选择等。

1. 输入量变换

对于实际的输入量,第一步首先需要进行尺度变换,将其变换到要求的论域范围。变换的方法可以是线性的,也可以是非线性的。例如,若实际的输入量为 x_0^* ,其变化范围为 $[x_{\min}^*, x_{\max}^*]$,若要求的论域为 $[x_{\min}, x_{\max}]$,若采用线性变换,则

$$x_0 = \frac{x_{\min} + x_{\max}}{2} + k \left(x_0^* - \frac{x_{\max}^* + x_{\min}^*}{2} \right)$$

$$k = \frac{x_{\max} - x_{\min}}{x_{\max}^* - x_{\min}^*}$$

其中 k 称为比例因子。

论域可以是连续的也可以是离散的。如果要求离散的论域,则需要将连续的论域离散化或量化。量化可以是均匀的,也可以是非均匀的。表 2.6 和表 2.7 中分别表示均匀量化和非均匀量化的情形。

表 2.6 均匀量化

量化等级	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
变化范围	≤ -5.5	$(-5.5, -4.5]$	$(-4.5, -3.5]$	$(-3.5, -2.5]$	$(-2.5, -1.5]$	$(-1.5, -0.5]$	$(-0.5, 0.5]$	$(0.5, 1.5]$	$(1.5, 2.5]$	$(2.5, 3.5]$	$(3.5, 4.5]$	$(4.5, 5.5]$	> 5.5

表 2.7 非均匀量化

量化等级	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
变化范围	≤ -3.2	$(-3.2, -1.6]$	$(-1.6, -0.8]$	$(-0.8, -0.4]$	$(-0.4, -0.2]$	$(-0.2, -0.1]$	$(-0.1, 0.1]$	$(0.1, 0.2]$	$(0.2, 0.4]$	$(0.4, 0.8]$	$(0.8, 1.6]$	$(1.6, 3.2]$	> 3.2

2. 输入和输出空间的模糊分割

模糊控制规则中前提的语言变量构成模糊输入空间,结论的语言变量构成模糊输出空间。每个语言变量的取值为一组模糊语言名称,它们构成了语言名称的集合。每个模糊语言名称相应一个模糊集合。对于每个语言变量,其取值的模糊集合具有相同的论域。模糊分割是要确定对于每个语言变量取值的模糊语言名称的个数,模糊分割的个数决定了模糊控制精细化的程度。这些语言名称通常均具有一定的含义。如 NB : 负大(Negative Big); NM : 负中(Negative Medium); NS : 负小(Negative Small); ZE : 零(Zero); PS : 正小(Positive Small); PM : 正中(Positive Medium); PB : 正大(Positive Big)。图 2.18 表示了两个模糊分割的例子,论域均为 $[-1, +1]$,隶属度函数的形状为三角形或梯形。图 2.18(a)所示为模糊分割较粗的情况,图 2.18(b)为模糊分割较细的情况。图中所示的论域为正则化(normalization)的情况,即 $x \in [-1, +1]$,且模糊分割是完全对称的。这里假设尺度变换时已经作了预处理而变换成这样的标准情况。一般情况,模糊语言名称也可为非对称和非均匀地分布。

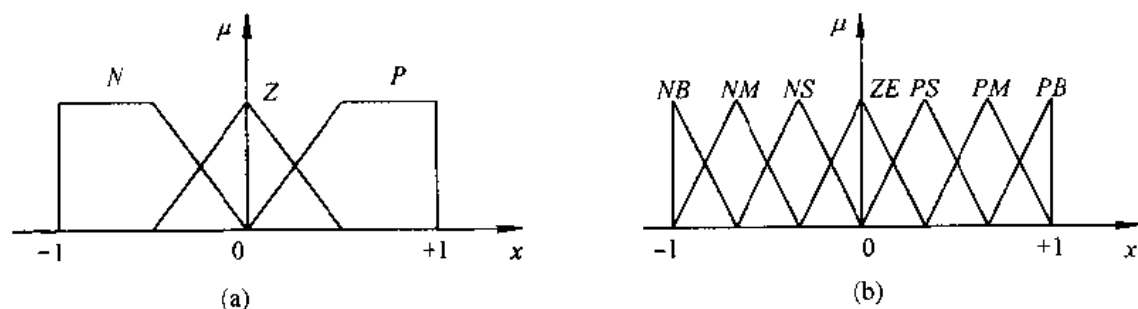


图 2.18 模糊分割的图形表示

模糊分割的个数也决定了最大可能的模糊规则的个数。如对于两输入单输出的模糊系统, x 和 y 的模糊分割数分别为 3 和 7, 则最大可能的规则数为 $3 \times 7 = 21$ 。可见, 模糊分割数越多, 控制规则数也越多, 所以模糊分割不可太细, 否则需要确定太多的控制规则, 这也是很困难的一件事。当然, 模糊分割数太小将导致控制太粗略, 难以对控制性能进行精

心的调整。目前尚没有一个确定模糊分割数的指导性的方法和步骤,它仍主要依靠经验和试凑。

3. 完备性

对于任意的输入,模糊控制器均应给出合适的控制输出,这个性质称为完备性。模糊控制的完备性取决于数据库或规则库。

(1) 数据库方面

对于任意的输入,若能找到一个模糊集合,使该输入对于该模糊集合的隶属度函数不小于 ϵ ,则称该模糊控制器满足 ϵ 完备性。图 2.18 所示即为 $\epsilon=0.5$ 的情况,它也是最常见的选择。

(2) 规则库方面

模糊控制的完备性对于规则库的要求是,对于任意的输入应确保至少有一个可适用的规则,而且规则的适用度应大于某个数,譬如说 0.5。根据完备性的要求,控制规则数不可太少。

4. 模糊集合的隶属度函数

根据论域为离散和连续的不同情况,隶属度函数的描述也有如下两种方法。

(1) 数值描述方法

对于论域为离散,且元素个数为有限时,模糊集合的隶属度函数可以用向量或者表格的形式来表示。表 2.8 给出了用表格表示的一个例子。

表 2.8 数值方法描述的隶属度

元素 隶属度 模糊集合	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
NB	1.0	0.7	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
NM	0.3	0.7	1.0	0.7	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
NS	0.0	0.0	0.3	0.7	1.0	0.7	0.3	0.0	0.0	0.0	0.0	0.0	0.0
ZE	0.0	0.0	0.0	0.0	0.3	0.7	1.0	0.7	0.3	0.0	0.0	0.0	0.0
PS	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.7	1.0	0.7	0.3	0.0	0.0
PM	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.7	1.0	0.7	0.3
PB	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.7	1.0

在上面的表格中,每一行表示一个模糊集合的隶属度函数。例如

$$NS = \frac{0.3}{-4} + \frac{0.7}{-3} + \frac{1}{-2} + \frac{0.7}{-1} + \frac{0.3}{0}$$

(2) 函数描述方法

对于论域为连续的情况,隶属度常常用函数的形式来描述,最常见的有铃形函数、三角形函数、梯形函数等。下面给出铃形隶属度函数的解析式子

$$\mu_A(x) = e^{-\frac{(x-x_0)^2}{2\sigma^2}}$$

其中 x_0 是隶属度函数的中心值, σ^2 是方差。图 2.19 表示了铃形隶属度函数的分布图。

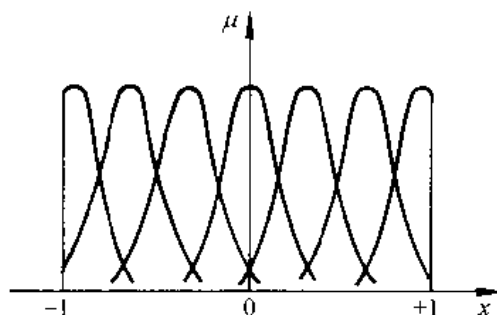


图 2.19 函数描述的隶属度函数

隶属度函数的形状对模糊控制器的性能有很大影响。当隶属度函数比较窄瘦时,控制较灵敏,反之,控制较粗略和平稳。通常当误差较小时,隶属度函数可取得较为窄瘦,误差较大时,隶属度函数可取得宽胖些。

2.6.4 规则库

模糊控制规则库是由一系列“IF-THEN”型的模糊条件句所构成。条件句的前件为输入和状态,后件为控制变量。

1. 模糊控制规则的前件和后件变量的选择

模糊控制规则的前件和后件变量也即模糊控制器的输入和输出的语言变量。输出量即为控制量,它一般比较容易确定。输入量选什么以及选几个则需要根据要求来确定。输入量比较常见的是误差 e 和它的导数 \dot{e} ,有时还可以包括它的积分 $\int e dt$ 等。输入和输出语言变量的选择以及它们隶属函数的确定对于模糊控制器的性能有着十分关键的作用。它们的选择和确定主要依靠经验和工程知识。

2. 模糊控制规则的建立

模糊控制规则是模糊控制的核心。因此如何建立模糊控制规则也就成为一个十分关键的问题。下面将讨论 4 种建立模糊控制规则的方法。它们之间并不是互相排斥的,相反,若能结合这几种方法则可以更好地帮助建立模糊规则库。

(1) 基于专家的经验和控制工程知识

模糊控制规则具有模糊条件句的形式,它建立了前件中的状态变量与后件中的控制变量之间的联系。我们在日常生活中用于决策的大部分信息主要是基于语义的方式而非数值的方式。因此,模糊控制规则是对人类行为和进行决策分析过程的最自然的描述方式。这也就是它为什么采用 IF-THEN 形式的模糊条件句的主要原因。

基于上面的讨论,通过总结人类专家的经验,并用适当的语言来加以表述,最终可表示成模糊控制规则的形式。一个典型的例子是,人们对人工控制水泥窑的操作手册进行总结归纳,最终建立起了模糊控制规则库。另一种方式是通过向有经验的专家和操作人员咨询,从而获得特定应用领域模糊控制规则的原型。在此基础上,再经一定的试凑和调整,可获得具有更好性能的控制规则。

(2) 基于操作人员的实际控制过程

在许多人工控制的工业系统中,很难建立控制对象的模型,因此用常规的控制方法来进行设计和仿真比较困难。而熟练的操作人员却能成功地控制这样的系统。事实上,操作人员有意或无意地使用了一组 IF-THEN 模糊规则来进行控制。但是他们往往并不能用语言明确地将它们表达出来,因此可以通过记录操作人员实际控制过程时的输入输出数据,并从中总结出模糊控制规则。

(3) 基于过程的模糊模型

控制对象的动态特性通常可用微分方程、传递函数、状态方程等数学方法来加以描述,这样的模型称为定量模型或清晰化模型。控制对象的动态特性也可以用语言的方法来描述,这样的模型称为定性模型或模糊模型。基于模糊模型,也能建立起相应的模糊控制规律。这样设计的系统是纯粹的模糊系统,即控制器和控制对象均是用模糊的方法来加以描述的,因而它比较适合于采用理论的方法来进行分析和控制。

(4) 基于学习

许多模糊控制主要是用来模仿人的决策行为,但很少具备有类似于人的学习功能,即根据经验和知识产生模糊控制规则并对它们进行修改的能力。Mamdani 于 1979 年首先提出了模糊自组织控制,它便是一种具有学习功能的模糊控制。该自组织控制具有分层递阶的结构,它包含有两个规则库。第一个规则库是一般的模糊控制的规则库,第二个规则库由宏规则组成,它能够根据对系统的整体性能要求来产生并修改一般的模糊控制规则,从而显示了类似人的学习能力。自 Mamdani 的工作之后,近来又有不少人在这方面作了大量的研究工作。最典型的例子是 Sugeno 的模糊小车,它是具有学习功能的模糊控制车,经过训练后它能够自动地停靠在要求的位置。

3. 模糊控制规则的类型

在模糊控制中,目前主要应用如下两种形式的模糊控制规则。

(1) 状态评估模糊控制规则。它具有如下的形式:

R_1 : 如果 x 是 A_1 and y 是 B_1 则 z 是 C_1

R_2 : 如果 x 是 A_2 and y 是 B_2 则 z 是 C_2

⋮

R_n : 如果 x 是 A_n and y 是 B_n 则 z 是 C_n

在现有的模糊控制系统中,大多数情况均采用这种形式。我们前面所讨论的也都是这种情形。

对于更一般的情形,模糊控制规则的后件可以是过程状态变量的函数,即

R_i : 如果 x 是 A_i and y 是 B_i 则 $z = f_i(x, \dots, y)$

它根据对系统状态的评估按照一定的函数关系计算出控制作用 z 。

(2) 目标评估模糊控制规则。典型的形式如下所示

R_i : 如果 $[u \text{ 是 } C_i \rightarrow (x \text{ 是 } A_i \text{ and } y \text{ 是 } B_i)]$ 则 u 是 C_i

其中 u 是系统的控制量, x 和 y 表示要求的状态和目标或者是对系统性能的评估,因而 x 和 y 的取值常常是“好”、“差”等模糊语言。对于每个控制命令 C_i ,通过预测相应的结果 (x, y) ,从中选用最适合的控制规则。

上面的规则可进一步解释为:当控制命令选 C_i 时,如果性能指标 x 是 A_i , y 是 B_i 时,那么选用该条规则且将 C_i 取为控制器的输出。例如,用在日本仙台的地铁模糊自动火车运行系统中,就采用了这种类型的模糊控制规则。列出其中典型的一条如“如果控制标志不改变则火车停在预定的容许区域,那么控制标志不改变”。

采用目标评估模糊控制规则,它对控制的结果加以预测,并根据预测的结果来确定采取的控制行动。因此它本质上是一种模糊预报控制。本章一开始所介绍的模糊自动火车运行系统及模糊自动集装箱吊车操纵系统均采用了这样的控制方法。

4. 模糊控制规则的其它性能要求

(1) 完备性。

前面讨论数据库时已对其进行了讨论。即对于任意的输入应确保它至少有一个可适用的规则,而且规则的适用程度应大于一定的数,譬如 0.5。

(2) 模糊控制规则数。

若模糊控制器的输入有 m 个,每个输入的模糊分级数分别为 n_1, n_2, \dots, n_m , 则最大可能的模糊规则数为 $N_{\max} = n_1 n_2 \dots n_m$, 实际的模糊控制数应该取多少取决于很多因素,目前尚无普遍适用的一般步骤。总的原则是,在满足完备性的条件下,尽量取较少的规则数,以简化模糊控制器的设计和实现。

(3) 模糊控制规则的一致性。

模糊控制规则主要基于操作人员的经验,它取决于对多种性能的要求,而不同的性能指标要求往往互相制约,甚至是互相矛盾的。这就要求按这些指标要求确定的模糊控制不能出现互相矛盾的情况。

2.6.5 模糊推理与清晰化计算

1. 模糊推理

前面已对模糊推理进行了专门的讨论,这里再扼要叙述如下。

对于多输入多输出(MIMO)模糊控制器,其规则库具有如下形式:

$$R = \{R_{\text{MIMO}}^1, R_{\text{MIMO}}^2, \dots, R_{\text{MIMO}}^r\}$$

其中

R_{MIMO}^i : 如果(x 是 A_i and \dots and y 是 B_i) 则(z_1 是 C_{i1}, \dots, z_q 是 C_{iq})

R_{MIMO}^i 的前件是直积空间 $X \times \dots \times Y$ 上的模糊集合,后件是 q 个控制作用的并,它们之间是互相独立的。因此 R_{MIMO}^i 可以看成是 q 个独立的 MISO 规则,即

$$R_{\text{MIMO}}^i = \{R_{\text{MISO}}^{i1}, R_{\text{MISO}}^{i2}, \dots, R_{\text{MISO}}^{iq}\}$$

其中

R_{MISO}^{ij} : 如果(x 是 A_i and \dots and y 是 B_i) 则(z_j 是 C_{ij})

因此只需考虑 MISO 子系统的模糊推理问题。

不失一般性,考虑两个输入一个输出的模糊控制器。设已建立的模糊控制规则库为

R_1 : 如果 x 是 A_1 and y 是 B_1 则 z 是 C_1

R_2 : 如果 x 是 A_2 and y 是 B_2 则 z 是 C_2

⋮

R_n : 如果 x 是 A_n and y 是 B_n 则 z 是 C_n

设已知模糊控制器的输入模糊量为: x 是 A' and y 是 B' , 则根据模糊控制规则进行近似推理, 可以得出输出模糊量 z (用模糊集合 C' 表示) 为

$$C' = (A' \text{ and } B') \circ R$$

$$R = \bigcup_{i=1}^n R_i$$

$$R_i = (A_i \text{ and } B_i) \rightarrow C_i$$

其中包括了三种主要的模糊逻辑运算: and 运算, 合成运算“ \circ ”, 蕴含运算“ \rightarrow ”。and 运算通常采用求交(取小)或求积(代数积)的方法; 合成运算“ \circ ”通常采用最大-最小或最大-积(代数积)的方法; 蕴含运算“ \rightarrow ”通常采用求交(R_C)或求积(R_P)的方法。

2. 清晰化计算

以上通过模糊推理得到的是模糊量, 而对于实际的控制则必须为清晰量, 因此需要将模糊量转换成清晰量, 这就是清晰化计算所要完成的任务。清晰化计算通常有以下几种方法。

(1) 最大隶属度法

若输出量模糊集合 C' 的隶属度函数只有一个峰值, 则取隶属度函数的最大值为清晰值, 即

$$\mu_{C'}(z_0) \geq \mu_{C'}(z) \quad z \in Z$$

其中 z_0 表示清晰值。若输出量的隶属度函数有多个极值, 则取这些极值的平均值为清晰值。

例 2.11 已知输出量 z_1 的模糊集合为

$$C_1' = \frac{0.1}{2} + \frac{0.4}{3} + \frac{0.7}{4} + \frac{1.0}{5} + \frac{0.7}{6} + \frac{0.3}{7}$$

z_2 的模糊集合为

$$C_2' = \frac{0.3}{-4} + \frac{0.8}{-3} + \frac{1}{-2} + \frac{1}{-1} + \frac{0.8}{0} + \frac{0.3}{1} + \frac{0.1}{2}$$

求相应的清晰量 z_{10} 和 z_{20} 。

根据最大隶属度法, 很容易求得

$$z_{10} = \text{df}(z_1) = 5$$

$$z_{20} = \text{df}(z_2) = \frac{-2-1}{2} = -1.5$$

(2) 中位数法

如图 2.20 所示, 采用中位数法是取 $\mu_{C'}(z)$ 的中位数作为 z 的清晰量, 即 $z_0 = \text{df}(z) = \mu_{C'}(z)$ 的中位数, 它满足

$$\int_a^{z_0} \mu_{C'}(z) dz = \int_{z_0}^b \mu_{C'}(z) dz$$

也就是说, 以 z_0 为分界, $\mu_{C'}(z)$ 与 z 轴之间面积两边相等。

(3) 加权平均法

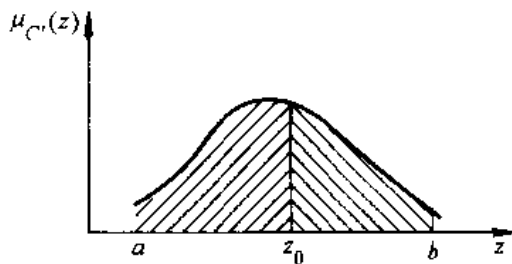


图 2.20 清晰化计算的中位数法

这种方法取 $\mu_C(z)$ 的加权平均值为 z 的清晰值,即

$$z_0 = df(z) = \frac{\int_a^b z \mu_C(z) dz}{\int_a^b \mu_C(z) dz}$$

它类似于重心的计算,所以也称重心法。对于论域为离散的情况则有

$$z_0 = \frac{\sum_{i=1}^n z_i \mu_C(z_i)}{\sum_{i=1}^n \mu_C(z_i)}$$

例 2.12 题设条件同例 2.11, 用加权平均法计算清晰值 z_{10} 和 z_{20} 。

解:

$$z_{10} = \frac{0.1 \times 2 + 0.4 \times 3 + 0.7 \times 4 + 1 \times 5 + 0.7 \times 6 + 0.3 \times 7}{0.1 + 0.4 + 0.7 + 1 + 0.7 + 0.3} = 4.84$$

$$z_{20} = \frac{0.3 \times (-4) + 0.8 \times (-3) + 1 \times (-2) + 1 \times (-1) + 0.8 \times 0 + 0.3 \times 1 + 0.1 \times 2}{0.3 + 0.8 + 1 + 1 + 0.8 + 0.3 + 0.1} = -1.42$$

在以上各种清晰化方法中,加权平均法应用最为普遍。

在求得清晰值 z_0 后,还需经尺度变换变为实际的控制量。变换的方法可以是线性的,也可以是非线性的。若 z_0 的变化范围为 $[z_{\min}, z_{\max}]$,实际控制量的变化范围为 $[u_{\min}, u_{\max}]$,若采用线性变换,则

$$u = \frac{u_{\max} + u_{\min}}{2} + k \left(z_0 - \frac{z_{\max} + z_{\min}}{2} \right)$$

$$k = \frac{u_{\max} - u_{\min}}{z_{\max} - z_{\min}}$$

其中 k 称为比例因子。

2.6.6 论域为离散时模糊控制的离线计算

当论域为离散时,经过量化后的输入量的个数是有限的。因此可以针对输入情况的不同组合离线计算出相应的控制量,从而组成一张控制表,实际控制时只要直接查这张控制表即可,在线的运算量是很少的。这种离线计算、在线查表的模糊控制方法比较容易满足

实时控制的要求。图 2.21 表示了这种模糊控制系统的结构,图中假设采用误差 e 和误差导数 \dot{e} 作为模糊控制器的输入量,这是最常使用的情况。

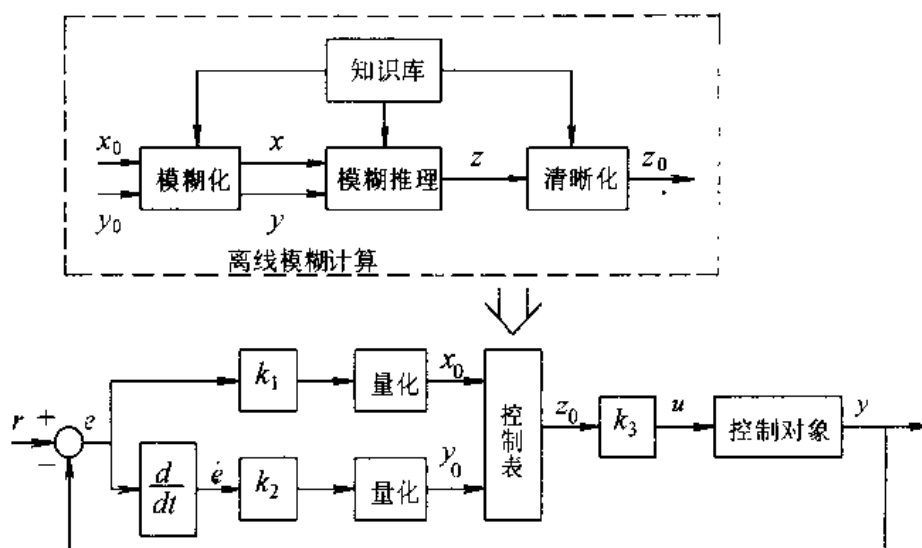


图 2.21 论域为离散时的模糊控制系统结构

图中 k_1 、 k_2 和 k_3 为尺度变换的比例因子。设 e 、 \dot{e} 和 u 的实际变化范围分别为 $[-e_m, e_m]$ 、 $[-\dot{e}_m, \dot{e}_m]$ 和 $[-u_m, u_m]$ ，并设 x, y 和 z 的论域分别为

$$\{-n_i, -n_i + 1, \dots, 0, 1, \dots, n_i\} \quad (i = 1, 2, 3)$$

则

$$k_1 = \frac{n_1}{e_m} \quad k_2 = \frac{n_2}{\dot{e}_m} \quad k_3 = \frac{u_m}{n_3}$$

图中量化的功能是将比例变换后的连续值经四舍五入变为整数量。

从 x_0, y_0 到 z_0 的模糊推理计算过程采用前面已经讨论过的方法进行。由于 x_0, y_0 的个数是有限的,因此可以将它们的所有可能的组合情况事先计算出来(即图中的离线模糊计算部分),将计算的结果列成一张控制表。实际控制时只需查询该控制表即可由 x_0, y_0 求得 z_0 。求得 z_0 后再经比例变换 k_3 变成实际的控制量。

在该例中控制器的输入量为 e 和 \dot{e} ,因此它相当于是非线性的 PD 控制, k_1, k_2 分别是比例项和导数项前面的比例系数,它们对系统性能有很大影响,要仔细地加以选择。 k_3 串联于系统的回路中,它直接影响整个回路的增益,因此 k_3 也对系统的性能有很大影响,一般说来, k_3 选得大,系统反应快。但过大有可能使系统不稳定。

下面通过一个具体例子来说明离线模糊计算的过程。设

$$X, Y, Z \in \{-6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6\}$$

$$T(x) = \{NB(\text{负大}), NM(\text{负中}), NS(\text{负小}), NZ(\text{负零}), PZ(\text{正零}),$$

$$PS(\text{正小}), PM(\text{正中}), PB(\text{正大})\}$$

$$T(y) = T(z) = \{NB, NM, NS, ZE, PS, PM, PB\}$$

表 2.9 表示语言变量 x 的隶属度函数。 y 和 z 的隶属度函数同表 2.8。

表 2.9 语言变量 x 的隶属度函数

隶属度 模糊集合 \ x	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
NB	1.0	0.8	0.7	0.4	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
NM	0.2	0.7	1.0	0.7	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
NS	0.0	0.1	0.3	0.7	1.0	0.7	0.2	0.0	0.0	0.0	0.0	0.0	0.0
NZ	0.0	0.0	0.0	0.0	0.1	0.6	1.0	0.0	0.0	0.0	0.0	0.0	0.0
PZ	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.6	0.1	0.0	0.0	0.0	0.0
PS	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.7	1.0	0.7	0.3	0.1	0.0
PM	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.7	1.0	0.7	0.3
PB	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.4	0.7	0.8	1.0

表 2.8 和表 2.9 是一种表示离散论域的模糊集合及其隶属度函数的简洁形式。例如对于表 2.9, 它表示

$$NB = \frac{1.0}{-6} + \frac{0.8}{-5} + \frac{0.7}{-4} + \frac{0.4}{-3} + \frac{0.1}{-2}$$

$$NM = \frac{0.2}{-6} + \frac{0.7}{-5} + \frac{1.0}{-4} + \frac{0.7}{-3} + \frac{0.3}{-2}$$

$$\vdots$$

$$PB = \frac{0.1}{2} + \frac{0.4}{3} + \frac{0.7}{4} + \frac{0.8}{5} + \frac{1.0}{6}$$

表 2.10 列出了该模糊控制器所采用的模糊控制规则。

表 2.10 模糊控制规则表

$\begin{matrix} y \\ z \\ x \end{matrix}$	NB	NM	NS	ZE	PS	PM	PB
NB	NB	NB	NB	NB	NM	ZE	ZE
NM	NB	NB	NB	NB	NM	ZE	ZE
NS	NM	NM	NM	NM	ZE	PS	PS
NZ	NM	NM	NS	ZE	PS	PM	PM
PZ	NM	NM	NS	ZE	PS	PM	PM
PS	NS	NS	ZE	PM	PM	PM	PM
PM	ZE	ZE	PM	PB	PB	PB	PB
PB	ZE	ZE	PM	PB	PB	PB	PB

表 2.10 是表示模糊控制规则的简洁形式。该表中共包含 56 条规则,由于 x 的模糊分割数为 8, y 的模糊分割数为 7, 所以该表包含了最大可能的规则数。一般情况下规则数可以少于 56, 这时表中相应栏内可以为空。表 2.10 中所表示的规则依次为

R_1 : 如果 x 是 NB and y 是 NB 则 z 是 NB

R_2 : 如果 x 是 NB and y 是 NM 则 z 是 NB

\vdots

R_{56} : 如果 x 是 PB and y 是 PB 则 z 是 PB

设已知输入为 x_0 和 y_0 , 模糊化运算采用单点模糊集合, 则相应的输入量模糊集合 A' 和 B' 分别为

$$\mu_{A'}(x) = \begin{cases} 1 & x = x_0 \\ 0 & x \neq x_0 \end{cases} \quad \mu_{B'}(y) = \begin{cases} 1 & y = y_0 \\ 0 & y \neq y_0 \end{cases}$$

根据前面介绍的模糊推理方法及性质, 可求得输出量的模糊集合 C' 为 (假设 and 用求交法, also 用求并法, 合成用最大-最小法, 模糊蕴含用求交法)

$$\begin{aligned} C' &= (A' \times B') \circ R = (A' \times B') \circ \bigcup_{i=1}^{56} R_i \\ &= \bigcup_{i=1}^{56} (A' \times B') \circ [(A_i \times B_i) \rightarrow C_i] \\ &= \bigcup_{i=1}^{56} [A' \circ (A_i \rightarrow C_i)] \cap [B' \circ (B_i \rightarrow C_i)] \\ &= \bigcup_{i=1}^{56} C'_{iA} \cap C'_{iB} \\ &= \bigcup_{i=1}^{56} C'_i \end{aligned}$$

下面以 $x_0 = -6, y_0 = -6$ 为例说明计算过程。

$$R_{1A} = A_1 \rightarrow C_1 = A_{NB} \rightarrow C_{NB} = \begin{bmatrix} 1 \\ 0.8 \\ 0.7 \\ 0.4 \\ 0.1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \wedge [1 \quad 0.7 \quad 0.3 \quad 0 \quad \cdots \quad 0]$$

$$= \begin{bmatrix} 1 & 0.7 & 0.3 \\ 0.8 & 0.7 & 0.3 \\ 0.7 & 0.7 & 0.3 & 0 \\ 0.4 & 0.4 & 0.3 \\ 0.1 & 0.1 & 0.1 \\ & 0 & 0 \end{bmatrix}_{13 \times 13}$$

$$C'_{1A} = A' \circ (A_1 \rightarrow C_1) = [1 \quad 0 \quad \cdots \quad 0] \circ R_{1A} = [1 \quad 0.7 \quad 0.3 \quad 0 \quad \cdots \quad 0]$$

$$R_{1B} = B_1 \rightarrow C_1 = B_{NB} \rightarrow C_{NB} = \begin{bmatrix} 1.0 \\ 0.7 \\ 0.3 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \wedge [1 \quad 0.7 \quad 0.3 \quad 0 \quad \cdots \quad 0]$$

$$= \begin{bmatrix} 1 & 0.7 & 0.3 & & \\ 0.7 & 0.7 & 0.3 & 0 & \\ 0.3 & 0.3 & 0.3 & & \\ & 0 & & 0 & \end{bmatrix}_{13 \times 13}$$

$$C'_{1B} = B' \circ (B_1 \rightarrow C_1) = [1 \quad 0 \quad \cdots \quad 0] \circ R_{1B} = [1 \quad 0.7 \quad 0.3 \quad 0 \quad \cdots \quad 0]$$

$$C'_1 = C'_{1A} \cap C'_{1B} = [1 \quad 0.7 \quad 0.3 \quad 0 \quad \cdots \quad 0]$$

$$R_{2A} = A_2 \rightarrow C_2 = A_{NB} \rightarrow C_{NB} = R_{1A}$$

$$C'_{2A} = A' \circ (A_2 \rightarrow C_2) = C'_{1A}$$

$$R_{2B} = B_2 \rightarrow C_2 = B_{NB} \rightarrow C_{NB} = \begin{bmatrix} 0.3 \\ 0.7 \\ 1.0 \\ 0.7 \\ 0.3 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \wedge [1 \quad 0.7 \quad 0.3 \quad 0 \quad \cdots \quad 0]$$

$$= \begin{bmatrix} 0.3 & 0.3 & 0.3 & & \\ 0.7 & 0.7 & 0.3 & 0 & \\ 1 & 0.7 & 0.3 & & \\ 0.7 & 0.7 & 0.3 & & \\ 0.3 & 0.3 & 0.3 & & \\ & 0 & & 0 & \end{bmatrix}_{13 \times 13}$$

$$C'_{2B} = B' \circ (B_2 \rightarrow C_2) = B' \circ R_{2B} = [0.3 \quad 0.3 \quad 0.3 \quad 0 \quad \cdots \quad 0]$$

$$C'_2 = C'_{2A} \cap C'_{2B} = [0.3 \quad 0.3 \quad 0.3 \quad 0 \quad \cdots \quad 0]$$

按同样的方法可依次求出 C'_3, C'_4, \dots, C'_5 , 并最终求得

$$C' = \bigcup_{i=1}^{56} C'_i = [1 \quad 0.7 \quad 0.3 \quad 0 \quad \cdots \quad 0]$$

对所求得的输出量模糊集合进行清晰化计算(用加权平均法)得

$$z_0 = \text{df}(z) = \frac{1 \times (-6) + 0.7 \times (-5) + 0.3 \times (-4)}{1 + 0.7 + 0.3} = -5.35$$

按照同样的步骤,可以计算出当 x_0, y_0 为其它组合时的输出量 z_0 。最后可列出如表 2.11

所示的实时查询的控制表。作为练习,表中空白处请读者自行补齐。

表 2.11 控制表

$z_0 \backslash y_0 \backslash x_0$	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
-6	-5.35												0
-5													
-4			-5.0								-0.59		
-3													
-2					-3.16				0.18				
-1													
0							0						
1													
2					-0.18				3.16				
3													
4			0.59								5.0		
5													
6	0												5.35

2.7 模糊控制系统的分析和设计

对于什么是模糊控制系统,我们定义凡采用模糊控制器的系统称为模糊控制系统。对于模糊控制器,它不是如常规的控制器的控制那样,采用微分方程,传递函数或状态方程等精确的数学描述,而是通过定义模糊变量、模糊集合及相应的隶属度函数,采用一组模糊条件句来描述输入与输出之间的映射关系。这种用模糊条件句来表示的输入输出关系称为模糊模型,也称语言模型。在模糊控制系统中,若控制对象也用模糊模型表示,则称系统为纯粹的模糊系统;若控制对象采用常规的数学模型来表示,则称系统为混合的模糊系统。

2.7.1 模糊模型表示

由于模糊控制器已经是模糊模型表示,所以这里主要讨论如图 2.22 所示的控制对象的模型表示。

1. 输入输出模型

对于连续的控制对象,其模糊模型可采用如下的模糊条件句来描述。

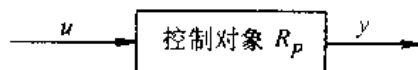


图 2.22 控制对象的模糊模型

R_i : 如果 y 是 A'_0 and...and $y^{(n-1)}$ 是 A'_{n-1} and u 是 B'_0 and...and $u^{(m)}$ 是 B'_m , 则 $y^{(n)}$ 是 C^i , $i = 1, 2, \dots, N$

上述模糊条件句也可写为如下的向量形式

R_i : 如果 \bar{y} 是 A^i and \bar{u} 是 B^i , 则 $y^{(n)}$ 是 C^i , $i=1, 2, \dots, N$

其中

$$\bar{y} = [y \ \dot{y} \ \dots \ y^{(n-1)}]^T, \bar{u} = [u \ \dot{u} \ \dots \ u^{(m)}]^T, A^i = A'_0 \times A'_1 \times \dots \times A'_{n-1} \\ B^i = B'_0 \times B'_1 \times \dots \times B'_m$$

上述模糊模型也可写成

$$R_p = (\bar{y} \times \bar{u}) \rightarrow y^{(n)}$$

以及

$$y^{(n)} = (\bar{y} \times \bar{u}) \circ R_p$$

该模糊模型相当于常规的高阶微分方程模型。

对于离散的控制对象,其模糊模型可表示为

R_i : 如果 $\bar{y}(k)$ 是 A^i and $\bar{u}(k)$ 是 B^i , 则 $y(k+1)$ 是 C^i

其中

$$\bar{y}(k) = [y(k) \ y(k-1) \ \dots \ y(k-n+1)]^T \\ \bar{u}(k) = [u(k) \ u(k-1) \ \dots \ u(k-m)]^T$$

A^i 和 B^i 的意义同上。它也可写成

$$R_p = [\bar{y}(k) \times \bar{u}(k)] \rightarrow y(k+1) \\ y(k+1) = [\bar{y}(k) \times \bar{u}(k)] \circ R_p$$

该模糊模型相当于常规的高阶差分方程模型。

2. 状态空间模型

对于连续的控制对象,其模糊模型可采用如下的模糊条件句来描述。

状态: 如果 x 是 A^i and u 是 B^i , 则 \dot{x} 是 C^i

输出: 如果 x 是 A^i 则 y 是 D^i

其中 $x = [x_1 \ x_2 \ \dots \ x_n]^T$, $u = [u_1 \ u_2 \ \dots \ u_m]^T$, $y = [y_1 \ y_2 \ \dots \ y_r]^T$ 。也可写成如下形式

$$\begin{cases} \dot{x} = (x \times u) \circ R_S \\ y = x \circ R_O \end{cases}$$

其中 $R_S = x \times u \rightarrow \dot{x}$, $R_O = x \rightarrow y$ 。

可见,该模糊模型相当于常规的连续状态空间模型。相应的离散状态空间模型可表示为

$$\begin{cases} x(k+1) = [x(k) \times u(k)] \circ R_S \\ y(k) = x(k) \circ R_O \end{cases}$$

2.7.2 模糊系统分析

1. 稳定性分析

稳定性是控制系统的一个最基本的性能要求,因此如何分析系统的稳定性是控制理论中的一个重要内容。常规的稳定性理论比较成熟,内容相当丰富,而模糊控制系统的稳定性理论尚较缺乏。下面介绍其中的一些内容。

首先分析控制对象的稳定性。若给定控制对象的离散模糊模型为

$$y(k+1) = [\bar{y}(k) \times \bar{u}(k)] \circ R_p$$

其中

$$\bar{y}(k) = [y(k) \ y(k-1) \ \cdots \ y(k-n+1)]^T,$$

$$\bar{u}(k) = [u(k) \ u(k-1) \ \cdots \ u(k-m)]^T$$

上面的模型可以进一步增广为如下的向量形式

$$\bar{y}(k+1) = [\bar{y}(k) \times \bar{u}(k)] \circ \bar{R}_p$$

其中

$$\bar{y}(k+1) = [y(k+1) \ y(k) \ \cdots \ y(k-n+2)]^T$$

$$\mu_{\bar{R}_p}[\bar{y}(k), \bar{u}(k), \bar{y}(k+1)] = \mu_{R_p}[y(k), u(k), y(k+1)]$$

对于上述模糊模型,相应的隶属度函数具有如下关系

$$\begin{aligned} \mu_{y(k+1)} &= \max_{\bar{y}(k), \bar{u}(k)} [(\mu_{\bar{y}(k)} \wedge \mu_{\bar{u}(k)}) \wedge \mu_{\bar{R}_p}(\bar{y}(k), \bar{u}(k), \bar{y}(k+1))] \\ &= \max_{\bar{y}(k), \bar{u}(k)} \min[\mu_{\bar{y}(k)}, \mu_{\bar{u}(k)}, \mu_{\bar{R}_p}(\bar{y}(k), \bar{u}(k), \bar{y}(k+1))] \\ &= \max_{\bar{y}(k)} \min\{\mu_{\bar{y}(k)}, \max_{\bar{u}(k)} \min[\mu_{\bar{u}(k)}, \mu_{\bar{R}_p}(\bar{y}(k), \bar{u}(k), \bar{y}(k+1))]\} \end{aligned}$$

上式表明有如下的关系成立

$$\bar{y}(k+1) = [\bar{y}(k) \times \bar{u}(k)] \circ \bar{R}_p = \bar{y}(k) \circ [\bar{u}(k) \circ \bar{R}_p] = \bar{y}(k) \circ \bar{u}(k) \circ \bar{R}_p$$

为了验证在某一输入情况下的稳定性,令 $\bar{u}(k) = \bar{u}_c$ 为常量,从而上式变为

$$\bar{y}(k+1) = \bar{y}(k) \circ [\bar{u}_c \circ \bar{R}_p]$$

令

$$\bar{R}_p' = \bar{u}_c \circ \bar{R}_p$$

则上式变为

$$\bar{y}(k+1) = \bar{y}(k) \circ \bar{R}_p'$$

若已知系统的初始条件为 $\bar{y}(0)$,则由上式得

$$\bar{y}(1) = \bar{y}(0) \circ \bar{R}_p'$$

$$\bar{y}(2) = \bar{y}(1) \circ \bar{R}_p' = \bar{y}(0) \circ \bar{R}_p' \circ \bar{R}_p' = \bar{y}(0) \circ (\bar{R}_p')^2$$

$$\vdots$$

$$\bar{y}(k) = \bar{y}(0) \circ (\bar{R}_p')^k$$

由此可见,若 $\lim_{k \rightarrow \infty} (\bar{R}_p')^k = \lim_{k \rightarrow \infty} (\bar{u}_c \circ \bar{R}_p)^k = R_\infty'$ 为常量,则可判定系统是稳定的。否则,若当 $k \rightarrow \infty$ 时有 $(\bar{R}_p')^k = (\bar{R}_p')^{k+k_0}$,则可判定系统是振荡的,且振荡周期 $T = k_0 T_0$ 。 T_0 为采样周期。

下面考虑如图 2.23 所示的闭环系统的稳定性,其中模糊控制器的模糊模型为

$$\bar{u}(k) = [\bar{y}(k) \times r(k)] \circ R_c = \bar{y}(k) \circ [r(k) \circ R_c] = \bar{y}(k) \circ r(k) \circ R_c$$

注意它也是增广了的模型,其中

$$\bar{u}(k) = [u(k) \ u(k-1) \ \cdots \ u(k-m)]^T$$

控制对象的模糊模型为

$$\bar{y}(k+1) = [\bar{u}(k) \times \bar{y}(k)] \circ R_p = \bar{u}(k) \circ \bar{y}(k) \circ R_p$$

代入控制器模型可得闭环系统的模糊模型为

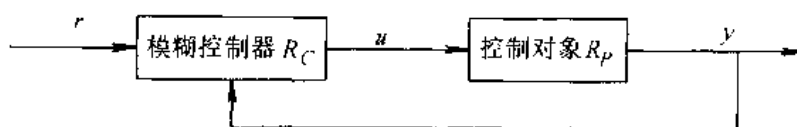


图 2.23 闭环系统的模糊特性

$$\bar{y}(k+1) = \bar{y}(k) \circ r(k) \circ R_C \circ y(k) \circ R_P$$

令

$$C = r(k) \circ R_C$$

则上式变为

$$\bar{y}(k+1) = \bar{y}(k) \circ C \circ \bar{y}(k) \circ R_P$$

为了检验系统在某确定输入下的稳定性,可令 $r(k)$ 为常数,从而 C 也为常数。设系统的初始条件为 $y(0)$,则由上式可依次推得

$$y(1) = \bar{y}(0) \circ C \circ \bar{y}(0) \circ R_P = \bar{y}(0) \circ T_0 \quad T_0 = C \circ \bar{y}(0) \circ R_P$$

$$\begin{aligned} \bar{y}(2) &= \bar{y}(1) \circ C \circ \bar{y}(1) \circ R_P = \bar{y}(0) \circ T_0 \circ C \circ \bar{y}(0) \circ T_0 \circ R_P \\ &= \bar{y}(0) \circ T_1 \quad T_1 = T_0 \circ C \circ \bar{y}(0) \circ T_0 \circ R_P \end{aligned}$$

⋮

$$\begin{aligned} y(k+1) &= \bar{y}(k) \circ C \circ \bar{y}(k) \circ R_P = \bar{y}(0) \circ T_{k-1} \circ C \circ \bar{y}(0) \circ T_{k-1} \circ R_P \\ &= \bar{y}(0) \circ T_k \quad T_k = T_{k-1} \circ C \circ \bar{y}(0) \circ T_{k-1} \circ R_P \end{aligned}$$

由此可见,若 $\lim_{k \rightarrow \infty} T_k = T_\infty$ 为常量,则可判定闭环系统是稳定的,否则,若当 $k \rightarrow \infty$ 时有 $T_k = T_{k+k_0}$,则可判定系统是振荡的,且振荡周期 $T = k_0 T_0$, T_0 为采样周期。

以上所进行的稳定性分析均是针对纯粹的模糊控制系统,即控制对象及控制器均采用模糊模型表示。但在实际问题中经常遇到的是混合的模糊系统,即控制对象采用常规的数学模型,控制器是模糊模型。这时可以有以下两种方法来分析该混合模糊系统的稳定性。

(1) 利用模糊系统辨识的方法将控制对象变换为模糊模型表示。它不仅适用于控制对象解析模型已知的情况,更重要的是它也适用于解析模型未知的情况,这时可根据实验测得的输入输出数据来辨识出控制对象的模糊模型。采用模糊神经网络来对系统模糊辨识是一种较为简单易行的方法。关于模糊神经网络的内容将在下一章中介绍。在求得控制对象的模糊模型后,整个系统变为纯粹的模糊模型,从而可利用上面介绍的方法来进行稳定性的分析。

(2) 将控制器的模糊模型变为确定性的模型,从而混合模糊系统变为常规的控制系统,进而采用常规的方法对系统进行稳定性分析。一个典型的情况是:当模糊控制器经离线计算后得到一张可在线查询的控制表时,这时的模糊控制器可看成为一个多级继电器特性的非线性控制器。这时可按照一般的非线性系统来对系统进行稳定性分析。最常用的方法是采用描述函数法,即求出非线性控制器部分的描述函数。假设控制对象是线性的,则可通过绘制系统的 Nyquist 图来制定系统是否存在自持振荡,以及所产生自持振荡的频率和幅度。

2. 模糊相平面分析

相平面分析法是分析非线性二阶系统的一种直观的图解方法。该方法也可以推广到模糊系统。虽然这种方法只适用于二阶系统,但很大一类系统均可用二阶系统来近似。

设单输入单输出二阶系统的模糊模型用如下的模糊条件句来描述:

R_i : 如果 y 是 A_0^i and \dot{y} 是 A_1^i and u 是 B^i 则 \ddot{y} 是 C^i

$i=1,2,\dots,N$ 。它也可表示为

$$\ddot{y} = (y \times \dot{y} \times u) \circ R_i$$

模糊相平面法是通过图形的方法在相平面上显示每一条件句的影响来说明整个系统的动态特性。例如考虑其中的一个条件句为上面所给出的 R_i ,对于条件句中的每一个模糊集合均定义了相应的隶属度函数,即 $\mu_{A_0^i}(y)$, $\mu_{A_1^i}(\dot{y})$, $\mu_{B^i}(u)$ 及 $\mu_{C^i}(\ddot{y})$ 均为已知。定义该模糊条件句在相平面上的作用中心区域 (y_0, \dot{y}_0) 为

$$(y_0 : \forall_y \mu_{A_0^i}(y) = 1, \dot{y}_0 : \forall_{\dot{y}} \mu_{A_1^i}(\dot{y}) = 1)$$

该模糊条件句的总的影响区域为 $(y : \forall_y \mu_{A_0^i}(y) > 0, \dot{y} : \forall_{\dot{y}} \mu_{A_1^i}(\dot{y}) > 0)$

一般情况下,在点 (y, \dot{y}) 处的相点运动方向角为 $\text{tg}\alpha = \lim_{\Delta y \rightarrow 0} \frac{\Delta \dot{y}}{\Delta y} = \lim_{\Delta t \rightarrow 0} \left(\frac{\Delta \dot{y} / \Delta t}{\Delta y / \Delta t} \right) = \frac{\ddot{y}}{\dot{y}}$ 对于模糊系统,在模糊条件句的作用中心区域的相点运动方向角可以通过清晰化方法求得

$$\text{tg}\alpha = \frac{df(C^i)}{df(A_1^i)}$$

例 2.13 考虑如下的一条模糊条件句:

R_1 : 如果 y 是 NS and \dot{y} 是 PM and u 是 PM 则 \ddot{y} 是 PS 。其中各模糊语言值的隶属度函数如图 2.24 所示。画出对应该模糊条件句的相点运动方向。

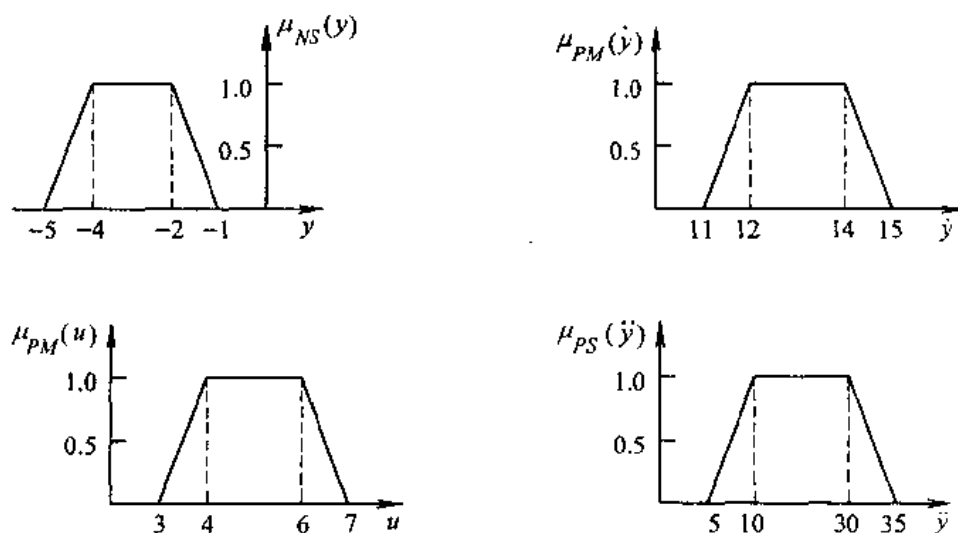


图 2.24 例 2.13 中模糊变量语言值的隶属度函数

根据上面的讨论,可以求得 $\operatorname{tg} \alpha = \frac{df[\mu_{PS}(\dot{y})]}{df[\mu_{PM}(\dot{y})]} = \frac{20}{13} = 1.54$, $\alpha = 57^\circ$

由此并根据所给已知条件可画出该模糊条件句的作用中心区域、总的影响区域以及相点的运动方向如图 2.25 所示。

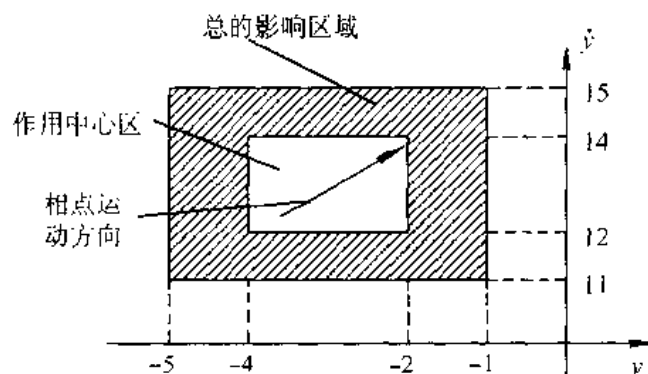


图 2.25 例 2.13 中 R_1 的相平面图

上面的例子只画出了一条模糊规则的相平面图。按照同样的方法可以画出当 u 为常数(在上例中 $u = PM$)时的所有模糊规则的相平面图,如图 2.26 所示,利用该相平面图,可以大致画出对于给定的初始条件的相轨迹。

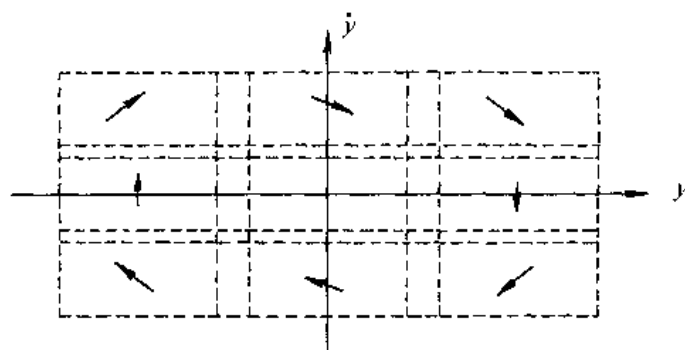


图 2.26 当 u 固定时所有模糊规则的相平面图

图 2.26 所示为当 u 固定为某一常数值时的相平面图,当 u 取不同值时可以画出不同的相平面图,将这些相平面图重叠在一起可以获得如图 2.27 所示的三维相平面图组。

模糊系统的相平面图主要有以下一些用途。

(1) 检验模糊建模的正确性。根据模糊模型可以画出模糊相平面图,根据实测数据可以画出实际的相轨迹图,两者的符合程度可用来检验模糊建模的正确性。

(2) 检验模糊规则的一致性(consistency)、完备性(completeness)以及互相影响(interaction)。模糊规则一致性要求在相平面图上同一区域或非常靠近的区域不存在相点运动方向的不一致;完备性要求在相平面的每个区域至少属于一条规则的影响区域,互相影响是指每条规则的影响区域与邻近规则的影响区域具有一定程度的互相覆盖。

(3) 帮助设计控制规则。三维图可用来确定在不同的状态时应采用怎样的控制才能

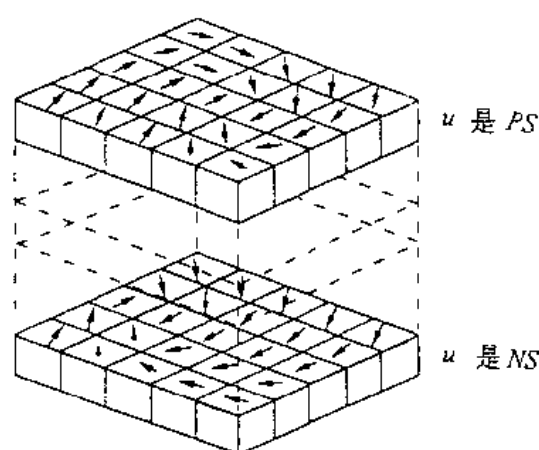


图 2.27 三维相平面图组

获得满意的相轨迹。从而可从图形上直观地确定出模糊控制规则。

(4) 检验系统的稳定性和分析系统的性能。利用相平面图可以大致画出对于给定初始条件的相轨迹,根据该相轨迹可以判断系统的稳定性。若相轨迹终止在一点,说明系统是稳定的;若相轨迹最终形成极限环,说明系统产生自持振荡。若系统稳定,可根据相轨迹确定系统的动态响应性能(过渡过程时间和超调量等)。

(5) 若期望的闭环特性是用语言模型来描述的,则可以通过画出它的相平面图来校核所给闭环特性的正确性。

例 2.14 考虑如下的二阶系统

$$\ddot{y} = (y \times \dot{y} \times u) \circ R_F$$

已知控制量输入 $u_0 = 6.6$, 初始条件 $(y_0, \dot{y}_0) = (-4, -30)$, 各量的语言值及相应的隶属度函数如图 2.28 所示,描述该系统的模糊语言规则如表 2.12 所示,其中“*”处表示不存在相应的规则。该模型可以通过模糊自适应辨识方法获得。用相平面图来验证模糊建模的正确性。

表 2.12 描述系统模型的模糊规则表

$\begin{matrix} \ddot{y} \\ y \end{matrix}$	NB	NM	NS	ZE	PS	PM	PB
PB	*	PB	PB	PB	NS	NM	NB
PM	PB	PB	PB	*	*	*	NB
PS	PB	PB	PB	*	PM	NS	NM
ZE	PB	PB	*	PB	PB	ZE	NM
NS	PB	*	*	*	PB	PS	NM
NM	*	*	*	*	PB	PM	NS
NB	*	*	*	*	*	PB	PM

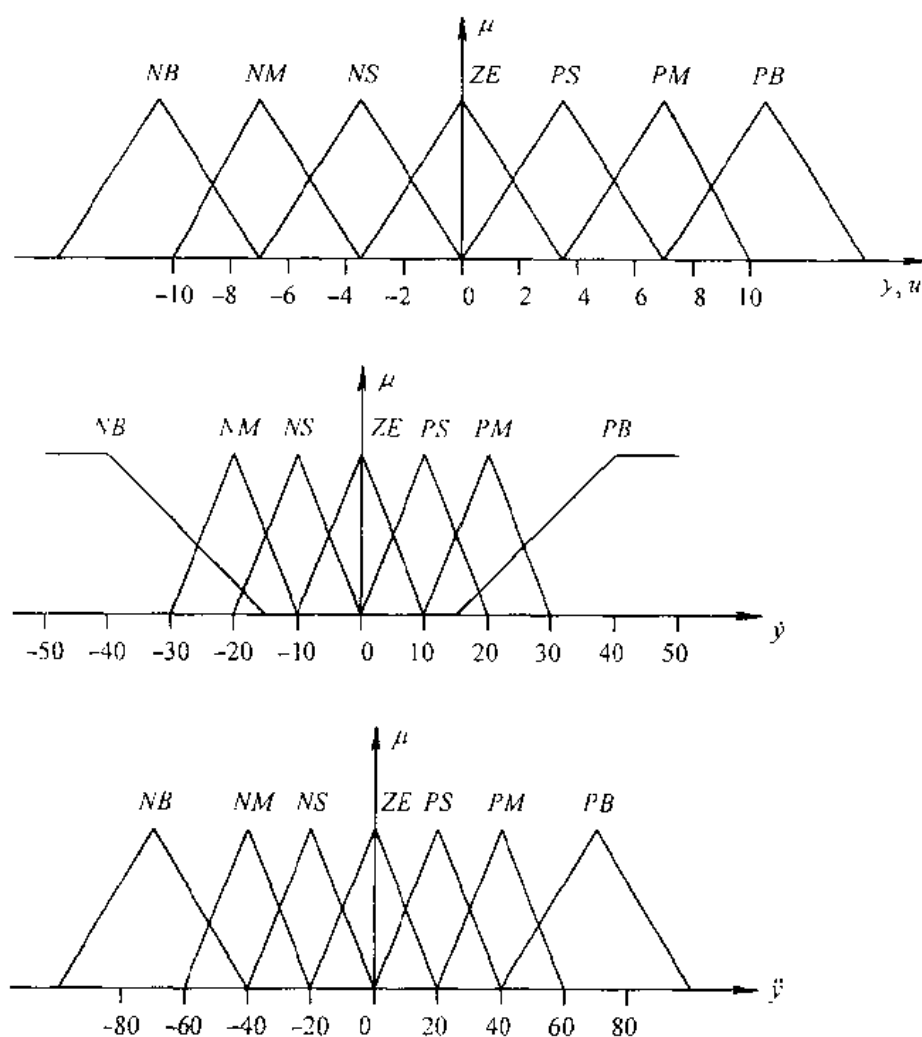


图 2.28 例 2.14 中模糊变量的语言值及相应的隶属度函数

根据给定的模型规则及相应的隶属度函数,可以画出相应的模糊相平面图,如图2.29所示。此时由于隶属度函数均为三角形,因而每一模糊规则的作用中心区域为一个点。图中同时画出了实际的相轨迹图。可见两者有很好的符合程度,从而验证了所辨识的模糊模型的正确性。

2.7.3 模糊系统设计

1. 模糊PID控制

在常规控制中,PID控制是最简单实用的一种控制方法,它既可以依靠数学模型通过解析的方法进行设计,也可不依赖模型而凭借经验和试凑来确定。前面讨论的模糊控制,一般均假设用误差 e 、误差变化 Δe (或误差导数)作为模糊控制的输入量,因而它本质上相当于一种非线性PD控制。为了消除稳态误差,也需加入积分控制,图2.30画出了两种典型的模糊PID控制的结构图,其中图2.30(a)为常规的模糊PID控制,图2.30(b)为增量模糊PID控制。

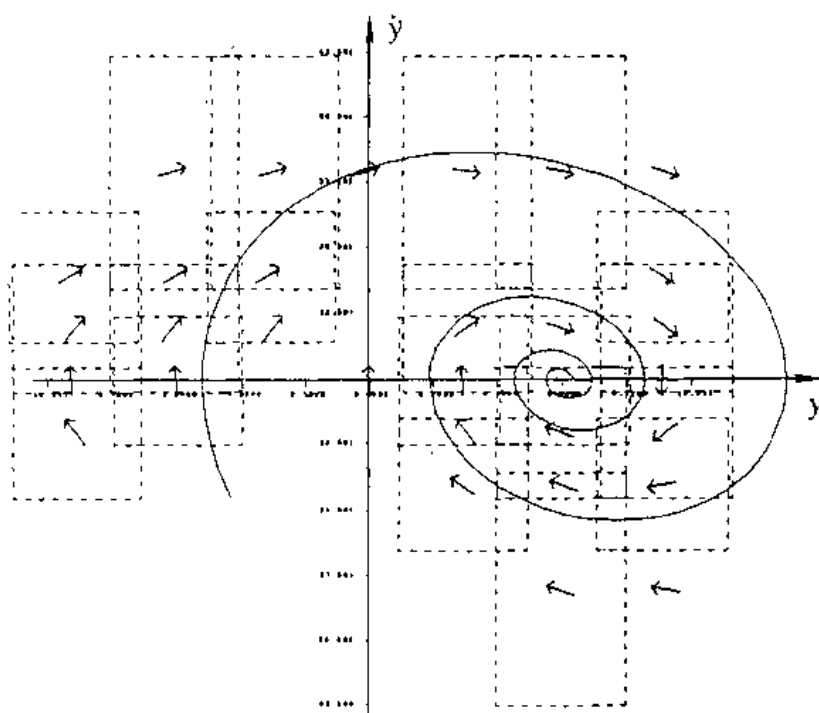


图 2.29 例 2.14 的相平面图与实际系统响应的比较

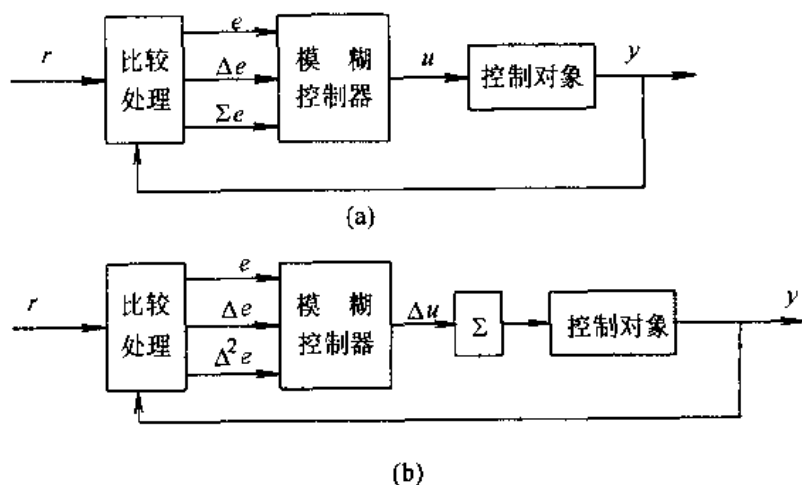


图 2.30 典型的模糊 PID 控制结构

在如图 2.30 所示典型结构中,模糊控制器有三个输入。若每个输入量分 7 个等级,则最多可能需要 $7^3=343$ 条模糊规则,而当输入量为两个时,最多只需要 $7^2=49$ 条模糊规则。可见增加了一个输入量大大增加了模糊控制器设计和计算的复杂性,为此可以考虑采用如图 2.31 所示的变形结构,它同样可以实现模糊 PID 控制的功能。

在图 2.31 的变形结构中,采用两个模糊控制器,其中一个常见的 PD 控制器,它有

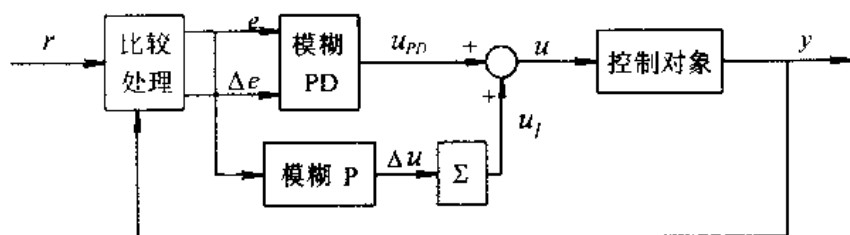


图 2.31 模糊 PID 控制的变形结构

两个输入,最多需 49 条规则(仍假设每个变量分 7 个等级)。另外一个模糊 P 控制,它只有一个输入,最多只需 7 条规则。因此总共最多只需 56 条规则,而图 2.30 的典型结构最多需 343 条规则。可见这种变形结构比通常的模糊 PD 并未增加太大的复杂性,同时也实现了模糊 PID 控制的功能。

2. 基于语言模型求逆的模糊控制器设计

对于如图 2.32 所示的常规控制系统,若已知控制对象模型及期望的闭环系统特性,则可以设计控制器为 $D(s) = \frac{1}{G(s)} = \frac{M(s)}{1 - M(s)}$,即根据期望的闭环特性及开环对象特性的逆即可求得。对于模糊系统,也可依照同样的思路设计模糊控制器,即

$$(\text{开环对象}) \times (\text{期望闭环特性}) \rightarrow (\text{模糊控制器})$$

其中也要求开环对象的逆特性。由于开环特性是用语言模型来描述的,因此这种设计方法需要对语言模型求逆。

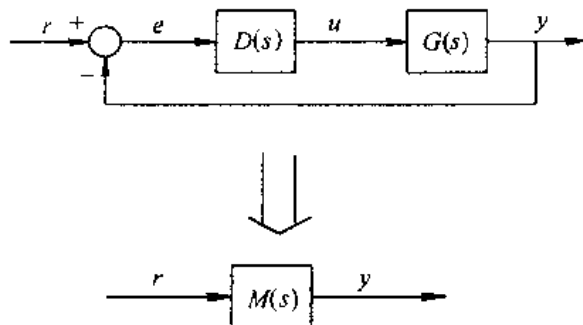


图 2.32 控制系统的常规结构

下面通过对如图 2.33 所示的自动小车侧向模糊控制的设计为例来说明这种方法。图中 s 表示小车关于自由通道中心线的侧向偏移,控制的目标是要求 $(s, \dot{s}) \rightarrow 0$,即要求小车尽量沿着通道的中心线行进。

表 2.13 和表 2.14 分别给出了小车的语言动力学模型和期望的闭环语言动力学模型。图 2.34 给出了各语言变量关于各语言值的隶属度函数。由于小车动力学模型与 s 无关,所以表 2.13 中缺 s 项。

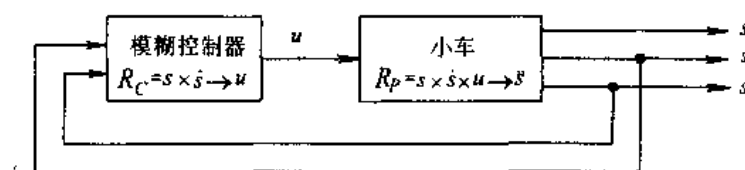
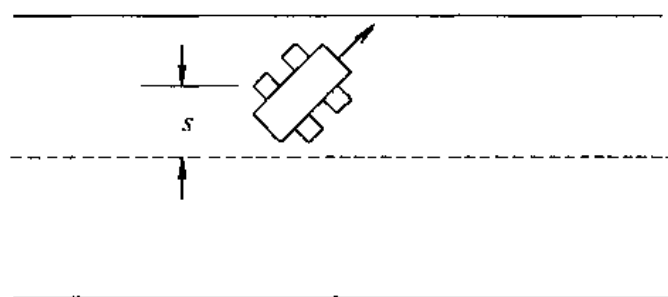


图 2.33 自动小车的侧向控制

表 2.13 描述小车语言动力学模型的规则表

$\begin{matrix} \nearrow s \\ \searrow \dot{s} \end{matrix} \quad u$	NB	NS	ZE	PS	PB
NB	ZE	PS	PB	PB	PB
NS	NS	ZE	PS	PB	PB
ZE	NB	NS	ZE	PS	PB
PS	NB	NB	NS	ZE	PS
PB	NB	NB	NB	NS	ZE

表 2.14 描述期望闭环语言动力学模型的规则表

$\begin{matrix} \nearrow s_d \\ \searrow \dot{s} \end{matrix} \quad s$	NB	NS	ZE	PS	PB
NB	PB	PB	PB	PB	NS
NS	PB	PB	PS	PS	NS
ZE	PB	PS	ZE	NS	NB
PS	PS	NS	NS	NB	NB
PB	PS	NB	NB	NB	NB

根据期望的闭环语言动力学模型可以画出如图 2.35 所示的相平面图。从该相平面图可以看出,描述闭环特性的规则库是完备的、一致的,闭环系统是稳定的,对于任意的初始条件,相轨迹均能收敛到原点。

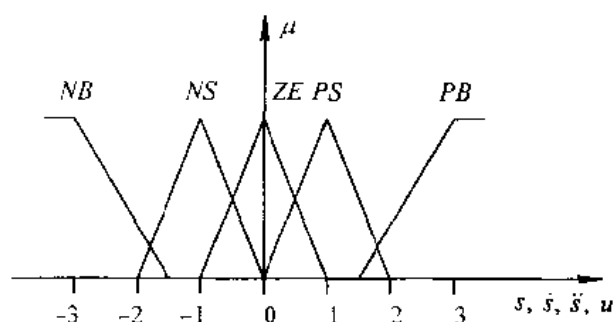


图 2.34 各语言变量值的隶属度函数

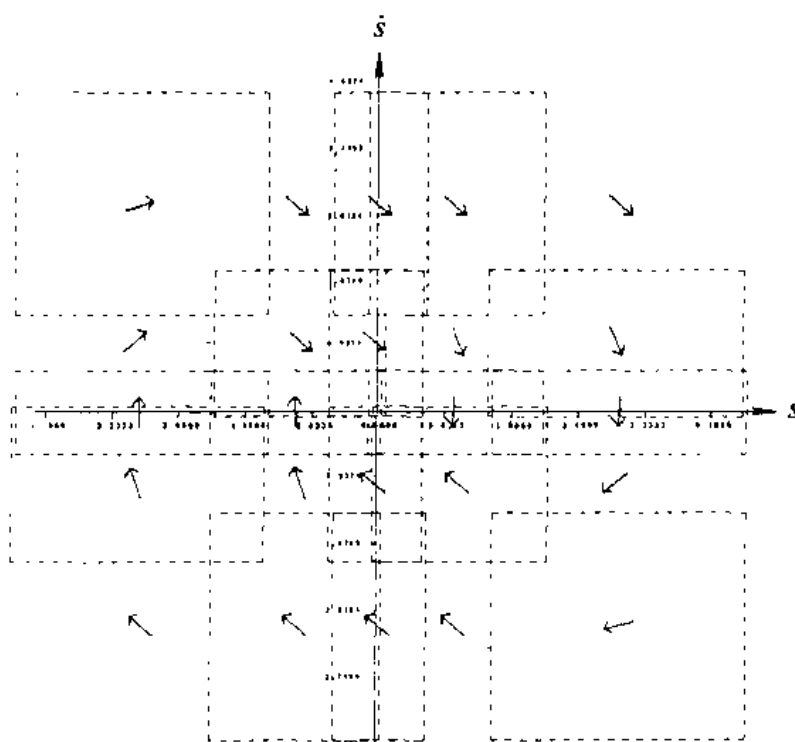


图 2.35 期望闭环特性的相平面图

下面根据小车模型及期望的闭环特性求取控制规则库,即 $s \times \dot{s} \rightarrow u$ 。由于每个变量均分为 5 个模糊等级,因而最多需求取 25 条控制规则,具体步骤如下:

(1) 选定 (s, \dot{s}) 对,在期望的闭环特性规则表中找到相应的 \ddot{s}_d ;

(2) 取 $\ddot{s} = \ddot{s}_d$,在小车模型的规则表中根据 (\dot{s}, \ddot{s}) 求取相应的 u ,这是语言模型求逆的过程,即已知小车模型 $\dot{s} \times u \rightarrow \ddot{s}$ 求取它的逆模型 $\dot{s} \times \ddot{s} \rightarrow u$ 。在进行这一步中可能出现以下三种情况:

- 根据 (\dot{s}, \ddot{s}) 找到一个 u ,这个 u 便是要求的解。
- 根据 (\dot{s}, \ddot{s}) 可以找到多个 u ,这时通常取最小的 u 作为要求的解以尽量减小控制能量;

- 根据 (\dot{s}, \bar{s}) 找不到合适的 u ,这时取最近的解来代替,若有多个最近解,则取其中的最小解。

下面举两个具体例子来说明具体求取的过程

(1) 取 $(s, \dot{s}) = (PS, PB)$,查期望特性规则表得 $\bar{s}_d = NB$ 。取 $\bar{s} = \bar{s}_d = NB$,再由 $(\dot{s}, s) = (PB, NB)$ 查小车模型规则表得 $u = NB/NS/ZE$,这时出现了多解的情况,取最小的解 $u = ZE$ 。

(2) 取 $(s, \dot{s}) = (NB, PB)$,查期望特性规则表得 $\bar{s}_d = PS$,取 $\bar{s} = \bar{s}_d = PS$,再由 $(\dot{s}, \bar{s}) = (PB, PS)$ 查小车模型规则表无解,取最近解 $(\dot{s}, \bar{s}) = (PB, ZE) \rightarrow u = PB$ 来代替,这时相当于将 $\bar{s} = PS$ 用邻近的 ZE 来代替。若 \bar{s} 不变, \dot{s} 用邻近的 PS 代替也可得出同样的 $u (=PB)$ 。

按照上面步骤,最后设计出该系统的模糊控制规则表,如表 2.15 所示。

表 2.15 设计的模糊控制规则表

$\begin{matrix} u \\ \dot{s} \end{matrix} \backslash s$	NB	NS	ZE	PS	PB
NB	ZE	ZE	ZE	ZE	NB
NS	PS	PS	ZE	ZE	NB
ZE	PB	PS	ZE	NS	NB
PS	PB	ZE	ZE	NS	NS
PB	PB	ZE	ZE	ZE	ZE

利用所设计的模糊控制器,对模糊控制系统进行仿真,图 2.36 画出了当初始条件 $(s_0, \dot{s}_0) = (3, 1)$ 时的仿真响应曲线。图 2.37 为相应的相轨迹图,其中图(b)是图(a)的局

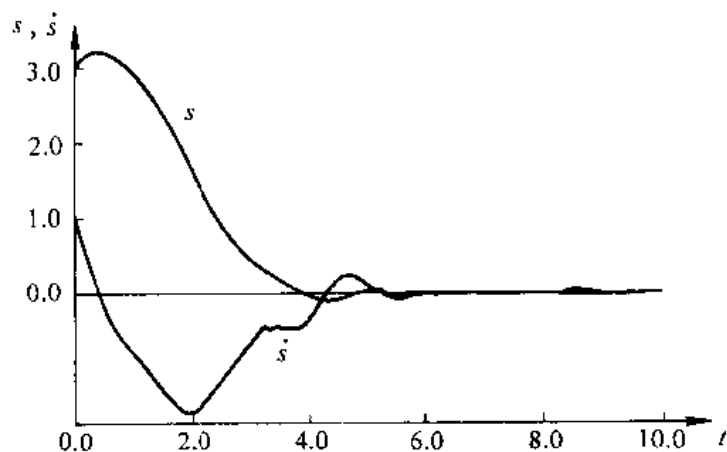


图 2.36 给定初始条件下的系统响应

部放大图。可见所设计的模糊控制器使得系统稳定,且具有满意的性能。同时从相平面图上看到,实际仿真得到的相轨迹与期望的闭环特性相平面图具有良好的符合程度,说明所

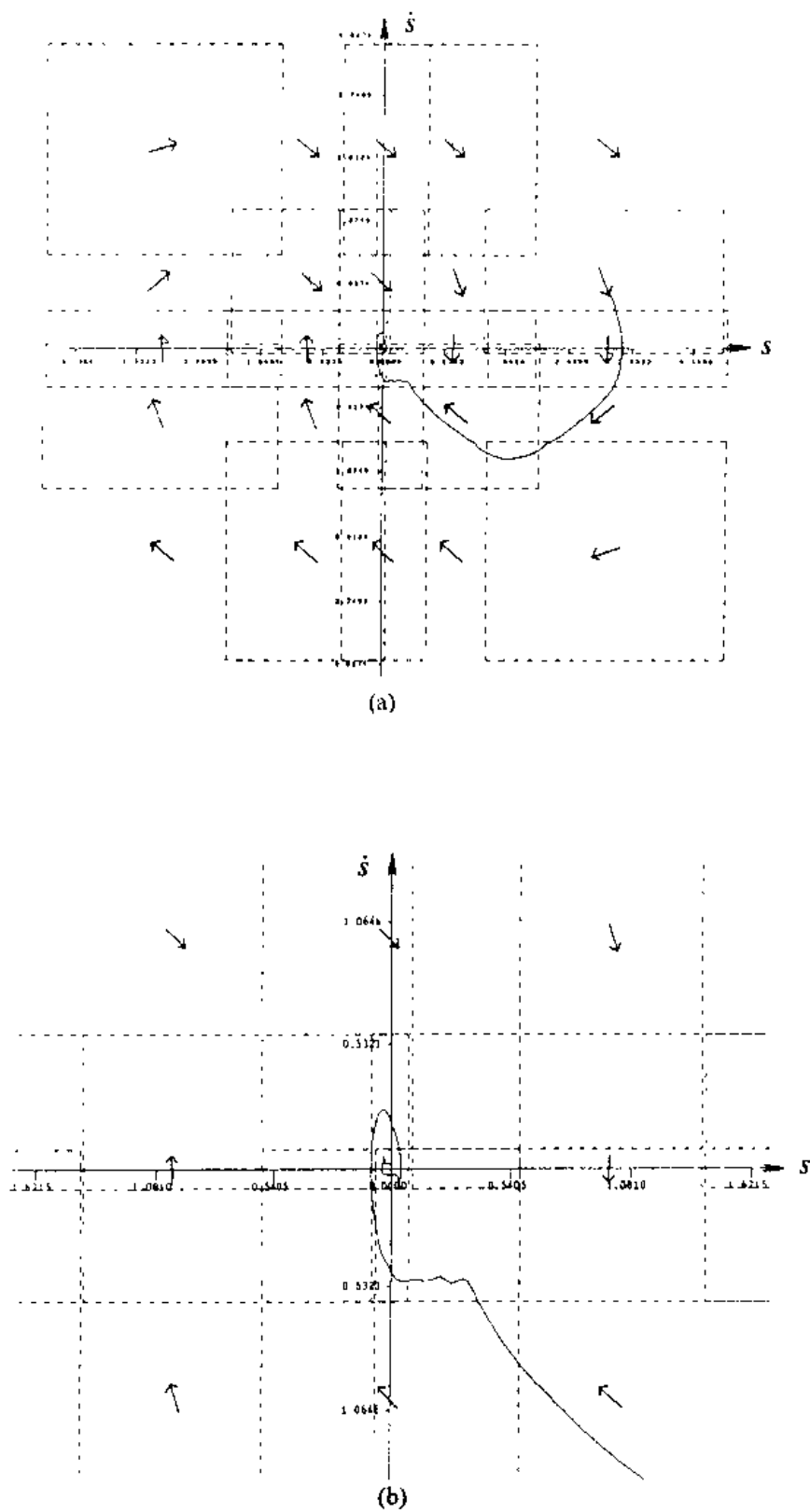


图 2.37 给定初始条件下系统的响应的相轨迹

设计的模糊控制器实现了期望的闭环性能。

2.7.4 基于 Takagi-Sugeno 模型的稳定性分析和设计

1. Takagi-Sugeno 模糊模型

前面关于模糊系统分析和设计的讨论均是基于通常的模糊模型表示。即该模糊模型是用一组模糊蕴含条件句(亦称模糊语言规则)来描述。典型地如“若 x 是 A 则 y 是 B ”, 这些蕴含条件句的后件是用语言值表示的模糊集合。T. Takagi 和 M. Sugeno 提出了另外一种模糊模型的表示方式(以下简称 T-S 模糊模型), 其模糊蕴含条件句形如“若 x 是 A 则 $y=f(x)$ ”, 其中 $f(x)$ 是 x 的线性函数。下面首先介绍这种模糊模型的表示方式。

对于离散系统模型, 其典型的模糊蕴含条件句为

L^i : 若 $y(k)$ 是 A_1^i and \cdots and $y(k-n+1)$ 是 A_n^i and $u(k)$ 是 B_1^i and \cdots and $u(k-m+1)$ 是 B_m^i , 则

$$y^i(k+1) = a_1^i y(k) + \cdots + a_n^i y(k-n+1) + b_1^i u(k) + \cdots + b_m^i u(k-m+1)$$

$L^i (i=1, 2, \cdots, l)$ 表示第 i 条模糊蕴含条件句, 综合 l 条模糊蕴含条件句的输出为

$$y(k+1) = \frac{\sum_{i=1}^l w^i y^i(k+1)}{\sum_{i=1}^l w^i}$$

其中 w^i 是第 i 条模糊语言规则的适用度, 即

$$w^i = \prod_{p=1}^n A_p^i[y(k-p+1)] \times \prod_{q=1}^m B_q^i[u(k-q+1)]$$

这里定义符号 $A(x)$ 为 x 属于 A 的隶属度函数, 即 $A(x) \triangleq \mu_A(x)$

若令

$$\mathbf{x}(k) = \begin{bmatrix} y(k) \\ y(k-1) \\ \vdots \\ y(k-n+1) \end{bmatrix} \quad \mathbf{u}(k) = \begin{bmatrix} u(k) \\ u(k-1) \\ \vdots \\ u(k-m+1) \end{bmatrix}$$

$$A^i = A_1^i \times A_2^i \times \cdots \times A_n^i \quad B^i = B_1^i \times B_2^i \times \cdots \times B_m^i$$

则上面的模糊蕴含条件句可以写成如下的简洁形式

L^i : 若 $\mathbf{x}(k)$ 是 A^i and $\mathbf{u}(k)$ 是 B^i , 则

$$y^i(k+1) = \sum_{p=1}^n a_p^i y(k-p+1) + \sum_{q=1}^m b_q^i u(k-q+1)$$

对于连续系统模型, 其典型的模糊蕴含条件句为

L^i : 若 $y(t)$ 是 A_1^i and \cdots and $y^{(n-1)}(t)$ 是 A_n^i and $u(t)$ 是 B_1^i and \cdots and $u^{(m-1)}(t)$ 是 B_m^i ,

则 $y_i^{(n)}(t) = a_1^i y(t) + \cdots + a_n^i y^{(n-1)}(t) + b_1^i u(t) + \cdots + b_m^i u^{(m-1)}(t)$

由 $L^i (i=1, 2, \cdots, l)$ 所描述的模糊模型的输出为 $y^{(n)}(t) = \frac{\sum_{i=1}^l w^i y_i^{(n)}(t)}{\sum_{i=1}^l w^i}$

w^i 的意义同前。若令

$$\mathbf{x}(t) = \begin{bmatrix} y(t) \\ \dot{y}(t) \\ \vdots \\ y^{(n-1)}(t) \end{bmatrix} \quad \mathbf{u}(t) = \begin{bmatrix} u(t) \\ \dot{u}(t) \\ \vdots \\ u^{(m-1)}(t) \end{bmatrix}$$

则连续系统模糊模型也可写成如下的简洁形式

L^i : 若 $\mathbf{x}(t)$ 是 A^i and $\mathbf{u}(t)$ 是 B^i , 则

$$y_i^{(n)}(t) = \sum_{p=1}^n a_p^i y^{(n-p)}(t) + \sum_{q=1}^m b_q^i u^{(m-q)}(t)$$

2. T-S 模糊模型结构图的化简

以上给出了 T-S 模糊模型的一般表示方式, 它既可以表示控制对象的模型, 也可表示控制器的模型, 同时它也可表示整个闭环系统的模型。若已知闭环系统的 T-S 模型, 则可根据此分析闭环系统的稳定性。但实际上, 人们通过分析或系统辨识的方法所获得的常常只是控制对象的模型, 然后在此基础上设计出控制器, 最后通过结构图的化简才能求得整个闭环系统的模型, 从而根据闭环模型来分析系统的稳定性。

图 2.38(a) 所示为模糊模型表示的系统结构图。如果只是为了分析闭环系统的稳定性, 可令参考输入 r 为常数, 通过坐标平移可进一步假设 $r=0$, 从而图 2.38(a) 可简化为图 2.38(b) (设图(a)比较器前的负号包括在图(b)的模糊控制器中)。下面对图 2.38(b) 的结构进行化简, 以求得整个闭环系统的模糊模型。

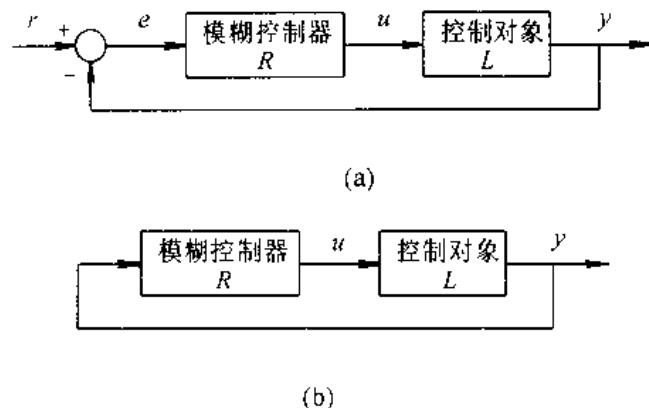


图 2.38 模糊模型表示的系统结构图

定理 2.7.1 如果控制对象和模糊控制器用如下的离散模糊模型表示:

$$L^i: \text{若 } \mathbf{x}(k) \text{ 是 } A^i, \text{ 则} \quad y'(k+1) = \sum_{p=1}^n a_p^i y(k-p+1) + b^i u(k)$$

$$R^j: \text{若 } \mathbf{x}(k) \text{ 是 } C^j, \text{ 则} \quad u'(k) = \sum_{p=1}^m c_p^j y(k-p+1)$$

其中 $m \leq n$, 且

$$x(k) = \begin{bmatrix} y(k) \\ y(k-1) \\ \vdots \\ y(k-n+1) \end{bmatrix} \quad A' = A_1' \times A_2' \times \cdots \times A_n' \quad C' = C_1' \times C_2' \times \cdots \times C_n'$$

则等效的闭环系统可以用如下的离散模糊模型来表示:

S^j : 若 $x(k)$ 是 $(A^i \text{ and } C^j)$, 则 $y^j(k+1) = \sum_{p=1}^n (a_p^i + b^j c_p^j) y(k-p+1)$ 其中 $i=1, 2, \dots, l_1, j=1, 2, \dots, l_2$, 且当 $m < p \leq n$ 时取 $c_p^j = 0$

证明: 根据控制对象的模糊模型可得

$$\begin{aligned} y(k+1) &= \sum_{i=1}^{l_1} w^i y^i(k+1) / \sum_{i=1}^{l_1} w^i \\ &= \sum_{i=1}^{l_1} w^i \left[\sum_{p=1}^n a_p^i y(k-p+1) + b^i u(k) \right] / \sum_{i=1}^{l_1} w^i \end{aligned}$$

其中 w^i 是规则 L^i 的适用度。根据控制器的模糊模型可得

$$\begin{aligned} u(k) &= \sum_{j=1}^{l_2} v^j u^j(k) / \sum_{j=1}^{l_2} v^j \\ &= \sum_{j=1}^{l_2} v^j \sum_{p=1}^m c_p^j y(k-p+1) / \sum_{j=1}^{l_2} v^j \end{aligned}$$

将上式代入前面 $y(k+1)$ 的表达式中得

$$\begin{aligned} y(k+1) &= \sum_{i=1}^{l_1} w^i \left[\sum_{p=1}^n a_p^i y(k-p+1) + b^i \left(\sum_{j=1}^{l_2} v^j \sum_{p=1}^m c_p^j y(k-p+1) / \sum_{j=1}^{l_2} v^j \right) \right] / \sum_{i=1}^{l_1} w^i \\ &= \sum_{i=1}^{l_1} \sum_{j=1}^{l_2} w^i v^j \left[\sum_{p=1}^n (a_p^i + b^i c_p^j) y(k-p+1) \right] / \sum_{i=1}^{l_1} \sum_{j=1}^{l_2} w^i v^j \end{aligned}$$

可见它正好是用 $S^{ij} (i=1, 2, \dots, l_1, j=1, 2, \dots, l_2)$ 所描述的系统的输出, 从而上述定理得证。

从上述定理可知, 等效闭环系统模糊模型 S^j 理论上规则总数应为 $l_1 \times l_2$, 而实际规则总数为 $l_1 \times l_2 - l_3$, 其中 l_3 为 $w^i v^j = 0$ 的个数, 也即使得 $(A_1^i \cap C_1^j = 0 \text{ 或 } \cdots \text{ 或 } A_n^i \cap C_n^j = 0)$ 的个数。

在上面的定理中, 所假定的控制对象和控制器的模型比前面讨论的一般模糊模型结构简单, 对于大多数的实际系统, 采用这种简单模型表示便已足够了。对于一般的模糊模型表示, 如何将其化简为闭环系统的模糊模型, 尚需要进一步深入研究。

例 2.15 已知控制对象的 T-S 模糊模型为

L^1 : 若 $y(k)$ 是 A^1 , 则 $y^1(k+1) = 2.178y(k) - 0.588y(k-1) + 0.603u(k)$

L^2 : 若 $y(k)$ 是 A^2 , 则 $y^2(k+1) = 2.256y(k) - 0.361y(k-1) + 1.120u(k)$

控制器的 T-S 模糊模型为

R^1 : 若 $y(k)$ 是 C^1 , 则 $u^1(k) = -k_1^1 y(k) - k_2^1 y(k-1)$

R^2 : 若 $y(k)$ 是 C^2 , 则 $u^2(k) = -k_1^2 y(k) - k_2^2 y(k-1)$

要求整个闭环系统的模糊模型,进而求当 $A^1=C^1$ 、 $A^2=C^2$ 时的结果。

根据定理 2.7.1 可得闭环系统的模糊模型为

S^{11} :若 $y(k)$ 是 $(A^1 \text{ and } C^1)$, 则

$$y^{11}(k+1) = (2.178 - 0.603k_1^1)y(k) + (-0.588 - 0.603k_2^1)y(k-1)$$

S^{12} :若 $y(k)$ 是 $(A^1 \text{ and } C^2)$, 则

$$y^{12}(k+1) = (2.178 - 0.603k_1^2)y(k) + (-0.588 - 0.603k_2^2)y(k-1)$$

S^{21} :若 $y(k)$ 是 $(A^2 \text{ and } C^1)$, 则

$$y^{21}(k+1) = (2.256 - 1.120k_1^1)y(k) + (-0.361 - 1.120k_2^1)y(k-1)$$

S^{22} :若 $y(k)$ 是 $(A^2 \text{ and } C^2)$, 则

$$y^{22}(k+1) = (2.256 - 1.120k_1^2)y(k) + (-0.361 - 1.120k_2^2)y(k-1)$$

模糊模型的总的输出为

$$y(k+1) = \frac{w^{11}y^{11}(k+1) + w^{12}y^{12}(k+1) + w^{21}y^{21}(k+1) + w^{22}y^{22}(k+1)}{w^{11} + w^{12} + w^{21} + w^{22}}$$

当 $A^1=C^1$ 、 $A^2=C^2$ 时,上面的模糊模型可化简为

S^{11} :若 $y(k)$ 是 $(A^1 \text{ and } A^1)$, 则

$$y^{11}(k+1) = (2.178 - 0.603k_1^1)y(k) + (-0.588 - 0.603k_2^1)y(k-1)$$

$2S^{12*}$:若 $y(k)$ 是 $(A^1 \text{ and } A^2)$, 则

$$\begin{aligned} y^{12*}(k+1) &= \frac{y^{12}(k+1) + y^{21}(k+1)}{2} \\ &= (2.217 - \frac{0.603k_1^1 + 1.120k_1^1}{2})y(k) \\ &\quad + (-0.4745 - \frac{0.603k_2^2 + 1.120k_2^1}{2})y(k-1) \end{aligned}$$

S^{22} :若 $y(k)$ 是 $(A^2 \text{ and } A^2)$, 则

$$y^{22}(k+1) = (2.256 - 1.120k_1^2)y(k) + (-0.361 - 1.120k_2^2)y(k-1)$$

模糊模型总的输出为

$$y(k+1) = \frac{w^{11}y^{11}(k+1) + 2w^{12}y^{12*}(k+1) + w^{22}y^{22}(k+1)}{w^{11} + 2w^{12} + w^{22}}$$

上面关于离散系统的结果可以很容易推广到连续系统。

定理 2.7.2 如果控制对象和模糊控制器用如下的连续模糊模型表示:

$$L^i: \text{若 } x(t) \text{ 是 } A^i, \text{ 则 } y_i^{(n)}(t) = \sum_{p=1}^n a_p^i y^{(p-1)}(t) + b^i u(t)$$

$$R^j: \text{若 } x(t) \text{ 是 } C^j, \text{ 则 } u^j(t) = \sum_{p=1}^m c_p^j y^{(p-1)}(t)$$

其中 $m \leq n$, 且

$$x(t) = \begin{bmatrix} y(t) \\ \dot{y}(t) \\ \vdots \\ y^{(n-1)}(t) \end{bmatrix} \quad A^i = \begin{bmatrix} A_1^i \\ A_2^i \\ \vdots \\ A_n^i \end{bmatrix} \quad C^j = \begin{bmatrix} c_1^j \\ c_2^j \\ \vdots \\ c_m^j \end{bmatrix}$$

则等效的闭环系统可以用如下的连续模糊模型来表示:

S^y : 若 $x(t)$ 是 $(A^i \text{ and } C^j)$, 则

$$y_{ij}^{(n)}(t) = \sum_{p=1}^n (a_p^i + b^j c_p^j) y^{(j-1)}(t)$$

其中 $i=1, 2, \dots, l_1$, $j=1, 2, \dots, l_2$, 且当 $m < p \leq n$ 时取 $c_p^j = 0$

按照证明定理 2.7.1 的同样步骤可以证得该定理, 此处略。

3. 基于 T-S 模糊模型的稳定性分析

该系统可用如下的离散模糊模型来表示:

$$L^i: \text{若 } x(k) \text{ 是 } A^i, \text{ 则 } y^i(k+1) = \sum_{p=1}^n a_p^i y(k-p+1)$$

其中 $i=1, 2, \dots, l$, 该模糊模型也可进一步表示成如下的矩阵形式:

$$L^i: \text{若 } x(k) \text{ 是 } A^i, \text{ 则 } x^i(k+1) = A_i x(k)$$

其中

$$x(k) = \begin{bmatrix} y(k) \\ y(k-1) \\ \vdots \\ y(k-n+1) \end{bmatrix} \quad A_i = \begin{bmatrix} a_1^i & a_2^i & \cdots & a_{n-1}^i & a_n^i \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}$$

模糊系统的输出为

$$x(k+1) = \sum_{i=1}^l w^i A_i x(k) / \sum_{i=1}^l w^i$$

这里的模型表示既适用于开环系统, 也适用于闭环系统, 由于主要利用该模型来判断系统的稳定性, 所以取其中的外部输入为零。

类似地, 系统的连续模糊模型可以表示为

$$L^i: \text{若 } x(t) \text{ 是 } A^i, \text{ 则 } \dot{x}^i(t) = A_i x(t)$$

其中

$$x(t) = \begin{bmatrix} y(t) \\ \dot{y}(t) \\ \vdots \\ y^{(n-1)}(t) \end{bmatrix} \quad A_i = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ a_1^i & a_2^i & a_3^i & \cdots & a_n^i \end{bmatrix}$$

模糊系统的输出为

$$\dot{x}(t) = \sum_{i=1}^l w^i A_i x(t) / \sum_{i=1}^l w^i$$

下面给出判断系统稳定性的一个充分条件。

定理 2.7.3 对于上面描述的离散模糊模型, 如果存在一个共同的正定矩阵 P , 对于所有的子系统均有 $A_i^T P A_i - P < 0$ ($i=1, 2, \dots, l$)。则所论模糊系统的平衡状态是全局渐近稳定的。

证明:考虑如下的标量函数 $V[x(k)]$

$$V[x(k)] = x^T(k)Px(k)$$

它满足

$$(a) V(0) = 0$$

$$(b) \text{ 当 } x(k) \neq 0 \text{ 时, } V[x(k)] > 0$$

$$(c) \text{ 当 } \|x(k)\| \rightarrow \infty \text{ 时, } V[x(k)] \rightarrow \infty$$

进而我们求

$$\begin{aligned} \Delta V[x(k)] &= V[x(k+1)] - V[x(k)] = x^T(k+1)Px(k+1) - x^T(k)Px(k) \\ &= \left(\sum_{i=1}^l w^i A_i x(k) / \sum_{i=1}^l w^i \right)^T P \left(\sum_{j=1}^l w^j A_j x(k) / \sum_{j=1}^l w^j \right) - x^T(k)Px(k) \\ &= x^T(k) \left[\left(\sum_{i=1}^l w^i A_i^T / \sum_{i=1}^l w^i \right) P \left(\sum_{j=1}^l w^j A_j / \sum_{j=1}^l w^j \right) - P \right] x(k) \\ &= \sum_{i,j=1}^l w^i w^j x^T(k) (A_i^T P A_j - P) x(k) / \sum_{i,j=1}^l w^i w^j \\ &= \left[\sum_{i=1}^l (w^i)^2 x^T(k) (A_i^T P A_i - P) x(k) \right. \\ &\quad \left. + \sum_{i < j} w^i w^j x^T(k) (A_i^T P A_j + A_j^T P A_i - 2P) x(k) \right] / \sum_{i,j=1}^l w^i w^j \end{aligned}$$

由于 w^i 为第 i 条规则的适用度, 所以 $w^i \geq 0$ ($i=1, 2, \dots, l$), 且有 $\sum_{i=1}^l w^i > 0$ 。也即上式中分母大于零。根据定理已知条件, 上式中分子第一项小于零, 下面来考察分子中的第二项。
 $A_i^T P A_j + A_j^T P A_i - 2P = -(A_i - A_j)^T P (A_i - A_j) + A_i^T P A_i + A_j^T P A_j - 2P$

$$= [-(A_i - A_j)^T P (A_i - A_j)] + [A_i^T P A_i - P] + [A_j^T P A_j - P]$$

由于假设 P 是正定矩阵, 所以上式第一项小于等于零。根据定理条件, 上式第二和第三项均小于零, 所以总的式子小于零, 也即前面 $\Delta V[x(k)]$ 式子中的分子第二项也小于零。最后推得 $\Delta V[x(k)] < 0$ 。可见 $V[x(k)]$ 是一个李雅普诺夫函数。根据李雅普诺夫稳定性理论, 该模糊系统是全局渐近稳定的, 从而定理得证。

上述定理当 $l=1$ 时便退化为通常的非模糊线性离散系统的稳定性判据。该定理也可用于能够用分段线性函数来逼近的非线性系统的稳定性分析。这时模糊条件句中前件的模糊集合可以用普通集合来代替, 从而 w^i 的取值为 0 或 1。由于许多非线性系统可以分段线性近似, 因而该定理不仅可用于模糊系统, 而且也可用于普通的非线性系统的稳定性分析。

上述定理只是判断稳定性的充分条件。当找不到共同的 P 满足上述定理条件时, 并不能由此定理来判定所设系统是否稳定。还有一点值得注意: 若每个子系统均稳定, 即对每个子系统均能找到正定矩阵 P_i 使 $A_i^T P_i A_i - P_i < 0$, 并不能保证整个系统一定稳定。而必须找到一个共同的 P , 使得 $A_i^T P A_i - P < 0$ ($i=1, 2, \dots, l$) 才能保证整个系统稳定。

定理 2.7.4 对于上面描述的连续模糊模型, 如果存在一个共同的正定矩阵 P , 对于所有的子系统均有 $A_i^T P + P A_i < 0$ ($i=1, 2, \dots, l$)。则所论模糊系统的平衡状态是全局渐

近稳定的。

证明:考虑如下的标量函数 $V[\mathbf{x}(t)]$

$$V[\mathbf{x}(t)] = \mathbf{x}^T(t)P\mathbf{x}(t)$$

它满足

$$(a) V(0) = 0$$

$$(b) \text{ 当 } \mathbf{x}(t) \neq 0 \text{ 时, } V[\mathbf{x}(t)] > 0$$

$$(c) \text{ 当 } \|\mathbf{x}(t)\| \rightarrow \infty \text{ 时, } V[\mathbf{x}(t)] \rightarrow \infty$$

进而可以求得

$$\begin{aligned} \dot{V}[\mathbf{x}(t)] &= \dot{\mathbf{x}}^T(t)P\mathbf{x}(t) + \mathbf{x}^T(t)P\dot{\mathbf{x}}(t) \\ &= \left[\sum_{i=1}^l w^i A_i \mathbf{x}(t) \right]^T P \mathbf{x}(t) + \mathbf{x}^T(t) P \left[\sum_{i=1}^l w^i A_i \mathbf{x}(t) \right] \\ &= \left[\sum_{i=1}^l w^i \mathbf{x}^T(t) (A_i^T P + P A_i) \mathbf{x}(t) \right] / \sum_{i=1}^l w^i \end{aligned}$$

由于 $\sum_{i=1}^l w^i > 0$, 且根据定理条件对所有 i 有: $A_i^T P + P A_i < 0$, 从而显然 $\dot{V}[\mathbf{x}(t)] < 0$ 。

可见 $V[\mathbf{x}(t)]$ 是一个李雅普诺夫函数, 根据李雅普诺夫稳定性理论, 该模糊系统是全局渐近稳定的, 从而定理得证。

定理 2.7.3 和定理 2.7.4 给出了判断模糊系统稳定性的充分条件, 关键是要找到共同的正定矩阵 P 。下面给出关于 P 存在的必要条件。

定理 2.7.5 对于离散模糊模型, 如果 $A_i (i=1, 2, \dots, l)$ 是稳定且非奇异矩阵, 若存在共同的正定矩阵 P 使得对所有 i 均有 $A_i^T P A_i - P < 0$, 则对于任意 $i, j=1, 2, \dots, l$, $A_i A_j$ 一定为稳定矩阵。这里稳定矩阵是指其特征值均在单位圆内。

证明: 根据定理条件有: $A_i^T P A_i < P$, 即 $P < (A_i^{-1})^T P A_i^{-1}$ 对任意 i 均成立。进而根据定理条件有 $A_i^T P A_i < P < (A_i^{-1})^T P A_i^{-1}$, 即有 $A_i^T A_j^T P A_i A_j - P < 0$, 它说明 $A_i A_j$ 为稳定矩阵对任意 $i, j=1, 2, \dots, l$ 均成立。从而定理得证。

定理 2.7.6 对于连续模糊模型, 如果 $A_i (i=1, 2, \dots, l)$ 是稳定且非奇异矩阵, 若存在一个共同的正定矩阵 P 使对所有 i 均有 $A_i^T P + P A_i < 0$, 则对于任意 $i, j=1, 2, \dots, l$, $A_i + A_j$ 一定为稳定矩阵。这里稳定矩阵是指其特征值均在左半平面。

证明: 根据定理条件, 对所有 i 和 j , 有

$$A_i^T P + P A_i < 0$$

$$A_j^T P + P A_j < 0$$

将以上两式相加得

$$(A_i + A_j)^T P + P(A_i + A_j) < 0$$

即 $A_i + A_j$ 为稳定矩阵对任意 $i, j=1, 2, \dots, l$ 均成立。从而定理得证。

定理 2.7.5 和 2.7.6 给出了是否存在共同正定矩阵 P 的必要条件, 对于一个具体问题, 若能找到一个 $A_i A_j$ (离散) 或 $A_i + A_j$ (连续) 是不稳定的, 则说明一定不存在共同的正定矩阵 P , 因而可不必做徒劳无益的工作来继续寻找 P 。

例如, 已知离散模型的两个子系统矩阵为

$$A_1 = \begin{bmatrix} 1 & -0.5 \\ 1 & 0 \end{bmatrix} \quad A_2 = \begin{bmatrix} -1 & -0.5 \\ 1 & 0 \end{bmatrix}$$

不难验证 A_1 和 A_2 均是稳定的。但对于 $A_1 A_2 = \begin{bmatrix} -1.5 & -0.5 \\ -1.0 & -0.5 \end{bmatrix}$ 可以求得其特征值为 -0.135 和 -1.865 , 因而它是不稳定的, 从而根据上面的定理可以断定不存在共同的正定矩阵 P , 使得 $A_i^T P A_i - P < 0$ ($i=1, 2$)。

为了判断模糊系统的稳定性, 需要寻找共同的正定矩阵 P 。目前尚无非常有效的方法来寻找 P 。一个通常的步骤是(以离散模型为例)。

(1) 对 $i=1, 2, \dots, l$, 求正定矩阵 P_i 以使得 $A_i^T P_i A_i - P_i < 0$ 。具体做法是, 设定一正定矩阵 Q , 求解离散李雅普诺夫方程 $P_i = A_i^T P_i A_i + Q$, 若 A_i 是稳定的, 即可求得正定矩阵 P_i 。

(2) 检验是否存在 $P_j \in (P_i | i=1, 2, \dots, l)$, 以使得对所有 i 有 $A_i^T P_j A_i - P_j < 0$, 若存在, 则 $P = P_j$; 若不存在, 则返回(1), 重设 Q , 重复上述步骤。

4. 基于 T-S 模型的模糊控制器设计

利用 T-S 模型表示以及基于该模型的稳定性分析方法, 可以帮助进行模糊控制器的设计。假设控制对象的 T-S 模糊模型是已知的, 同时假设已给定模糊控制器的 T-S 模糊模型的结构。设计问题变为根据闭环系统性能要求来确定模糊控制器中的参数。设计过程可以分为以下 3 个步骤。

(1) 进行模糊结构图的化简以获得闭环系统的模糊模型。为便于说明, 设控制对象的模糊模型为

L^1 : 若 $y(k)$ 是 A^1 则 $y^1(k+1) = a^1 y(k) + b^1 u(k)$

L^2 : 若 $y(k)$ 是 A^2 则 $y^2(k+1) = a^2 y(k) + b^2 u(k)$

设模糊控制器的模糊模型为

R^1 : 若 $y(k)$ 是 A^1 则 $u^1(k) = -f^1 y(k)$

R^2 : 若 $y(k)$ 是 A^2 则 $u^2(k) = -f^2 y(k)$

根据模糊结构图的化简方法可以求总的闭环系统的模糊模型为

S^{11} : 若 $y(k)$ 是 $(A^1 \text{ and } A^1)$, 则 $y^{11}(k+1) = (a^1 - b^1 f^1) y(k)$

$2S^{12*}$: 若 $y(k)$ 是 $(A^1 \text{ and } A^2)$, 则

$$y^{12*}(k+1) = \left\{ \frac{a^1 + a^2 - b^1 f^2 - b^2 f^1}{2} \right\} y(k)$$

S^{22} : 若 $y(k)$ 是 $(A^2 \text{ and } A^2)$, 则 $y^{22}(k+1) = (a^2 - b^2 f^2) y(k)$

(2) 确定模糊控制器的参数 f^1 和 f^2 , 以使每个子系统稳定。这一步可以采用通常的线性系统理论来设计。

(3) 寻找共同的正定矩阵来检验整个模糊系统的稳定性, 若找不到共同的 P , 则返回步骤(2), 直到找到共同的正定矩阵 P , 以确得整个模糊系统的稳定性。由于每个子系统的稳定性并不能确保整个系统稳定, 因而这第(3)步是十分必要的。

例 2.16 已知控制对象及控制器模糊模型同例 2.15, 且 $A^1 = C^1$, $A^2 = C^2$, 即

L^1 : 若 $y(k)$ 是 A^1 , 则 $y^1(k+1) = 2.178y(k) - 0.588y(k-1) + 0.603u(k)$

L^2 : 若 $y(k)$ 是 A^2 , 则 $y^2(k+1) = 2.256y(k) - 0.361y(k-1) + 1.120u(k)$

R^1 : 若 $y(k)$ 是 A^1 , 则 $u^1(k) = -k_1^1 y(k) - k_2^1 y(k-1)$

R^2 : 若 $y(k)$ 是 A^2 , 则 $u^2(k) = -k_1^2 y(k) - k_2^2 y(k-1)$

要求设计模糊控制器的参数 k_1^1, k_2^1, k_1^2 和 k_2^2 。其中 A^1 和 A^2 是如图 2.39 所示的模糊集合。

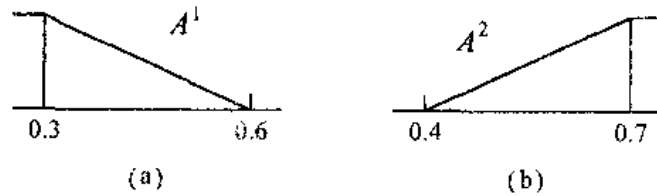


图 2.39 例 2.16 中的 A^1 和 A^2

(1) 进行模糊结构图的化简, 获得闭环系统的模糊模型。

S^{11} : 若 $y(k)$ 是 $(A^1 \text{ and } A^1)$, 则

$$y^{11}(k+1) = (2.178 - 0.603k_1^1)y(k) + (-0.558 - 0.603k_2^1)y(k-1)$$

$2S^{12*}$: 若 $y(k)$ 是 $(A^1 \text{ and } A^2)$, 则

$$y^{12*}(k+1) = (2.217 - \frac{0.603k_1^1 + 1.120k_1^2}{2})y(k) + (-0.4745 - \frac{0.603k_2^1 + 1.120k_2^2}{2})y(k-1)$$

S^{22} : 若 $y(k)$ 是 $(A^2 \text{ and } A^2)$, 则

$$y^{22}(k+1) = (2.256 - 1.120k_1^2)y(k) + (-0.361 - 1.120k_2^2)y(k-1)$$

(2) 按照每个子系统分别设计模糊控制器的参数。 S^{11} 中有两个参数: k_1^1 和 k_2^1 ; S^{22} 中有两个参数: k_1^2 和 k_2^2 ; $2S^{12*}$ 中有 4 个参数: k_1^1, k_2^1, k_1^2 和 k_2^2 。例如可按照二阶系统来设计, 取阻尼系数 ζ 在 0.707 附近, 或者可按照极点配置的方法来进行设计。

根据 S^{11} 来确定 k_1^1 和 k_2^1 , 根据 S^{22} 来确定 k_1^2 和 k_2^2 。这些参数的确定也不是唯一的, 下面取其中两组:

$$(A) \quad k_1^1 = 2.46, k_2^1 = -0.16, k_1^2 = 0.912, k_2^2 = 0.079$$

$$(B) \quad k_1^1 = 3.00, k_2^1 = -0.42, k_1^2 = 1.205, k_2^2 = -0.053$$

对于所选出的这两组参数, 还必须校核是否对 $2S^{12*}$ 子系统也合适。经校核对 (A) 组参数, $2S^{12*}$ 子系统的 $\zeta = 0.705$; 对 (B) 组参数, 相应的 $\zeta = 0.764$, 它们都在指标要求的范围。至此可见, 所有的子系统均是稳定的。

(3) 检验整个模糊系统的稳定性, 对于 (A) 组参数, 可求得各子系统的参数矩阵为

$$A_{11} = \begin{bmatrix} 1.235 & -0.4495 \\ 1 & 0 \end{bmatrix} \quad A_{12*} = \begin{bmatrix} 0.564 & -0.3975 \\ 1 & 0 \end{bmatrix} \quad A_{22} = \begin{bmatrix} 1.235 & -0.4195 \\ 1 & 0 \end{bmatrix}$$

按照前面给出的步骤可以求得一个共同的正定矩阵 P

$$P = \begin{bmatrix} 9.13 & -3.50 \\ -3.50 & 2.85 \end{bmatrix}$$

使得 $A_{11}^T P A_{11} - P < 0$, $A_{22}^T P A_{22} - P < 0$, $A_{12*}^T P A_{12*} - P < 0$, 从而说明 (A) 组参数使得整

个系统是稳定的。对于(B)组参数,可以求得

$$A_{11} = \begin{bmatrix} 0.906 & -0.302 \\ 1 & 0 \end{bmatrix} \quad A_{12} = \begin{bmatrix} 0.174 & -0.223 \\ 1 & 0 \end{bmatrix} \quad A_{22} = \begin{bmatrix} 0.906 & -0.302 \\ 1 & 0 \end{bmatrix}$$

对于该组参数。最后也可找到一个共同的正定矩阵 P

$$P = \begin{bmatrix} 4.19 & -0.88 \\ -0.88 & 1.38 \end{bmatrix}$$

使得 $A_{11}^T P A_{11} - P < 0$, $A_{22}^T P A_{22} - P < 0$, $A_{12}^T P A_{12} - P < 0$, 从而说明(B)组也能使得整个系统是稳定的。

分别对(A)组和(B)组参数对系统进行仿真,结果如图 2.40 所示。由图可见,(B)组参数具有比(A)组参数更快的动态响应。

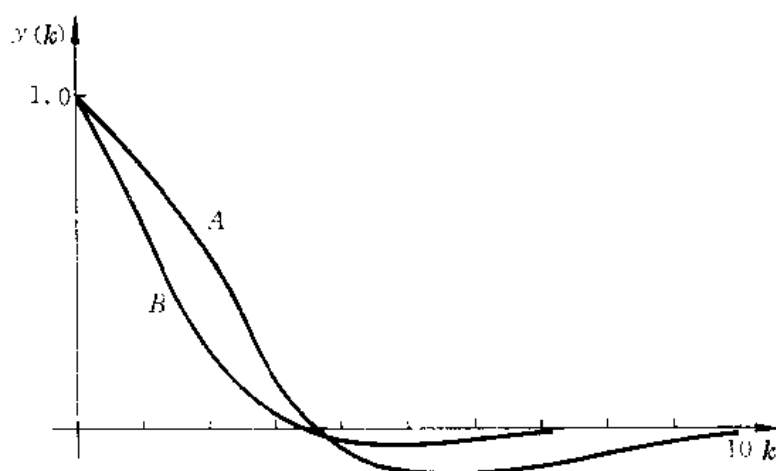


图 2.40 例 2.16 系统的响应

2.7.5 基于模糊状态方程模型的系统分析和设计

1. 模糊状态方程模型

上一节所介绍的 T-S 模型,对于离散系统可看成是模糊差分方程模型;对连续系统可以看成是模糊微分方程模型。因此自然可以想到,也可以建立模糊状态方程模型,因而它实质上是 T-S 模型的推广。

对于离散控制对象,其模糊状态方程模型可以表示为

R_p : 若 $x_1(k)$ 是 M_1^i and \cdots and $x_n(k)$ 是 M_n^i , 则

$$\begin{cases} x(k+1) = F_i x(k) + G_i u(k) \\ y(k) = C_i x(k) + D_i u(k) \end{cases}$$

$i=1,2,\cdots,l$, R_p 表示控制对象的第 i 条模糊规则。其中 $M_j^i (j=1,2,\cdots,n)$ 是模糊集合, l 是规则个数。 $x(k)$ 是状态向量, $u(k)$ 是控制向量, $y(k)$ 是输出向量。

$$\mathbf{x}(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_n(k) \end{bmatrix} \quad \mathbf{u}(k) = \begin{bmatrix} u_1(k) \\ u_2(k) \\ \vdots \\ u_m(k) \end{bmatrix} \quad \mathbf{y}(k) = \begin{bmatrix} y_1(k) \\ y_2(k) \\ \vdots \\ y_r(k) \end{bmatrix}$$

$$F_i \in R^{n \times n} \quad G_i \in R^{n \times m} \quad C_i \in R^{r \times n} \quad D_i \in R^{r \times m}$$

若定义 $M^i = M_1^i \times \cdots \times M_n^i$, 即 M^i 表示 $M_1^i, M_2^i, \dots, M_n^i$ 的直积模糊集合, 则上述的模糊模型可以表示为如下简洁的形式:

R_p^i : 若 $\mathbf{x}(k)$ 是 M^i , 则

$$\begin{cases} \mathbf{x}(k+1) = F_i \mathbf{x}(k) + G_i \mathbf{u}(k) \\ \mathbf{y}(k) = C_i \mathbf{x}(k) + D_i \mathbf{u}(k) \end{cases}$$

若设 $M_j^i(\mathbf{x})$ 表示 \mathbf{x} 属于 M_j^i 的隶属度函数, 若直积运算采用求积法, 则

$$M^i(\mathbf{x}) = \prod_{j=1}^n M_j^i(\mathbf{x})$$

$M^i(\mathbf{x})$ 表示 \mathbf{x} 属于 M^i 的隶属度函数, 同时它也表示第 i 条规则的适用度。

若模糊化采用单点模糊集合, 清晰化采用加权平均法, 则可得整个系统状态方程为

$$\begin{aligned} \mathbf{x}(k+1) &= \sum_{i=1}^l M^i(\mathbf{x}) [F_i \mathbf{x}(k) + G_i \mathbf{u}(k)] / \sum_{i=1}^l M^i(\mathbf{x}) = \sum_{i=1}^l \mu_i(\mathbf{x}) [F_i \mathbf{x}(k) + G_i \mathbf{u}(k)] \\ &= F \mathbf{x}(k) + G \mathbf{u}(k) \end{aligned}$$

$$\begin{aligned} \mathbf{y}(k) &= \sum_{i=1}^l M^i(\mathbf{x}) [C_i \mathbf{x}(k) + D_i \mathbf{u}(k)] / \sum_{i=1}^l M^i(\mathbf{x}) = \sum_{i=1}^l \mu_i(\mathbf{x}) [C_i \mathbf{x}(k) + D_i \mathbf{u}(k)] \\ &= C \mathbf{x}(k) + D \mathbf{u}(k) \end{aligned}$$

其中

$$\begin{aligned} \mu_i(\mathbf{x}) &= M^i(\mathbf{x}) / \sum_{j=1}^l M^j(\mathbf{x}) \quad F = \sum_{i=1}^l \mu_i(\mathbf{x}) F_i \quad G = \sum_{i=1}^l \mu_i(\mathbf{x}) G_i \\ C &= \sum_{i=1}^l \mu_i(\mathbf{x}) C_i \quad D = \sum_{i=1}^l \mu_i(\mathbf{x}) D_i \end{aligned}$$

并假设

$$M^i(\mathbf{x}) \geq 0 \quad \sum_{j=1}^l M^j(\mathbf{x}) > 0$$

因此有

$$0 \leq \mu_i(\mathbf{x}) \leq 1 \quad \sum_{i=1}^l \mu_i(\mathbf{x}) = 1$$

$\mu_i(\mathbf{x})$ 表示第 i 条规则的归一化后的适用度。

上述模糊状态空间模型可作如下的物理解释: 将整个 n 维状态空间分为 l 个模糊子空间集合 $M^i (i=1, 2, \dots, l)$ 。对于每个模糊子空间, 系统的动力学特性可用一个局部线性状态方程来描述。整个系统动力学特性则是这些局部线性模型的加权和。换句话说, 该模糊建模方法的本质在于: 一个整体非线性的动力学模型可以看成是许多个局部线性模型的模糊逼近。

从上面的系统描述可以看出,整个系统的状态方程形式上似乎仍是线性模型,但其系数矩阵 F, G, C, D 均为状态 x 的函数,因而实质上描述的是非线性模型。为反映该模型的非线性本质,上述模型可进一步表示为

$$\begin{cases} x(k+1) = F(x)x(k) + G(x)u(k) \\ y(k) = C(x)x(k) + D(x)u(k) \end{cases}$$

若整个状态空间的模糊分割数为 1, 即 $l=1$ 时, 上述的模糊模型即退化为通常的线性状态空间模型。从而常规的线性模型可看成为上述模糊模型的一个特例。

类似地, 对于连续控制对象, 其模糊状态方程模型可表示为

R_p^i : 若 $x_1(t)$ 是 M_1^i and \cdots and $x_n(t)$ 是 M_n^i , 则

$$\begin{cases} \dot{x}(t) = A_i x(t) + B_i u(t) \\ y(t) = C_i x(t) + D_i u(t) \end{cases}$$

或者表示为如下的简洁形式

R_p^i : 若 $x(t)$ 是 M^i , 则

$$\begin{cases} \dot{x}(t) = A_i x(t) + B_i u(t) \\ y(t) = C_i x(t) + D_i u(t) \end{cases}$$

$i=1, 2, \dots, l$ 。则整个系统的状态方程为

$$\begin{cases} \dot{x}(t) = A x(t) + B u(t) \\ y(t) = C x(t) + D u(t) \end{cases}$$

其中

$$\begin{aligned} \mu_i(x) &= M^i(x) / \sum_{j=1}^l M^j(x) & A &= \sum_{i=1}^l \mu_i(x) A_i & B &= \sum_{i=1}^l \mu_i(x) B_i \\ C &= \sum_{i=1}^l \mu_i(x) C_i & D &= \sum_{i=1}^l \mu_i(x) D_i & M(x) &= \prod_{j=1}^n M_j^i(x) \end{aligned}$$

同样假设

$$M(x) \geq 0 \quad \sum_{j=1}^l M^j(x) > 0$$

因此有

$$0 \leq \mu_i(x) \leq 1 \quad \sum_{i=1}^l \mu_i(x) = 1$$

2. 模糊控制器设计 I

对于上面的控制对象模型, 由于它是用局部的线性模型来描述的, 因此可以考虑对于每一个线性子系统首先设计一个局部的线性状态反馈控制器。例如, 可以采用极点配置设计方法或线性二次型最优控制的设计方法来设计这样的状态反馈控制器。因此模糊控制器可以表示为如下的模糊模型。

对于离散系统则为

R_c^i : 若 $x(k)$ 是 M^i , 则

$$u(k) = -L_i x(k)$$

$i=1, 2, \dots, l$ 。其中 R_c^i 表示控制器的第 i 条规则, 整个系统的控制规律则为各个子系统局

部反馈控制的加权和,即

$$u(k) = -Lx(k)$$

$$L = \sum_{i=1}^l \mu_i(x) L_i$$

$\mu_i(x)$ 的意义同前,即它表示第 i 条规则的归一化后的适用度。

对于连续系统则为

R_i : 若 $x(t)$ 是 M^i , 则

$$u(t) = -L_i x(t)$$

$i=1, 2, \dots, l$ 。整个系统的控制规律则为

$$u(t) = -Lx(t)$$

$$L = \sum_{i=1}^l \mu_i(x) L_i$$

由于 L 是 $\mu_i(x)$ 的函数,所以整个系统的控制实质上是非线性的状态反馈。

以上是根据局部的线性子系统来加以设计的,整个系统是否稳定需要加以检验。

结合整个系统的模糊动态模型及控制规律,可得整个系统的模糊状态方程,表示为(先以离散系统为例)

$$\begin{cases} x(k+1) = Fx(k) + Gu(k) = (F - GL)x(k) \\ y(k) = Cx(k) + Du(k) = (C - DL)x(k) \end{cases}$$

系统的稳定性取决于上面第一个方程,将其展开为

$$\begin{aligned} x(k+1) &= \left\{ \sum_{i=1}^l \mu_i(x) F_i - \sum_{i=1}^l \mu_i(x) G_i \sum_{j=1}^l \mu_j(x) L_j \right\} x(k) \\ &= \left[\frac{\sum_{i=1}^l M^i(x) F_i}{\sum_{i=1}^l M^i(x)} - \frac{\sum_{i=1}^l \sum_{j=1}^l M^i(x) M^j(x) G_i L_j}{\sum_{i=1}^l M^i(x) \sum_{j=1}^l M^j(x)} \right] x(k) \\ &= \frac{\sum_{i=1}^l \sum_{j=1}^l M^i(x) M^j(x) (F_i - G_i L_j)}{\sum_{i=1}^l \sum_{j=1}^l M^i(x) M^j(x)} x(k) = \sum_{i=1}^l \sum_{j=1}^l \mu_{ij}(x) H_{ij}(x) x(k) \end{aligned}$$

其中

$$\mu_{ij}(x) = \frac{M^i(x) M^j(x)}{\sum_{i=1}^l \sum_{j=1}^l M^i(x) M^j(x)} \quad H_{ij} = F_i - G_i L_j$$

根据定理 2.7.3,如果存在一个共同的正定矩阵 P ,使得

$$H_{ij}^T P H_{ij} - P < 0 \quad i = 1, 2, \dots, l \quad j = 1, 2, \dots, l$$

则整个系统是全局渐近稳定的。

上述判断稳定性的条件是一个充分条件,目前还没有一个系统的方法来寻找这个共同的 P ,通常只能靠试凑,上一小节给出了一个试凑的步骤。

定理 2.7.5 给出了存在共同的正定矩阵 P 的必要条件,它可以用来帮助确定是否存在这样的共同的 P 。即若 H_{ij} 均为稳定且非奇异矩阵,若能找到一个 $H_{ij}+H_{ji}$ 是不稳定的,则说明一定不存在共同的正定矩阵 P 。

对于连续系统,依照与上面类似的方法可以推得整个系统的模型为

$$\dot{x} = \sum_{i=1}^l \sum_{j=1}^l \mu_{ij}(x) H_{ij}(x) x$$

其中 $\mu_{ij}(x)$ 的意义同上, $H_{ij} = A_i - B_i L_j$ 。

根据定理 2.7.4,如果存在一个共同的正定矩阵 P ,使得

$$H_{ij}^T P + P H_{ij} < 0 \quad i = 1, 2, \dots, l \quad j = 1, 2, \dots, l$$

则整个系统是全局渐近稳定的。

根据定理 2.7.6,若 H_{ij} 均为稳定且非奇异矩阵,若能找到一个 $H_{ij}+H_{ji}$ 是不稳定的,则说明一定不存在共同的正定矩阵 P 。

3. 模糊控制器设计 II

上面所介绍的方法将整个系统的控制取为各个子系统局部反馈控制的加权和,即

$u(t) = -Lx(t)$, $L = \sum_{i=1}^l \mu_i(x) L_i$, 这个想法是十分自然的。其结果导致稳定性分析需要寻找一个共同的正定矩阵 P 。这常常是一件比较困难的工作。

下面介绍另外一种模糊控制器设计方法,其全局控制采用起主导作用的子系统的局部控制,所谓主导作用的子系统是指其隶属度函数取最大值。这个想法也是比较自然的,用模糊模型表示即为(先以离散系统为例)

R_i : 若 $x(k)$ 是 M^i , 则

$$u(k) = -L_i x(k)$$

$i=1, 2, \dots, l$ 。其中 L_i 是起主导作用子系统的反馈控制规律,即

$$\mu_i(x) = \max_j \{ \mu_j(x), i = 1, 2, \dots, l \}$$

整个系统的控制则为

$$u(k) = \sum_{i=1}^l \mu_i(x) [-L_i x(k)] = -L_k x(k)$$

将该控制规律代入控制对象模型方程得

$$\begin{aligned} x(k+1) &= Fx(k) - GL_k x(k) = (F - GL_k)x(k) = \left(\sum_{i=1}^l \mu_i(x) F_i - \sum_{i=1}^l \mu_i(x) G_i L_k \right) x(k) \\ &= \sum_{i=1}^l \mu_i(x) (F_i - G_i L_k) x(k) \end{aligned}$$

$$y(k) = Cx(k) + Du(k) = (C - DL_k)x(k)$$

对于上面的设计,整个系统的稳定性是需要加以检验的。设每个模糊子系统 (F_i, G_i) 是局部能控的,则一定可以设计出局部反馈控制 L_i , 以使得 $H_i = F_i - G_i L_i$ 的特征根均在单位圆内,即一定存在正定对称矩阵 P_i 和 Q_i , 满足如下的李雅普诺夫方程

$$H_i^T P_i H_i - P_i = -Q_i$$

取

$$P = \sum_{i=1}^l \beta_i P_i, \beta_i > 0$$

或

$$\beta_i = 1 \quad P = \sum_{i=1}^l P_i$$

并令

$$Q_k = (F_i - G_i L_k)^T P (F_i - G_i L_k) - P$$

定义

$$\lambda_{k_i} = \lambda_{\max}(Q_{k_i})$$

为 Q_{k_i} 的最大的特征值。则可给出如下的稳定性的定理。

定理 2.7.7 对于上面所描述的离散模糊系统, 如果对于 $k=1, 2, \dots, l$ 均有

$$\sum_{i=1}^l \sum_{j=1}^l \mu_i(x) \mu_j(x) \lambda_{k_i} < 0$$

即

$$\sum_{\substack{i=1 \\ i \neq k}}^l \sum_{\substack{j=1 \\ j \neq k}}^l \mu_i(x) \mu_j(x) \lambda_{k_i} < -\mu_k^2(x) \lambda_{k_k}$$

则整个闭环系统是全局渐近稳定的。

证明: 选取李雅普诺夫函数为

$$V[x(k)] = x^T(k) P x(k)$$

则有

$$\begin{aligned} \Delta V[x(k)] &= V[x(k+1)] - V[x(k)] = x^T(k+1) P x(k+1) - x^T(k) P x(k) \\ &= x^T(k) \sum_{i=1}^l \mu_i(x) (F_i - G_i L_k)^T P \sum_{j=1}^l \mu_j(x) (F_j - G_j L_k) x(k) \\ &\quad - x^T(k) \sum_{i=1}^l \mu_i(x) \sum_{j=1}^l \mu_j(x) P x(k) \\ &= x^T(k) \left\{ \sum_{i=1}^l \sum_{j=1}^l \mu_i(x) \mu_j(x) [(F_i - G_i L_k)^T P (F_j - G_j L_k) - P] \right\} x(k) \end{aligned}$$

令 $H_{ki} = F_i - G_i L_k$, 则上式变为

$$\begin{aligned} \Delta V[x(k)] &= x^T(k) \left\{ \sum_{i=1}^l \sum_{j=1}^l \mu_i(x) \mu_j(x) (H_{ki}^T P H_{kj} - P) \right\} x(k) \\ &= x^T(k) \left\{ \sum_{i=1}^l \mu_i^2(x) (H_{ki}^T P H_{ki} - P) \right. \\ &\quad \left. + \sum_{i < j}^l \mu_i(x) \mu_j(x) (H_{ki}^T P H_{kj} + H_{kj}^T P H_{ki} - 2P) \right\} x(k) \\ &= x^T(k) \left\{ \sum_{i=1}^l \mu_i^2(x) Q_{k_i} + \sum_{i < j}^l \mu_i(x) \mu_j(x) [-(H_{ki} - H_{kj})^T P (H_{ki} - H_{kj}) \right. \\ &\quad \left. + (H_{ki}^T P H_{ki} - P) + (H_{kj}^T P H_{kj} - P)] \right\} x(k) \end{aligned}$$

$$\begin{aligned}
&\leq \mathbf{x}^T(k) \left\{ \sum_{i=1}^l \mu_i^2(\mathbf{x}) Q_{k_i} + \sum_{i < j} \mu_i(\mathbf{x}) \mu_j(\mathbf{x}) (Q_{k_i} + Q_{k_j}) \right\} \mathbf{x}(k) \\
&= \mathbf{x}^T(k) \left\{ \sum_{i=1}^l \sum_{j=1}^l \mu_i(\mathbf{x}) \mu_j(\mathbf{x}) Q_{k_i} \right\} \mathbf{x}(k) \\
&\leq \mathbf{x}^T(k) \left\{ \sum_{i=1}^l \sum_{j=1}^l \mu_i(\mathbf{x}) \mu_j(\mathbf{x}) \lambda_{\max}(Q_{k_i}) \right\} \mathbf{x}(k) \\
&= \mathbf{x}^T(k) \left\{ \sum_{i=1}^l \sum_{j=1}^l \mu_i(\mathbf{x}) \mu_j(\mathbf{x}) \lambda_{k_i} \right\} \mathbf{x}(k)
\end{aligned}$$

所以只要定理条件满足即可保证 $\Delta V[\mathbf{x}(k)] < 0$, 从而保证整个系统是全局渐近稳定的。

可以看出, 为了尽可能比较容易地满足上述定理条件, 当第 k 条规则起主要作用 (即第 k 个模糊子系统为主导子系统) 时, 其它规则的作用应尽可能地小。也就是说, 各模糊集合的隶属度函数重叠部分应尽量少。

对于连续系统, 可采用类似的方法来设计模糊控制器, 即

R_i^l : 若 $\mathbf{x}(t)$ 是 M^l , 则

$$\mathbf{u}(t) = -L_k \mathbf{x}(t)$$

$i=1, 2, \dots, l$ 。其中 L_k 是起主导作用子系统的反馈控制规律, 即

$$\mu_k(\mathbf{x}) = \max_i \{\mu_i(\mathbf{x}), i=1, 2, \dots, l\}$$

整个系统的控制则为

$$\mathbf{u}(t) = \sum_{i=1}^l \mu_i(\mathbf{x}) [-L_k \mathbf{x}(t)] = -L_k \mathbf{x}(t)$$

将该控制规律代入控制对象模型方程得

$$\begin{aligned}
\dot{\mathbf{x}}(t) &= A\mathbf{x}(t) - BL_k \mathbf{x}(t) = (A - BL_k)\mathbf{x}(t) = \left\{ \sum_{i=1}^l \mu_i(\mathbf{x}) A_i - \sum_{i=1}^l \mu_i(\mathbf{x}) B_i L_k \right\} \mathbf{x}(t) \\
&= \sum_{i=1}^l \mu_i(\mathbf{x}) (A_i - B_i L_k) \mathbf{x}(t) \\
\mathbf{y}(t) &= C\mathbf{x}(t) + D\mathbf{u}(t) = (C - DL_k)\mathbf{x}(t)
\end{aligned}$$

设每个模糊子系统 (A_i, B_i) 是局部能控的, 则一定可以设计出局部反馈控制 L_i , 以使得 $H_i = A_i - B_i L_i$ 是稳定的, 即一定存在正定对称矩阵 P_i 和 Q_i , 满足如下的李雅普诺夫方程

$$H_i^T P_i + P_i H_i = -Q_i$$

取

$$P = \sum_{i=1}^l \beta_i P_i, \beta_i > 0$$

或

$$P = \sum_{i=1}^l P_i$$

并令

$$Q_{k_i} = (A_i - B_i L_k)^T P + P (A_i - B_i L_k)$$

定义

$$\lambda_{k_i} = \lambda_{\max}(Q_{k_i})$$

为 Q_{k_i} 的最大的特征值。则可给出如下的稳定性的定理。

定理 2.7.8 对于上面所描述的连续模糊系统,如果对于 $k=1,2,\dots,l$ 均有

$$\sum_{i=1}^l \mu_i(x) \lambda_{k_i} < 0$$

即

$$\sum_{\substack{i=1 \\ i \neq k}}^l \mu_i(x) \lambda_{k_i} < -\mu_k(x) \lambda_{k_k}$$

则整个闭环系统是渐近稳定的。

证明:选取李雅普诺夫函数为

$$V = \mathbf{x}^T(t) P \mathbf{x}(t)$$

结合上述状态方程并令 $H_{k_i} = A_i - B_i L_{k_i}$, 则有

$$\begin{aligned} \dot{V}[\mathbf{x}(t)] &= \dot{\mathbf{x}}^T(t) P \mathbf{x}(t) + \mathbf{x}^T(t) P \dot{\mathbf{x}}(t) = \mathbf{x}^T(t) \left[\sum_{i=1}^l \mu_i(x) H_{k_i}^T P + P \sum_{i=1}^l \mu_i(x) H_{k_i} \right] \mathbf{x}(t) \\ &= \mathbf{x}^T(t) \left[\sum_{i=1}^l \mu_i(x) (H_{k_i}^T P + P H_{k_i}) \right] \mathbf{x}(t) = \mathbf{x}^T(t) \left[\sum_{i=1}^l \mu_i(x) Q_{k_i} \right] \mathbf{x}(t) \\ &\leq \mathbf{x}^T(t) \left[\sum_{i=1}^l \mu_i(x) \lambda_{\max}(Q_{k_i}) \right] \mathbf{x}(t) = \mathbf{x}^T(t) \left[\sum_{i=1}^l \mu_i(x) \lambda_{k_i} \right] \mathbf{x}(t) \end{aligned}$$

所以只要定理条件满足,即可保证 $\dot{V} < 0$, 从保证整个闭环系统是渐近稳定的。

然而在有些情况下,上面的定理条件不一定能满足,因而也就难以确保按上述方法所设计的系统的稳定性。为此可考虑加入一附加的补偿控制来使系统稳定。下面以连续系统为例来加以说明。

取控制规律为

$$\mathbf{u}(t) = \mathbf{u}_f(t) + \mathbf{u}_s(t)$$

其中 $\mathbf{u}_f(t)$ 为按上述方法所设计的控制, $\mathbf{u}_s(t)$ 为补偿控制部分。将上述代入控制对象模型可得整个系统的方程为

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \sum_{i=1}^l \mu_i(x) (A_i - B_i L_{k_i}) \mathbf{x}(t) + B \mathbf{u}_s(t) \\ &= \mu_k(x) (A_k - B_k L_{k_k}) \mathbf{x}(t) + \sum_{\substack{i=1 \\ i \neq k}}^l \mu_i(x) (A_i - B_i L_{k_i}) \mathbf{x}(t) + B \mathbf{u}_s(t) \end{aligned}$$

令

$$H_k = A_k - B_k L_{k_k} \quad \bar{H}_k = \sum_{\substack{i=1 \\ i \neq k}}^l \mu_i(x) (A_i - B_i L_{k_i})$$

则整个系统的状态方程可简写为

$$\dot{\mathbf{x}}(t) = \mu_k(x) H_k \mathbf{x}(t) + \bar{H}_k \mathbf{x}(t) + B \mathbf{u}_s(t)$$

取李雅普诺夫函数为

$$V = \mathbf{x}^T(t) P \mathbf{x}(t)$$

其中

$$P = \sum_{i=1}^l \beta_i P_i, \quad \beta_i > 0$$

或

$$P = \sum_{i=1}^l P_i$$

同时定义

$$\bar{P} = \sum_{\substack{i=1 \\ i \neq k}}^l \beta_i P_i$$

或

$$\bar{P} = \sum_{\substack{i=1 \\ i \neq k}}^l P_i$$

令

$$\bar{Q}_k = H_k^T \bar{P} + \bar{P} H_k$$

则结合上述状态方程可求得

$$\begin{aligned} \dot{V}[\mathbf{x}(t)] &= \dot{\mathbf{x}}^T(t) P \mathbf{x}(t) + \mathbf{x}^T(t) P \dot{\mathbf{x}}(t) \\ &= [\mu_k H_k \mathbf{x}(t) + \bar{H}_k \mathbf{x}(t) - K u_s(t)]^T P \mathbf{x}(t) \\ &\quad + \mathbf{x}^T(t) P [\mu_k H_k \mathbf{x}(t) + \bar{H}_k \mathbf{x}(t) + B u_s(t)] \\ &= \mu_k \mathbf{x}^T(t) (H_k^T P_k + P_k H_k) \mathbf{x}(t) + \mu_k \mathbf{x}^T(t) (H_k^T \bar{P} + \bar{P} H_k) \mathbf{x}(t) \\ &\quad + 2 \mathbf{x}^T(t) P [\bar{H}_k \mathbf{x}(t) + B u_s(t)] \\ &= -\mu_k \mathbf{x}^T(t) Q_k \mathbf{x}(t) + \mu_k \mathbf{x}^T(t) \bar{Q}_k \mathbf{x}(t) + 2 \mathbf{x}^T(t) P [\bar{H}_k \mathbf{x}(t) + B u_s(t)] \end{aligned}$$

取

$$u_s = \begin{cases} -\frac{1}{2} [B^T P \mathbf{x}(t) \mathbf{x}^T(t) P B]^{-1} B^T P \mathbf{x}(t) [\mu_k \mathbf{x}^T(t) \bar{Q}_k \mathbf{x}(t) + 2 \mathbf{x}^T(t) P \bar{H}_k \mathbf{x}(t)] & B^T P \mathbf{x}(t) \mathbf{x}^T(t) P B \neq 0 \\ 0 & B^T P \mathbf{x}(t) \mathbf{x}^T(t) P B = 0 \end{cases}$$

则

$$\dot{V} = -\mu_k \mathbf{x}^T(t) Q_k \mathbf{x}(t)$$

由于 Q_k 为正定矩阵, 所以有 $\dot{V} < 0$, 从而可确保整个系统的全局渐近稳定。

4. 模糊观测器设计 I

上述两种设计方法均要求全部的状态反馈, 对于许多实际系统这是难以实现的。为此也需要构造观测器, 即根据能测量的输出量来重构或估计系统的状态。该小节与下一节分别介绍两种模糊观测器的设计方法。

对于离散系统, 控制对象的模糊模型仍同前, 其模糊观测器可表示为

R_i^o : 若 $\mathbf{x}(k)$ 是 M^i , 则

$$\hat{\mathbf{x}}(k+1) = F_i \hat{\mathbf{x}}(k) + G_i u(k) + K_i [y(k) - \hat{y}(k)]$$

$i=1, 2, \dots, l$ 。其中 R_i^o 表示模糊观测器的第 i 条规则, $y(k)$ 是系统的实际输出, 即

$$y(k) = \sum_{i=1}^l \mu_i C_i \mathbf{x}(k) + \sum_{i=1}^l \mu_i D_i u(k) = C \mathbf{x}(k) + D u(k)$$

$\hat{y}(k)$ 是系统的估计输出, 即

$$\hat{y}(k) = \sum_{i=1}^l \mu_i C_i \hat{x}(k) + \sum_{i=1}^l \mu_i D_i u(k) = C \hat{x}(k) + Du(k)$$

K_i 是按局部子系统 (F_i, C_i) 所设计的观测器增益矩阵。若 (F_i, C_i) 是能观测的, 则采用极点配置或状态最优估计方法等, 可以设计出稳定的局部线性状态观测器, 即 $(F_i - K_i C_i)$ 的极点均在单位圆内。

采用如前所述的典型模糊推理方法, 可得整个系统的观测器方程为

$$\begin{aligned} \hat{x}(k+1) &= \sum_{i=1}^l \mu_i F_i \hat{x}(k) + \sum_{i=1}^l \mu_i G_i u(k) + \sum_{i=1}^l \mu_i K_i [y(k) - \hat{y}(k)] \\ &= F \hat{x}(k) + Gu(k) + K[y(k) - \hat{y}(k)] \end{aligned}$$

其中

$$K = \sum_{i=1}^l \mu_i K_i$$

是整个系统的观测器增益矩阵。

整个系统的观测器方程的稳定性是需要加以检验的。只有它是稳定的, 才能实现观测器的功能。

将观测器方程与控制对象方程相比较, 求出状态估计误差方程为

$$\begin{aligned} \tilde{x}(k+1) &= x(k+1) - \hat{x}(k+1) = (F - KC)\tilde{x}(k) \\ &= \left[\sum_{i=1}^l \mu_i(x) F_i - \sum_{i=1}^l \mu_i(x) K_i \sum_{j=1}^l \mu_j(x) C_j \right] \tilde{x}(k) \\ &= \left[\frac{\sum_{i=1}^l M^i(x) F_i}{\sum_{i=1}^l M^i(x)} - \frac{\sum_{i=1}^l \sum_{j=1}^l M^i(x) M^j(x) K_i C_j}{\sum_{i=1}^l M^i(x) \sum_{j=1}^l M^j(x)} \right] \tilde{x}(k) = \sum_{i=1}^l \sum_{j=1}^l \mu_{ij}(x) S_{ij} \tilde{x}(k) \end{aligned}$$

其中

$$\mu_{ij}(x) = \frac{M^i(x) M^j(x)}{\sum_{i=1}^l \sum_{j=1}^l M^i(x) M^j(x)} \quad S_{ij} = F_i - K_i C_j$$

根据定理 2.7.3, 如果存在一个共同的正定矩阵 P , 使得

$$S_{ij}^T P S_{ij} - P < 0 \quad i = 1, 2, \dots, l \quad j = 1, 2, \dots, l$$

则整个观测器方程是全局渐近稳定的。

上面介绍的是预报观测器的设计方法, 对于现时观测器和降阶观测器, 也可采用类似的方法进行。

对于连续系统, 控制对象的模糊模型仍同前, 其模糊观测器可表示为

R_i : 若 $x(t)$ 是 M^i , 则

$$\dot{\hat{x}}(t) = A_i \hat{x}(t) + B_i u(t) + K_i [y(t) - \hat{y}(t)]$$

$i = 1, 2, \dots, l$ 。其中 R_i 表示模糊观测器的第 i 条规则, $y(t)$ 是系统的实际输出, 即

$$y(t) = \sum_{i=1}^l \mu_i C_i x(t) + \sum_{i=1}^l \mu_i D_i u(t) = Cx(t) + Du(t)$$

$\hat{y}(t)$ 是系统的估计输出, 即

$$\hat{y}(t) = \sum_{i=1}^l \mu_i C_i \hat{x}(t) + \sum_{i=1}^l \mu_i D_i u(t) = C\hat{x}(t) + Du(t)$$

K_i 是按局部子系统 (A_i, C_i) 所设计的观测器增益矩阵。若 (A_i, C_i) 是能观测的, 则采用极点配置或状态最优估计方法等, 可以设计出稳定的局部线性状态观测器, 即 $(A_i - K_i C_i)$ 的极点均在左半平面。

采用如前所述的典型模糊推理方法, 可得整个系统的观测器方程为

$$\begin{aligned} \dot{\hat{x}}(t) &= \sum_{i=1}^l \mu_i A_i \hat{x}(t) + \sum_{i=1}^l \mu_i B_i u(t) + \sum_{i=1}^l \mu_i K_i [y(t) - \hat{y}(t)] \\ &= A\hat{x}(t) + Bu(t) + K[y(t) - \hat{y}(t)] \end{aligned}$$

其中

$$K = \sum_{i=1}^l \mu_i K_i$$

是整个系统的观测器增益矩阵

将观测器方程与控制对象方程相比较, 求出状态估计误差方程为

$$\begin{aligned} \dot{\tilde{x}}(t) &= \dot{x}(t) - \dot{\hat{x}}(t) = (A - KC)\tilde{x}(t) \\ &= \left[\sum_{i=1}^l \mu_i(x) A_i - \sum_{i=1}^l \mu_i(x) K_i \sum_{j=1}^l \mu_j(x) C_j \right] \tilde{x}(t) \\ &= \left[\frac{\sum_{i=1}^l M^i(x) A_i}{\sum_{i=1}^l M^i(x)} - \frac{\sum_{i=1}^l \sum_{j=1}^l M^i(x) M^j(x) K_i C_j}{\sum_{i=1}^l M^i(x) \sum_{j=1}^l M^j(x)} \right] \tilde{x}(t) = \sum_{i=1}^l \sum_{j=1}^l \mu_{ij}(x) S_{ij} \tilde{x}(t) \end{aligned}$$

其中

$$\mu_{ij}(x) = \frac{M^i(x) M^j(x)}{\sum_{i=1}^l \sum_{j=1}^l M^i(x) M^j(x)} \quad S_{ij} = A_i - K_i C_j$$

根据定理 2.7.3, 如果存在一个共同的正定矩阵 P , 使得

$$S_{ij}^T P + P S_{ij} < 0 \quad i = 1, 2, \dots, l \quad j = 1, 2, \dots, l$$

则整个观测器方程是全局渐近稳定的。

5. 模糊观测器设计 I

上面所介绍的方法将整个系统的观测器增益矩阵取为各局部子系统的增益矩阵的加权和, 这个想法是很自然的。其结果导致稳定性分析需要寻找一个共同的正定矩阵 P 。这常常是一件比较困难的工作。

下面介绍另外一种模糊观测器的设计方法, 整个系统的观测器增益矩阵采用起主导作用的局部子系统的增益矩阵。所谓主导作用的子系统是指其隶属度函数取最大值。这个想法也是比较自然的。用模糊模型表示即为(先以离散系统为例)

R_0 : 若 $(x)(k)$ 是 M , 则

$$\hat{x}(k+1) = F_i \hat{x}(k) + G_i u(k) + K_k [y(k) - \hat{y}(k)]$$

$i=1, 2, \dots, l$ 。其中 K_k 是起主导作用的局部子系统的观测器增益矩阵, 即

$$\mu_k(x) = \max_i \{\mu_i(x), i=1, 2, \dots, l\}$$

整个系统的观测器方程则为

$$\begin{aligned} \hat{x}(k+1) &= \sum_{i=1}^l \mu_i F_i \hat{x}(k) + \sum_{i=1}^l \mu_i G_i u(k) + \sum_{i=1}^l \mu_i K_k [y(k) - \hat{y}(k)] \\ &= F \hat{x}(k) + G u(k) + K_k [y(k) - \hat{y}(k)] \end{aligned}$$

容易求得状态估计的误差方程为

$$\begin{aligned} \tilde{x}(k+1) &= (F - K_k C) \tilde{x}(k) = \left[\sum_{i=1}^l \mu_i(x) F_i - K_k \sum_{i=1}^l \mu_i(x) C_i \right] \tilde{x}(k) \\ &= \sum_{i=1}^l \mu_i(x) (F_i - K_k C_i) \tilde{x}(k) \end{aligned}$$

设每个模糊子系统 (F_i, C_i) 是局部能观测的, 则一定可设计出局部的观测器增益矩阵 K_i , 以使得 $S_i = F_i - K_i C_i$ 的特征根均在单位圆内, 即一定存在正定对称矩阵 P_i 和 Q_i , 满足如下的李雅普诺夫方程

$$S_i^T P_i S_i - P_i = -Q_i$$

取

$$P = \sum_{i=1}^l \beta_i P_i \quad \beta_i > 0$$

或

$$P = \sum_{i=1}^l P_i$$

并令

$$Q_{k_i} = (F_i - K_i C_i)^T P (F_i - K_i C_i) - P$$

定义

$$\lambda_{k_i} = \lambda_{\max}(Q_{k_i})$$

为 Q_{k_i} 的最大的特征值, 则可给出如下的稳定性的定理。

定理 2.7.9 对于上面所描述的离散模糊观测器系统, 如果对于 $k=1, 2, \dots, l$ 均有

$$\sum_{i=1}^l \sum_{j=1}^l \mu_i(x) \mu_j(x) \lambda_{k_i} < 0$$

即

$$\sum_{\substack{i=1 \\ i \neq k}}^l \sum_{\substack{j=1 \\ j \neq k}}^l \mu_i(x) \mu_j(x) \lambda_{k_i} < -\mu_k^2(x) \lambda_{k_k}$$

则整个模糊观测器系统是全局渐近稳定的。

该定理的证明过程与定理 2.7.7 相类似, 此处从略。

对于连续系统, 可采用类似的方法来设计模糊观测器, 即

R_0 : 若 $x(t)$ 是 M' , 则

$$\dot{\hat{x}}(t) = A_i \hat{x}(t) + B_i u(t) + K_k [y(t) - \hat{y}(t)]$$

$i=1,2,\dots,l$ 。其中 K_k 是起主导作用的局部子系统的观测器增益矩阵,即

$$\mu_k(x) = \max_i \{\mu_i(x), i=1,2,\dots,l\}$$

整个系统的观测器方程则为

$$\begin{aligned}\dot{\hat{x}}(t) &= \sum_{i=1}^l \mu_i A_i \hat{x}(t) + \sum_{i=1}^l \mu_i B_i u(t) + \sum_{i=1}^l \mu_i K_i [y(t) - \hat{y}(t)] \\ &= A \hat{x}(t) + B u(t) + K_k [y(t) - \hat{y}(t)]\end{aligned}$$

容易求得状态估计的误差方程为

$$\dot{\tilde{x}}(t) = (A - K_k C) \tilde{x}(t) = \sum_{i=1}^l \mu_i(x) (A_i - K_k C_i) \tilde{x}(t)$$

设每个模糊子系统 (A_i, C_i) 是局部能观测的,则一定可设计出局部的观测器增益矩阵 K_i , 以使得 $S_i = A_i - K_i C_i$ 的特征根均在左半平面,即一定存在正定对称矩阵 P_i 和 Q_i , 满足如下的李雅普诺夫方程

$$S_i^T P_i + P_i S_i = -Q_i$$

取

$$P = \sum_{i=1}^l \beta_i P_i, \quad \beta_i > 0$$

或

$$P = \sum_{i=1}^l P_i$$

并令

$$Q_{ki} = (A_i - K_k C_i)^T P + P (A_i - K_k C_i)$$

定义

$$\lambda_{ki} = \lambda_{\max}(Q_{ki})$$

为 Q_{ki} 的最大的特征值,则可给出如下的稳定性的定理。

定理 2.7.10 对于上面所描述的连续模糊观测器系统,如果对于 $k=1,2,\dots,l$ 均有

$$\sum_{i=1}^l \mu_i(x) \lambda_{ki} < 0$$

即

$$\sum_{\substack{i=1 \\ i \neq k}}^l \mu_i(x) \lambda_{ki} < -\mu_k(x) \lambda_{kk}$$

则整个模糊观测器系统是全局渐近稳定的。

该定理的证明过程与定理 2.7.8 类似,此处略。

6. 举例

下面举两个例子来说明前面介绍的第 II 种模糊控制器设计方法。

例 2.17 设已知模糊控制对象为

R_1^1 : 若 $y(t)$ 是 N (负), 则

$$\begin{cases} \dot{x} = A_1x + B_1u \\ y = C_1x \end{cases}$$

R_1^2 : 若 $y(t)$ 是 P (正), 则

$$\begin{cases} \dot{x} = A_2x + B_2u \\ y = C_2x \end{cases}$$

其中

$$A_1 = \begin{bmatrix} 0 & 1 \\ -2.484 & 2.391 \end{bmatrix} \quad B_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad C_1 = [1.227 \quad 0]$$

$$A_2 = \begin{bmatrix} 0.5 & 1 \\ -4.425 & -4.722 \end{bmatrix} \quad B_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad C_2 = [2.025 \quad 0]$$

其归一化隶属度函数分别为

$$\mu_1(y) = \mu_N(y) = 1 - \frac{1}{1 + e^{-2(y-0.5)}}$$

$$\mu_2(y) = \mu_P(y) = \frac{1}{1 + e^{-2(y-0.5)}}$$

注意在该例中的模糊分割是在输出空间而非前面介绍的状态空间。这一点不影响前述设计方法的应用。

设选择两个子系统的闭环特征方程为

$$s^2 + 7.07s + 25 = 0$$

则可分别求得各自的状态反馈控制增益为

$$L_1 = [22.516 \quad 9.461] \quad L_2 = [24.36 \quad 2.848]$$

求解如下的李雅普诺夫方程

$$H_i^T P_i + P_i H_i = -Q_i$$

其中 $H_i = A_i - B_i L_i, i=1, 2$ 。取 $Q_1 = Q_2 = I$ (单位阵), 则可解得

$$P_1 = \begin{bmatrix} 1.9802 & 0.0200 \\ 0.0200 & 0.0736 \end{bmatrix} \quad P_2 = \begin{bmatrix} 2.5767 & 0.0621 \\ 0.0621 & 0.0743 \end{bmatrix}$$

取 $P = P_1 + P_2$, 计算

$$Q_{ki} = (A_i - B_i L_k)^T P + P (A_i - B_i L_k)$$

$$\lambda_{ki} = \lambda_{\max}(Q_{ki})$$

其中 $i=1, 2$ 及 $k=1, 2$, 最后算得

$$\lambda_{11} = -1.8911 \quad \lambda_{22} = -0.1309 \quad \lambda_{12} = 0.2032 \quad \lambda_{21} = 0.0963$$

检验定理 2.7.8 的条件, 当 $k=1 (\mu_1 > \mu_2)$ 时

$$0.2032\mu_2 < 1.8911\mu_1$$

成立。当 $k=2 (\mu_2 > \mu_1)$ 时

$$0.0963\mu_1 < 0.1309\mu_2$$

成立。因而满足定理条件, 说明整个闭环系统是渐近稳定的。

该例中设两个状态均可测, 因而不需要观测器。图 2.41 显示了当 $x(0)$

$= [1.0 \quad 1.0]^T$ 时的仿真结果。

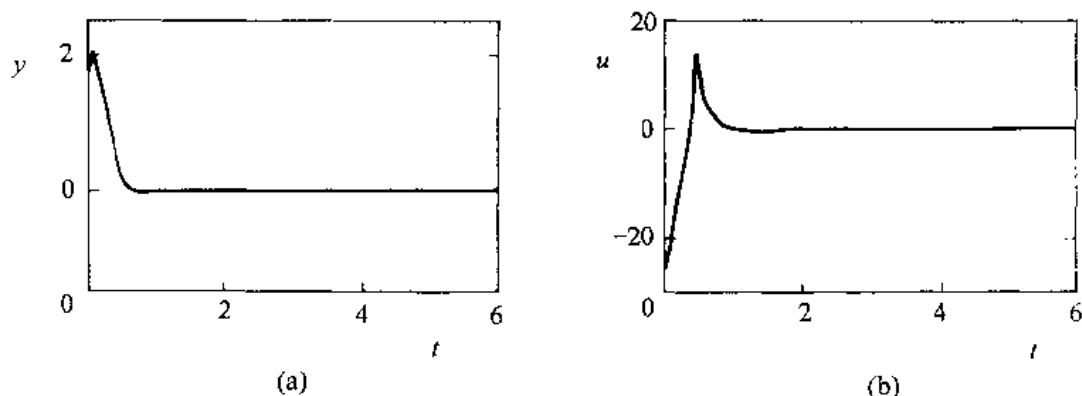


图 2.41 模糊系统的响应

为了比较,图 2.42 给出了只用状态反馈增益 L_2 时的系统的响应。

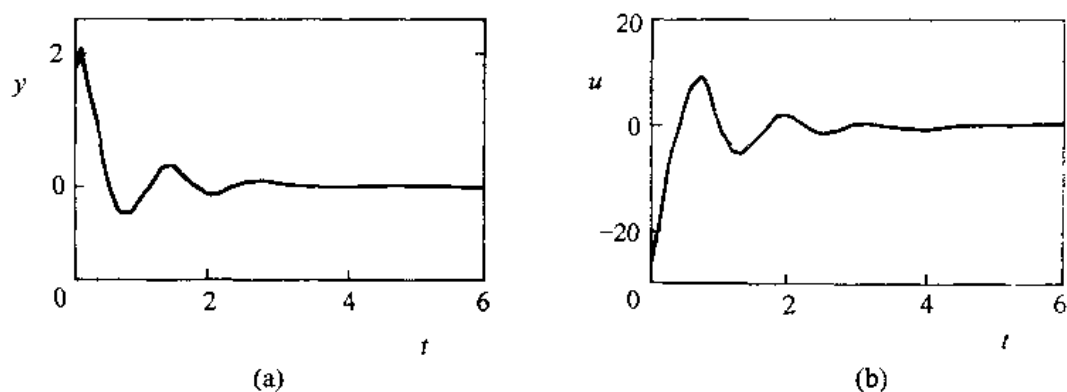


图 2.42 只用单个状态反馈增益的系统响应

通过比较可以看出,上面的模糊控制器设计方法给出了较好的结果。

例 2.18 这是一个单倒立摆的平衡控制问题。已知摆的动力学方程为

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{g \sin(x_1) - a m l x_2^2 \sin(2x_1)/2 - a \cos(x_1)u}{4l/3 - a m l \cos^2(x_1)} \end{aligned}$$

其中 x_1 表示摆与垂直线的偏角, x_2 表示角速度, $g=9.8\text{m/s}^2$ 是重力加速度, m 是摆的质量, $2l$ 是摆的长度, $a=1/(m+M)$, M 是小车的质量, u 是应用于小车的作用力。该例中, $m=2\text{kg}$, $M=8\text{kg}$, $2l=1\text{m}$ 。

通过在 (x_1, x_2) 平面的不同点, 对上述非线性模型进行局部线性化, 可以求得如下的模糊模型。

R_p^1 : 若 x_1 是零附近 and x_2 是零附近, 则

$$\dot{x} = A_1 x + B_1 u$$

R_p^2 : 若 x_1 是零附近 and x_2 是 ± 4 附近, 则

$$\dot{x} = A_2 x + B_2 u$$

R_p^3 : 若 x_1 是 $\pm \pi/3$ 附近 and x_2 是零附近, 则

$$\dot{x} = A_3 x + B_3 u$$

R_p^4 : 若 x_1 是 $\pi/3$ 附近 and x_2 是 4 附近, or x_1 是 $-\pi/3$ 附近 and x_2 是 -4 附近, 则

$$\dot{x} = A_4 x + B_4 u$$

R_p^5 : 若 x_1 是 $\pi/3$ 附近 and x_2 是 -4 附近, or x_1 是 $-\pi/3$ 附近 and x_2 是 4 附近, 则

$$\dot{x} = A_5 x + B_5 u$$

其中

$$\begin{aligned} A_1 &= \begin{bmatrix} 0 & 1 \\ 17.2941 & 0 \end{bmatrix} & B_1 &= \begin{bmatrix} 0 \\ -0.1765 \end{bmatrix} \\ A_2 &= \begin{bmatrix} 0 & 1 \\ 14.4706 & 0 \end{bmatrix} & B_2 &= \begin{bmatrix} 0 \\ -0.1765 \end{bmatrix} \\ A_3 &= \begin{bmatrix} 0 & 1 \\ 5.8512 & 0 \end{bmatrix} & B_3 &= \begin{bmatrix} 0 \\ -0.0779 \end{bmatrix} \\ A_4 &= \begin{bmatrix} 0 & 1 \\ 7.2437 & 0.5399 \end{bmatrix} & B_4 &= \begin{bmatrix} 0 \\ -0.0779 \end{bmatrix} \\ A_5 &= \begin{bmatrix} 0 & 1 \\ 7.2437 & -0.5399 \end{bmatrix} & B_5 &= \begin{bmatrix} 0 \\ -0.0779 \end{bmatrix} \end{aligned}$$

图 2.43 和图 2.44 表示了 x_1 和 x_2 模糊集合。

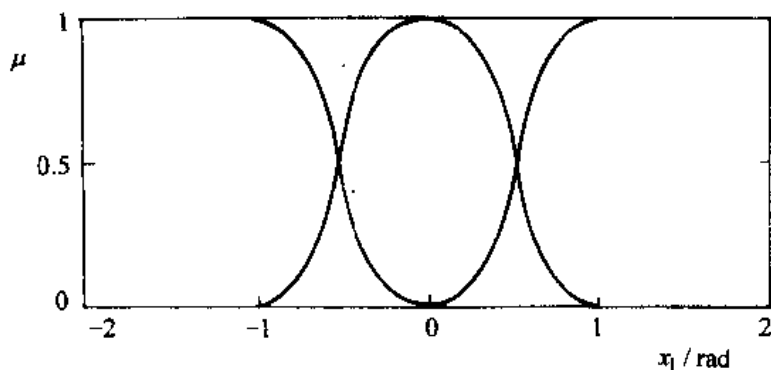


图 2.43 x_1 的模糊集合

对每个局部模型的期望闭环极点选为 $(-4 \pm j5.457)$, 它相当于 10% 的超调量和 1 秒的过渡过程。从而可以求得每个子系统的局部反馈增益矩阵为

$$\begin{aligned} L_1 &= [-357.4135 \quad -45.3333] \\ L_2 &= [-341.4135 \quad -45.3333] \\ L_3 &= [-662.5861 \quad -102.6667] \end{aligned}$$

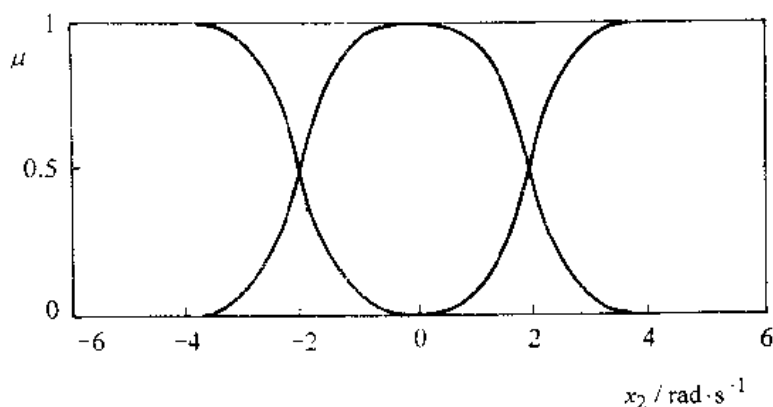


图 2.44 x_2 的模糊集合

$$L_{x_2} = [-680.4563 \quad -109.5949]$$

$$R_{x_2} = [-680.4563 \quad -95.7385]$$

常数 $\beta_i (i=1,2,\dots,5)$ 选择为 $\beta_1=0.5, \beta_2=\beta_3=\beta_4=\beta_5=0.2$, β_1 相当于平衡点状态的系数,因而给予了较大的权重。

对该例进行大量的仿真表明,当 $x_1(0) \in (-80^\circ, 80^\circ)$ 及 $x_2(0)=0$ 时,不管是否采用补偿控制均能使摆倒立平衡,但当 $x_1(0) > 85^\circ$ 或 $x_1(0) < -85^\circ$ 时,若不加补偿控制则不能使摆倒立平衡。加了补偿控制后,任何情况下均能使摆倒立平衡。图 2.45 和图 2.46 显示了两种情况下的仿真结果。可以看出,加上补偿控制后可以确保闭环系统的稳定性。

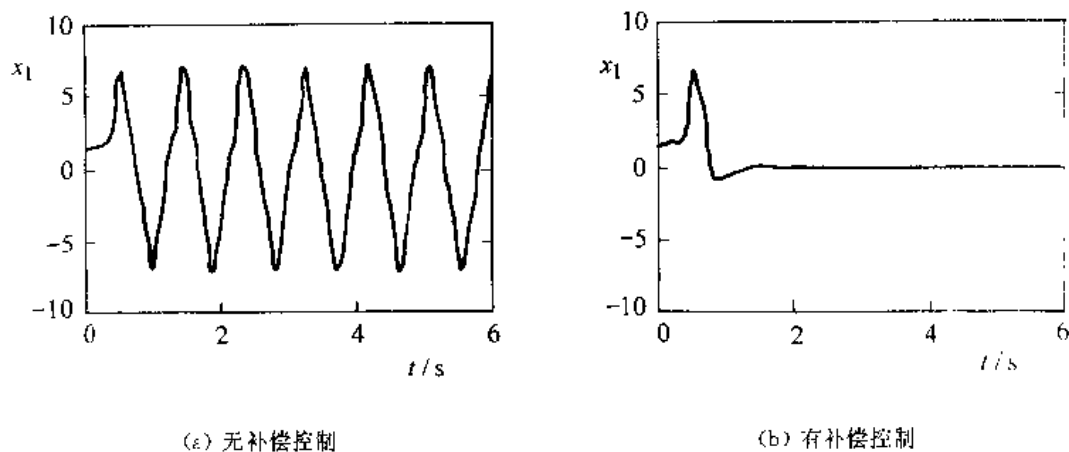
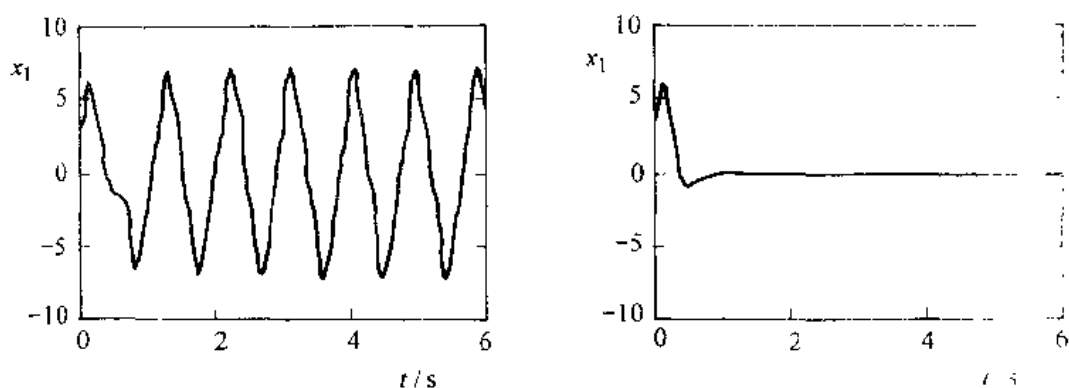


图 2.45 $x(0)=[85^\circ \quad 0]^\top$ 时的系统响应

本节介绍了基于模糊状态模型的模糊系统设计和稳定性分析。它将模糊系统的思想与常规的线性控制系统理论相结合,建立了常规控制与模糊系统控制之间的联系,提出了一种用线性系统理论解决非线性系统问题的新思路和新方法。

应用上述方法的前提是需要已知控制对象的模糊模型,因此如何进行模型系统建模是一个需要进一步研究和解决的问题。



(a) 无补偿控制

(b) 有补偿控制

图 2.46 $x(0)=[175 \quad 0]^T$ 时的系统响应

2.8 自适应模糊控制

模糊控制器的设计主要不依靠被控对象的模型,但是它却非常依靠控制专家或操作人员的经验和知识。模糊控制器的结构非常适于表示人的定性或模糊的经验和知识,这样的经验和知识通常采用 IF-THEN 的模糊条件句(控制规则)来表示。若缺乏这样的控制经验,很难期望它能获得满意的控制效果。仿照常规的控制方法,也许可以采用自适应控制的方法来解决这个问题。

模糊控制器能够比较容易地将人的控制经验溶入到控制器中,这是它的一个非常突出的优点。但是由于采用了 IF-THEN 的控制规则,它不便于控制参数的学习和调整,这一点应该说是模糊控制器的缺点。也正是由于这一点,它使得构造自适应的模糊控制器相对较难。由于神经网络具有易于学习和参数调整的优点,因此模糊控制与神经网络相结合可以吸取两者的优点,构成具有良好性能的模糊神经网络自适应控制。关于这方面内容将在下一章详细介绍。本章只扼要介绍几种典型的自适应模糊控制方法。

与常规自适应控制的结构类似,自适应模糊控制也主要有两类不同的结构形式:一种是根据实际系统性能与要求性能之间的偏差,通过一定的方法来直接调整控制器的参数,其结构如图 2.47 所示。这样的结构通常称为直接自适应模糊控制。另一种是通过在线地进行模糊系统辨识得到控制对象的模型,然后根据所得模型在线地设计模糊控制器,其结构如图 2.48 所示。这样的结构通常称为间接自适应模糊控制。

2.8.1 基于性能反馈的直接自适应模糊控制

本节介绍一种如图 2.47 所示的直接自适应模糊控制方法。在这种结构中,随着可调整的模糊控制参数的不同也有许多不同的自适应控制方法。最常见的有调整尺度变换因子、调整隶属度函数、调整模糊控制规则等。这里介绍一种主要是调整模糊控制规则的自适应控制方法。

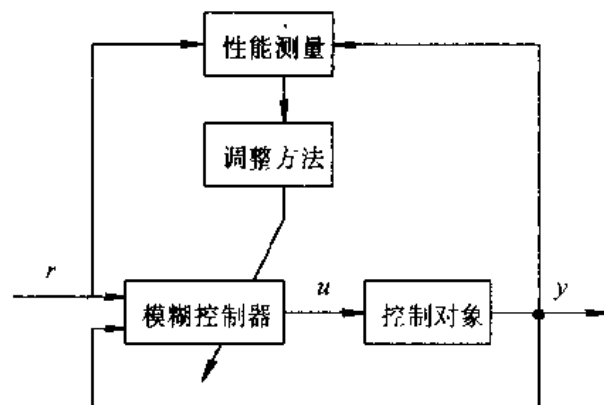


图 2.47 直接自适应模糊控制

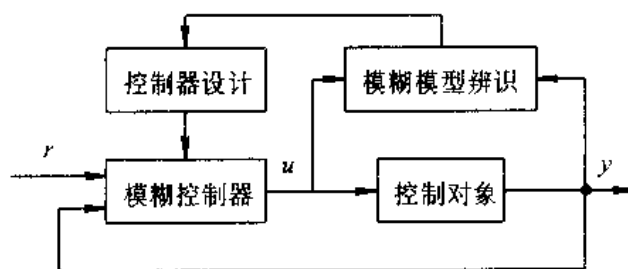


图 2.48 间接自适应模糊控制

1. 性能测量

对于常规的控制系统,其控制系统性能通常用过渡过程时间、超调量及积分指标(如 ISE、ITAE)等性能指标来描述,常规的自适应控制便是要找到这些指标与控制作用之间的联系。由于模糊逻辑控制器的非线性本质,要找出这些指标与模糊控制器控制作用之间的联系是十分困难的。然而,找出系统的局部性能与最近的控制作用之间的联系是可能的。

对控制系统的性能要求是多方面的,如尽量短的过渡过程时间、尽量小的超调量、零稳态误差以及尽量小的控制作用等。有些要求如尽量小的控制作用与尽量短的过渡过程时间,它们之间是互相矛盾的。这些矛盾的指标要求可以用一个期望的闭环系统的响应或一个参考的模型响应来加以统一的描述。控制器的性能与控制对象输出之间的联系可以通过输出误差及变化量等输出状态来加以测量,并由此确定出对控制作用所需的修正量。对于模糊控制系统,这样的性能测量可以用语言规则表来描述,如表 2.16 所示。该语言规则的前件为误差 e 及误差变化 Δe ,后件为期望的输出修正量。若给定各模糊语言的论域范围及模糊语言值的隶属度函数,表 2.16 所示的规则表描述也可转换成如表 2.17 所示的直接数学查表。

由表 2.16 或表 2.17 可以看出,表中的右斜上对角线是不需要进行修正的区域,它表明这时不需要对模糊控制规则进行修改,系统已达到期望的闭环性能。事实上,这时的闭环系统性能可以具体求出。当系统处于零修正区域时有

表 2.16 语言变量描述的性能测量

$\begin{matrix} P \\ \Delta e \end{matrix} \backslash e$	NB	NM	NS	ZE	PS	PM	PB
NB	NB	NB	NB	NB	NM	NS	ZE
NM	NB	NB	NM	NM	NS	ZE	PS
NS	NB	NM	NM	NS	ZE	PS	PM
ZE	NB	NM	NS	ZE	PS	PM	PB
PS	NM	NS	ZE	PS	PM	PM	PB
PM	NS	ZE	PS	PM	PM	PB	PB
PB	ZE	PS	PM	PB	PB	PB	PB

表 2.17 性能测量的直接数字查表

$\begin{matrix} P \\ e \end{matrix} \backslash e$	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
-6	-6	-6	-6	-6	-6	-6	-6	-5	-4	-3	-2	-1	0
-5	-6	-6	-6	-5.5	-5	-5	-5	-4	-3	-2	-1	0	1
-4	-6	-6	-6	-5	-4	-4	-4	-3	-2	-1	0	1	2
-3	-6	-5.5	-5	-4.5	-4	-3.5	-3	-2	-1	0	1	2	3
-2	-5.5	-5	-4	-4	-4	-3	-2	-1	0	1	2	3	4
-1	-5.5	-5	-4	-3.5	-3	-2	-1	0	1	2	3	4	5
0	-5	-4.5	-4	-3	-2	-1	0	1	2	3	4	5	6
1	-4.5	-4	-3	-2	-1	0	1	2	3	3.5	4	5	6
2	-4	-3	-2	-1	0	1	2	3	4	4	4	5	6
3	-3	-2	-1	0	1	2	3	3.5	4	4.5	5	5.5	6
4	-2	-1	0	1	2	3	4	4	4	5	6	6	6
5	-1	0	1	2	3	4	4.5	5	5	5.5	6	6	6
6	0	1	2	3	4	4.5	5	5.5	5.5	6	6	6	6

$$e + \Delta e = 0$$

这里 e 和 Δe 均为经尺度变换后的量, 设 $e = K_e \bar{e}$, $\Delta e = K_{\Delta} \Delta \bar{e}$ (K_e 和 K_{Δ} 表示尺度变换因子), 上式变为

$$K_e \bar{e} + K_{\Delta} \Delta \bar{e} = 0$$

注意到 $\bar{e} = r - y$, 并设参考输入 r 为常数, 则上式可进一步化为

$$K_e y + K_\Delta T \dot{y} = K_e r$$

其中 T 为采样周期, 写成传递函数形式则有

$$\frac{y}{r} = \frac{1}{T_m s + 1}$$

其中 $T_m = K_\Delta T / K_e$ 。可见若系统处于零修正区域, 闭环系统为时间常数为 T_m 的一阶响应, 从而通过适当设定 K_e, K_Δ 和 T , 可以获得要求的闭环系统响应。

类似地, 若设定模糊控制器的输入为 $e, \Delta e$ 和 $\Delta^2 e$, 则相应的零修正区域为

$$e + \Delta e + \Delta^2 e = 0$$

类似地可以求得

$$\frac{y}{r} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

其中

$$\omega_n = \sqrt{\frac{K_e}{K_{\Delta^2} T^2}} \quad \zeta = \frac{K_\Delta}{2\sqrt{K_e K_{\Delta^2}}}$$

通过适当地选择 $K_e, K_\Delta, K_{\Delta^2}$ 和 T , 可以获得要求的二阶响应。

2. 控制对象的增量模型

上面的性能测量给出了为达到期望的系统性能所需要的输出修正量。为了实现自适应控制, 需要将该输出修正量变换为所需的控制修正量, 因而它需要对控制对象的特性有一定的了解。其中主要的有以下几个问题。

(1) 需要知道过去哪一时刻的控制量影响当前时刻的系统性能, 这就需要知道控制对象的延迟时间 dT (T 为采样周期, d 为延迟的拍数), 它决定了应对哪一时刻的控制作用加以修正。它取决于系统的动力学特性, 对于高阶系统, 也许需要对过去一系列时刻的控制作用加以修正。

(2) 对于多变量系统, 对于给定的输出修正量, 需要知道应修正哪一个输入控制作用以及所需的修正量。多变量系统带来了输入与输出之间的交叉耦合, 因而需要知道控制对象的增量模型 J (J 表示控制对象输出对输入的雅可比矩阵), 从而可求得相应于控制输入的修正量应为

$$P_i(kT) = J^{-1} P_o(kT)$$

其中 $P_o(kT)$ 表示输出修正量, $P_i(kT)$ 表示输入修正量。这里对于增量模型 J 并不要求很准确。

(3) 如何修改控制规则库, 以实现所要求的修正。这是下面需要详细讨论的问题。

3. 控制规则库的修正

设 $e(kT-dT), \Delta e(kT-dT), u(kT-dT)$ 表示 d 拍之前的误差、误差变化及控制量。根据上面的讨论, 已求得 $P_i(kT)$ 为控制输入的校正量, 也就是说, 为使在 kT 时刻获得期望的响应性能, $(k-d)T$ 时刻的控制量应为 $u(kT-dT) + P_i(kT)$, 将这些量模糊化得

$$E(kT-dT) = \text{fz}[e(kT-dT)]$$

$$\Delta E(kT-dT) = \text{fz}[\Delta e(kT-dT)]$$

$$U(kT-dT) = \text{fz}[u(kT-dT)]$$

$$V(kT-dT) = \text{fz}[u(kT-dT) + P_i(kT)]$$

这里一般不采用单点模糊集合, 而采用三角模糊集合的模糊化方法(具体方法见第 2.6.2

节),以使每次修正不只是一个点,而是该点附近一个局部区域,从而增强自适应控制的泛化能力。经如此模糊化后,原来的控制相当于执行了如下的控制规则:

若 $e(kT-dT)$ 是 $E(kT-dT)$ and $\Delta e(kT-dT)$ 是 $\Delta E(kT-dT)$, 则 u 是 $U(kT-dT)$ 该控制规则需修改为

若 $e(kT-dT)$ 是 $E(kT-dT)$ and $\Delta e(kT-dT)$ 是 $\Delta E(kT-dT)$, 则 u 是 $V(kT-dT)$ 写成模糊关系矩阵则为

$$R_1(kT) = E(kT-dT) \times \Delta E(kT-dT) \times U(kT-dT)$$

$$R_2(kT) = E(kT-dT) \times \Delta E(kT-dT) \times V(kT-dT)$$

设 kT 时刻控制器的总模糊关系矩阵为 $R(kT)$, 修改后的模糊关系矩阵为 $R(kT+T)$, 则为实现上述修正可使

$$R(kT+T) = [R(kT) \wedge \bar{R}_1(kT)] \vee R_2(kT)$$

其中 $\bar{R}(kT)$ 是 $R(kT)$ 的补。根据所测得的误差 $e(kT)$ 、误差变化 $\Delta e(kT)$, 将它们模糊化后与 $R(kT+T)$ 进行合成运算使得模糊控制量 $U(kT)$, 再对 $U(kT)$ 进行清晰化运算使得清晰控制量 $u(kT)$ 。每一采样时刻都按照这样的步骤进行计算, 便实现了自适应模糊控制的功能。

这种方法的缺点是: (1) 原有的控制规则丢失了, 而且难以恢复; (2) 计算工作量大且需占用较多的存储容量; (3) $R_1(kT)$ 和 $R_2(kT)$ 是稀疏矩阵, 浪费很多计算时间; (4) 对于多变量系统, 关系矩阵 $R(kT)$ 非常庞大。

上述第(1)条缺点并不是本质的, 在模糊推理计算中, 模糊关系矩阵是最关键的, 是否保留了原来的控制规则并不重要。第(4)条缺点并不是自适应模糊控制方法带来的, 即使对于通常的模糊控制计算, 当输入维数很高时也存在 R 很庞大的缺点。第(2)和第(3)条缺点实质上是一条, 即是第(3)条所引起。这一缺点可通过在算法中采取措施在一定程度上加以克服。该自适应模糊控制方法虽然有上述一些缺点, 但对于较为简单的系统且计算速度和容量许可的情况下, 这种方法还是可行的。

为了确保自适应模糊控制不产生发散的响应, 恰当地选取初始的模糊控制规则是很重要的, 下面几条规则是必须遵循的。

(1) R_0 : 若 E 是 ZE (零) and ΔE 是 ZE , 则 U 是 ZE 。这条规则保证当输出等于期望值时是系统的平衡状态。

(2) R_1 : 若 $(E, \Delta E)$ 符号相同时, 则 U 也应具有相同的符号。这些规则确保系统输出能快速地收敛到设定值。

(3) 控制规则库必须是对称的, 即 $R(E, \Delta E) = -R(-E, -\Delta E)$, 以便改善系统的收敛特性及控制超调。

4. 尺度变换因子的选择

输入尺度变换因子 $(K_e, K_{\Delta e}, K_{\Delta^2 e})$ 决定了在性能测量以及控制规则库中模糊集合的论域。根据前面的讨论, 它们可以根据期望的闭环响应性能来进行选择, 这样的选择并不是唯一的, 可以有很多种组合来达到同样的性能要求。

由于输入尺度变换是将实际的输入量变换为模糊集合的论域, 因而变换因子直接取决于容许的最大输入量及模糊集合论域量化的分级数。所以开始可将它们选择为

$$K_e = \frac{q}{e_m} \quad K_{\Delta e} = \frac{q}{\Delta e_m} \quad K_{\Delta^2 e} = \frac{q}{\Delta^2 e_m}$$

其中 q 表示模糊集合论域的分级数, \bar{e}_m , $\Delta \bar{e}_m$ 和 $\Delta^2 \bar{e}_m$ 表示实际输入量的最大变化范围。类似地, 输出尺度变换因子可初选为

$$K_u = \frac{\bar{u}_m}{u_m}$$

其中 u_m 表示模糊控制器输出的论域大小, \bar{u}_m 是实际控制量的最大变化范围。

以上是根据实际量的变化范围及论域大小所初选的尺度变换因子。这些参数对系统的性能有很大影响, 它们可以在自适应控制过程中根据性能的要求作适当的调整。

研究表明, 这些尺度变换因子对系统的性能具有如下的影响。

(1) K_e 小将引起较大的稳态误差, 有时由于在平衡位置的不灵敏而可能导致自持振荡; K_e 大将导致超调量变大。

(2) $K_{\Delta e}$ 小将使系统响应性能变差及模糊关系 $R(kT)$ 收敛变慢; $K_{\Delta e}$ 大将导致上升时间增加, 稳态误差增加及超调量的减少。

(3) 增加 $K_{\Delta^2 e}$ 导致上升时间和超调量的增加。

(4) 当控制对象的延迟拍数 d 增加时, 为了使 $R(kT)$ 有好的收敛特性, 必须相应增加 $K_{\Delta e}$ 和 $K_{\Delta^2 e}$ 。

(5) K_u 小将导致上升时间增加和 $R(kT)$ 的收敛速度加快; K_u 变大其作用相反。

如果选取尺度变换因子或论域分级数是误差 e 的函数, 则往往可以改善系统的动态响应性能和模糊关系的收敛特性。一般说来, 增加系统的增益将提高系统的稳态精度, 而暂态响应的性能变差。为此, 可以在开始(即误差较大)时采用小的增益, 当系统输出接近设定点时再转入高增益。采用以对数标度的非线性量化方法可以达到相同的效果。

5. 设计步骤

由图 2.47 可见, 该自适应模糊控制由两级组成: 下面一级是基本的模糊控制级, 上面是自适应级。下面具体介绍它们的设计步骤。

(1) 基本模糊控制级。

a. 确定实际输入量的最大变化范围 \bar{e}_m , $\Delta \bar{e}_m$ 和 $\Delta^2 \bar{e}_m$ 和模糊变量 e , Δe 和 $\Delta^2 e$ 的论域量化分级数。采用如上面所讨论的尺度变换及非线性量化方法将实际的输入量变换为论域范围的模糊变量。

b. 确定模糊语言值及相应的隶属度函数。

c. 给出如表 2.16 所示的性能测量语言变量描述及如表 2.17 所示的直接数字查表。

d. 按照前面所给出的几条原则, 恰当地选取初始模糊控制规则。

(2) 自适应级

a. 确定控制对象的增量模型 J 和延迟拍数 d 。

b. 根据实际输入量的范围及论域量化的分级数初选尺度变换因子 K_e , $K_{\Delta e}$ 和 $K_{\Delta^2 e}$ 。

c. 根据零修正区域的条件, 检验所选变换因子是否满足闭环系统的性能要求, 若不满足, 可对它们作适当调整。

d. 检验对于初选参数系统是否稳定, 若系统不稳定可适当增加 $K_{\Delta e}$ 和减少 $K_{\Delta^2 e}$ 以保

e. 若初始系统稳定但动态响应性能不满足要求,按照图 2.49 的流程图对系统进行自适应控制,其中除了自适应调整控制规则库外(实际上是调整模糊关系 R),重点标示了调整尺度变换增益参数的过程。

f. 将自适应调整获得的结果参数存储起来,当设定点改变时重复上述步骤,最后可获得尺度变换增益参数与设定点的对应关系表。实际运行时可通过查表方法来确定这些增益参数。当运行条件非表中给定的状态时,可通过插值的方式来确定。

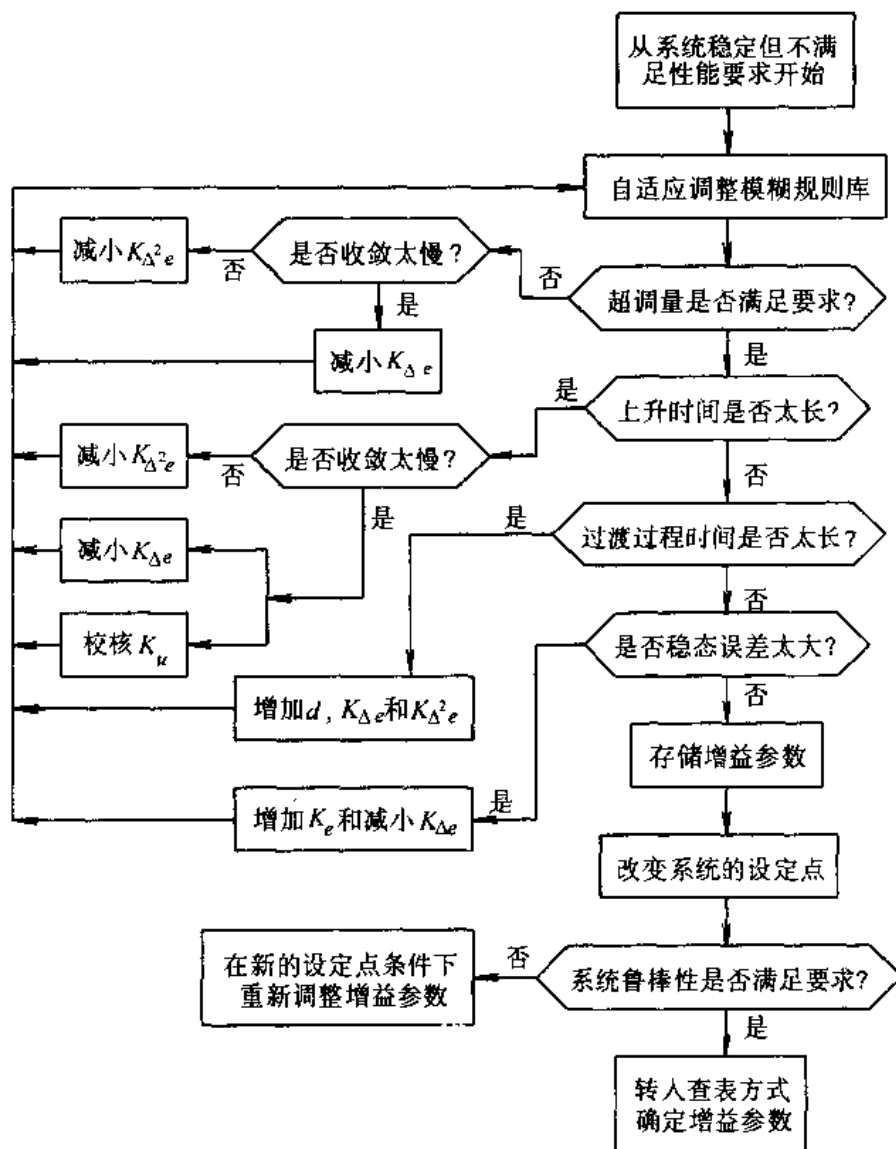


图 2.49 自适应模糊控制尺度变换增量参数调整流程图

2.8.2 基于模糊模型求逆的间接自适应模糊控制

本节介绍如图 2.48 所示的间接模糊自适应控制方法。该方法首先在线地辨识控制对

象的模糊模型,然后利用该模型并根据期望的闭环系统性能设计出模糊控制器。该方法主要有以下一些优点。

(1) 通过明显地包括模型辨识可以检测到模型参数的突然变化以及跟踪模型参数随时间变化的特性,这对于智能故障诊断是非常有用的。

(2) 将模型辨识与控制器设计的过程相分离可以将模型或参数辨识的收敛性与控制器的性能及系统的稳定性分析分开进行。这实质上是一种模糊形式的分离性原理。

(3) 可以改变控制器的性能指标来适应不同的环境限制而并不影响模型规则库。

1. 自适应模糊模型辨识

若考虑离散模型,其模糊关系可以表示为

R_i : 如果 $y(k)$ 是 A^i and $\bar{u}(k)$ 是 B^i , 则 $y(k+1)$ 是 C^i

其中

$$\bar{y}(k) = \begin{bmatrix} y(k) \\ y(k-1) \\ \vdots \\ y(k-n+1) \end{bmatrix} \quad \bar{u}(k) = \begin{bmatrix} u(k) \\ u(k-1) \\ \vdots \\ u(k-m) \end{bmatrix}$$

$$A^i = A_0^i \times A_1^i \times \cdots \times A_{n-1}^i \quad B^i = B_0^i \times B_1^i \times \cdots \times B_m^i$$

$i=1, 2, \dots, N$ 。上述模型也可表示成

$$R_P = [\bar{y}(k) \times \bar{u}(k)] \rightarrow y(k+1)$$

$$y(k+1) = [\bar{y}(k) \times \bar{u}(k)] \circ R_P$$

若模糊蕴含及 and 采用求交运算,则有

$$\mu_{R_i}[\bar{y}(k), \bar{u}(k), y(k+1)] = \min\{\mu_{A^i}[\bar{y}(k)], \mu_{B^i}[\bar{u}(k)], \mu_{C^i}[y(k+1)]\}$$

其中定义

$$\mu_{A^i}[\bar{y}(k)] = \min\{\mu_{A_0^i}[y(k)], \mu_{A_1^i}[y(k-1)], \dots, \mu_{A_{n-1}^i}[y(k-n+1)]\}$$

$$\mu_{B^i}[\bar{u}(k)] = \min\{\mu_{B_0^i}[u(k)], \mu_{B_1^i}[u(k-1)], \dots, \mu_{B_m^i}[u(k-m)]\}$$

若 also 采用求并运算,则有 $R_P = \bigcup_{i=1}^N R_i$,

即 $\mu_{R_P}[\bar{y}(k), \bar{u}(k), y(k+1)] = \max\{\mu_{R_i}[\bar{y}(k), \bar{u}(k), y(k+1)]\}_{i=1, 2, \dots, N}$

模糊系统辨识的问题是,如何根据测得的系统输入输出数据来构造系统的模糊关系 R_P 。设已知系统初始状态 $\bar{y}(0)$ 和 $\bar{u}(0)$, 并测得输入输出数据为 $y(k) (k=1, 2, \dots, N+1)$ 和 $u(k) (k=1, 2, \dots, N)$ 。首先根据这些原始数据构造如下的 N 个输入输出数据组:

$$\{\bar{y}(k), \bar{u}(k), y(k+1)\}_{k=1, 2, \dots, N}$$

对每一数据组均考虑为一条模糊规则 R_i , 其中 A^i, B^i 和 C^i 取为由这些数据经模糊化得到的模糊集合。这里模糊化方法一般不采用单点模糊集合而采用三角形模糊集合, 数据所在位置为模糊集合的中心点。

在获得 N 条模糊规则 $R_i (i=1, 2, \dots, N)$ 后, 即可按照上面给出的方法构造系统总的模糊关系 R_P 。

上面介绍的方法是对 N 组数据成批处理而获得系统的模糊模型, 它相当于常规系统

辨识中的批处理算法。这样的批处理算法适用于定常模型的离线辨识,而不适于时变系统的在线辨识。为便于自适应控制,可仿照常规系统的方法,采用如下的模糊系统辨识的递推算法。

$$R_p(k+1) = \lambda R_p(k) \cup R_{k+1}$$

其中 $0 < \lambda < 1$, λ 称为遗忘因子。 R_{k+1} 是由新获取的第 $(k+1)$ 组数据所建立的模型关系,即

$$R_{k+1} = [\mathbf{y}(k+1) \times \bar{\mathbf{u}}(k+1)] \rightarrow y(k+2)$$

注意这里 R_{k+1} 可以是如上所示的只有一组新数据所获得的模糊关系,也可以是多组数据所获得的模糊模型。设新获得 s 组数据,则

$$R_{k+1} = \bigcup_{i=1}^s R_{k+1}^i$$

其中 R_{k+1}^i 表示第 i 组新数据所获得的模糊关系。

以上考虑的离散模型的模糊辨识,对于连续模型可以得到类似的结果,即

$$R_p(k+1) = \lambda R_p(k) \cup R_{k+1}$$

其中 $R_{k+1} = (\bar{\mathbf{y}}_{k+1} \times \bar{\mathbf{u}}_{k+1}) \rightarrow y_{k+1}^{(n)}$, 且定义

$$\bar{\mathbf{y}}_{k+1} = \begin{bmatrix} y_{k+1} \\ \dot{y}_{k+1} \\ \vdots \\ y_{k+1}^{(n-1)} \end{bmatrix}, \quad \bar{\mathbf{u}}_{k+1} = \begin{bmatrix} u_{k+1} \\ \dot{u}_{k+1} \\ \vdots \\ u_{k+1}^{(m)} \end{bmatrix}$$

这里的下标 $(k+1)$ 表示第 $(k+1)$ 组数据。类似地,这里 R_{k+1} 也可表示多组数据所获得的模糊模型。

在具体应用上述自适应模糊系统辨识方法时,尚有以下几个问题需要进一步讨论。

(1) 当输入数据主要集中在输入信号空间的某一区域时,采用上述递推计算方法将导致在该区域之外的模糊关系逐渐衰减直至到零,这是不希望。为了避免这种情况,应考虑只对输入数据附近的区域进行修改。为此,可设置当

$$\min \{ \mu_A^a[\bar{\mathbf{y}}(k)], \mu_B^b[\bar{\mathbf{u}}(k)] \} \geq \theta$$

时才进行上述递推关系的修改,其中, $\theta \in [0, 1]$ 为阈值。

(2) 遗忘因子 λ 的选择。 λ 在 0 到 1 之间选择,当 λ 选择较大时,收敛较慢;但 λ 取得太小时,虽然收敛比较快,但对于数据噪声也比较敏感,这是不希望的,因此应恰当地选择 λ 的大小。

(3) 占用存储量大小。对于上述离散或连续模型,模糊关系 R_p 的维数为 $n+m+2$ ($\mathbf{y}(k) \cdots n$ 维, $\mathbf{u}(k) \cdots m+1$ 维, $y(k+1) \cdots 1$ 维),若每一维的量化等级数为 q ,则总共需 q^{n+m+2} 个存储空间来存储隶属度函数。量化等级数越多,模型表示越准确。因此这里需在存储容量与模型精度之间进行折中选择。

(4) 模型精度与泛化能力。前面提到每个测量数均模糊化为一个三角形模糊集合。三角形越窄瘦,当数据足够多时,所获得的模型有可能越准确。相反,三角形越宽时,泛化能力越强。因此应适当选择三角形的宽度,它至少应大于二倍的量化步长。

(5) 自适应模糊辨识算法的收敛性。由于算法中包含了取小和取大的非线性操作,所以严格地证明算法的收敛性是很困难的。但若取小运算改用相乘且隶属度函数用 k 阶基

样条函数,则上述辨识算法的收敛性是可以得到证明的。

下面举一个非常简单的例子来说明利用上述方法进行模糊系统辨识的过程

例 2.19 设有非线性的输入输出关系为 $y=u^2$,且该模型是未知的,通过测量获得该模型的输入输出数据如表 2.18 所示。要求根据这些量测数据建立它的模糊模型。

表 2.18 待建模型的输入输出数据

u	0	1	2	3	4	5
y	0	1	4	9	16	25

(1) 对实际的输入输出数据进行量化。设将输入和输出都均匀地量化为 6 个等级,则量化后的输入输出数据将变为如表 2.19 所示。

表 2.19 量化后的输入输出数据

u^*	0	1	2	3	4	5
y^*	0	0	5	10	15	25

(2) 每一组数据相当一条模糊规则,即

R_1 : 如果 u^* 是 A^1 ,则 y^* 是 B^1

R_2 : 如果 u^* 是 A^2 ,则 y^* 是 B^1

R_3 : 如果 u^* 是 A^3 ,则 y^* 是 B^2

R_4 : 如果 u^* 是 A^4 ,则 y^* 是 B^3

R_5 : 如果 u^* 是 A^5 ,则 y^* 是 B^4

R_6 : 如果 u^* 是 A^6 ,则 y^* 是 B^6

其中 A' 和 B' 是根据量化后的输入输出数据用三角形模糊集合的模糊化方法而求得的模糊集合,其宽度取为 2(这里定义三角形底边长的一半为宽度)。 A' 和 B' 的具体隶属度函数如表 2.20 和表 2.21,或如图 2.50 和图 2.51 所示。

表 2.20 输入量的隶属度函数

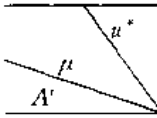
	0	1	2	3	4	5
A^1	1	0.5	0	0	0	0
A^2	0.5	1	0.5	0	0	0
A^3	0	0.5	1	0.5	0	0
A^4	0	0	0.5	1	0.5	0
A^5	0	0	0	0.5	1	0.5
A^6	0	0	0	0	0.5	1

表 2.21 输出量的隶属度函数

$u \backslash B^i$	0	5	10	15	20	25
B^1	1	0.5	0	0	0	0
B^2	0.5	1	0.5	0	0	0
B^3	0	0.5	1	0.5	0	0
B^4	0	0	0.5	1	0.5	0
B^5	0	0	0	0.5	1	0.5
B^6	0	0	0	0	0.5	1

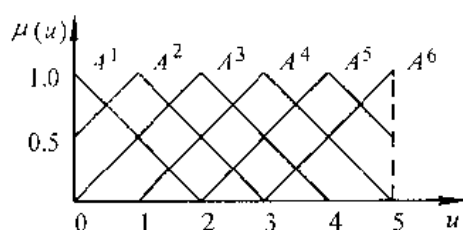


图 2.50 输入量的隶属度函数

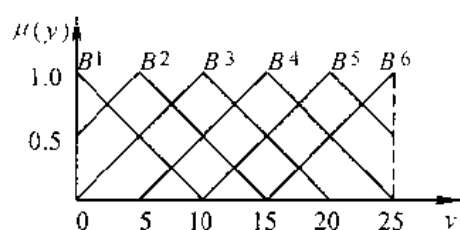


图 2.51 输出量的隶属度函数

(3) 计算模糊关系矩阵 R 。若用批处理算法,则

$$R = \bigcup_{i=1}^6 R_i$$

若用递推算法,即

$$R(k+1) = \lambda R(k) \cup R_{k+1}$$

$k=0, 1, \dots, 5$ 。不难求得

$$\begin{aligned}
 R_1 &= A^1 \rightarrow B^1 \\
 &= \begin{bmatrix} 1 \\ 0.5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \wedge [1 \quad 0.5 \quad 0 \quad 0 \quad 0 \quad 0] = \begin{bmatrix} 1 & 0.5 & 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}
 \end{aligned}$$

按照同样的方法可求出 $R_2 \sim R_6$ 。将它们代入上面的批处理算法或递推算法(在用递推算法时只在 $\mu_{A^i}(u) \geq 0.5$ 时才进行修正,并取 $R(0)=0$ 及 $\lambda=1$)均得到

$$R = \begin{bmatrix} 1 & 0.5 & 0 & 0 & 0 & 0 \\ 1 & 0.5 & 0.5 & 0 & 0 & 0 \\ 0.5 & 1 & 0.5 & 0.5 & 0 & 0 \\ 0.5 & 0.5 & 1 & 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 1 & 0.5 & 0.5 \\ 0 & 0 & 0.5 & 0.5 & 0.5 & 1 \end{bmatrix}$$

(1) 校核所求模型的精度。利用如下模糊逻辑推理计算过程

$$u \xrightarrow{\text{量化}} u' \xrightarrow{\text{模糊化}} U' \xrightarrow{\text{合成运算}} Y' = U' \circ R \xrightarrow{\text{清晰化}} y$$

可以根据给定的 u 推理计算出 y 。这里模糊化采用单点模糊集合,清晰化采用最大法,所得结果如表 2.22 和图 2.52 所示

表 2.22 模糊建模的精度校核

u	0	1	2	3	4	5
$y_{\text{实}}$	0	1	4	9	16	25
$y_{\text{估}}$	0	0	5	10	15	22

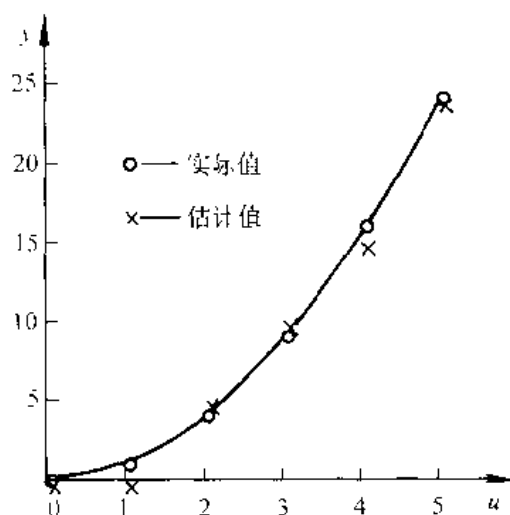


图 2.52 模糊建模的精度校核

从该例可以看出,利用上述模糊建模的方法可以获得对未知模型的估计。虽然估计的结果有一定的误差,但该误差可随着量化级数的增加而减少。当然建模精度是与存储量的要求相矛盾的,需折中考虑。

该例还说明了模糊辨识方法的另一突出优点,它不需要事先知道模型的非线性特性的结构形式。而常规的参数估计或曲线拟合需事先给定函数的形式,然后再进行参数的最优估计。

2. 模糊模型求逆

在第 2.7.3 节中,我们结合小车的模糊控制介绍了基于语言模型求逆的模糊控制器的设计。本节介绍的自适应模糊控制也是基于模型求逆。所以这里首先介绍模糊模型求

逆的方法。

若对于如下的一阶系统模型

R_i : 如果 y 是 A^i and u 是 B^i , 则 \dot{y} 是 $C^i, i=1, 2, \dots, N$ 。也即

$$R = (y \times u) \rightarrow \dot{y}$$

这里 R 称为该一阶系统的正向模糊模型。根据该正向模型, 可以根据输入量 u 及当前状态 y 求出输出量 \dot{y} , 即

$$\dot{y} = (y \times u) \circ R$$

该模型的逆问题是: 若已知 y 及 \dot{y} , 如何根据已知的正向模型求出对应输入量 u , 即要求

$$R^{-1} = (y \times \dot{y}) \rightarrow u$$

或写成模糊规则形式为

R_i^{-1} : 如果 y 是 A^i and \dot{y} 是 C^i , 则 u 是 $B^i, i=1, 2, \dots, N$ 。

由于基于模糊关系 R 的正向推理计算中主要包含了取小和取大的运算, 这是严重的非线性运算关系, 而且其输入输出关系并非一一对应的, 即可能许多组输入对应同一个输出。因此, 从模糊数学的角度, 求模糊关系的逆模型即根据给定的输出找到所有可能的输入是十分复杂的。不少书籍和文章中均给出了一些模糊关系求逆的方法。许多方法由于计算过于复杂而不适于自适应控制中的在线计算, 这里将不予介绍, 下面介绍两种较为简单实用的方法。

(1) 基于模糊关系定义的模糊模型求逆

仍考虑上面的一阶模型例子, 对于正向模型有

$$\mu_{R_i}(y, u, \dot{y}) = \min\{\mu_{A^i}(y), \mu_{B^i}(u), \mu_{C^i}(\dot{y})\}$$

对于逆模型, 则有

$$\mu_{R_i^{-1}}(y, \dot{y}, u) = \min\{\mu_{A^i}(y), \mu_{C^i}(\dot{y}), \mu_{B^i}(u)\}$$

比较上面两个式子可以看出, R 与 R^{-1} 的元素值是完全相同的, 只不过它们在其中的排列次序有所不同而已。

以上是根据模糊关系的定义面求得逆模糊关系。虽然从数学的角度它并不是非常严格的, 但是它具备简单和实用的优点, 适于自适应控制的在线计算。

例 2.20 若已知系统正向模型的模糊关系 R 为例 2.19 求得的所示。且已知 u 和 y 量化方法及隶属度函数均同例 2.19。现要求该系统的逆模糊关系 R^{-1} , 且利用 R^{-1} 计算当 $y=0, 1, 4, 9, 16, 25$ 时所对应的 u 值。

根据例 2.19, 已求得

$$R(u, y) = \begin{bmatrix} 1 & 0.5 & 0 & 0 & 0 & 0 \\ 1 & 0.5 & 0.5 & 0 & 0 & 0 \\ 0.5 & 1 & 0.5 & 0.5 & 0 & 0 \\ 0.5 & 0.5 & 1 & 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 1 & 0.5 & 0.5 \\ 0 & 0 & 0.5 & 0.5 & 0.5 & 1 \end{bmatrix}$$

则相应的逆模糊关系为

$$R^{-1}(y,u) = \begin{bmatrix} 1 & 1 & 0.5 & 0.5 & 0 & 0 \\ 0.5 & 0.5 & 1 & 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 1 & 0.5 & 0.5 \\ 0 & 0 & 0.5 & 0.5 & 1 & 0.5 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0.5 & 1 \end{bmatrix}$$

利用如下的模糊推理计算过程

$$y \xrightarrow{\text{量化}} y^* \xrightarrow{\text{模糊化}} Y^* \xrightarrow{\text{合成运算}} U^* = Y^* \circ R^{-1} \xrightarrow{\text{清晰化}} u$$

可以根据给定的 y 计算出 u 。这里量化采用最近的量化值 y^* 来替代 y ，模糊化采用单点模糊集合，清晰化采用最大隶属度法，所得结果如表 2.23 所示。可见它与实际值是非常接近的。

表 2.23 基于逆模型计算的结果

y	0	1	4	9	16	25
$u_{\text{实}}$	0	1	2	3	4	5
$u_{\text{估}}$	0.5	0.5	2	3	4	5

(2) 基于插值的模糊模型求逆

仍考虑前面给出的一阶系统模型，已知正向模型为

$$R = (y \times u) \rightarrow \dot{y}$$

现在已知 y 及 \dot{y} 求输入量 u 。这里介绍的插值法是不直接求逆模糊关系 R^{-1} ，而仍利用正模糊关系 R 进行正方向的推理计算。具体方法是：固定 y 不变，用不同的 u 作为输入量，利用正向模糊推理计算出不同的 \dot{y} 。例如令 $u = u_i$ ，计算出相应的 $\dot{y}_i (i=1, 2, \dots, n)$ 。若给定的 \dot{y} 位于 \dot{y}_i 内，则用内插法求出相应的 u ；若 \dot{y} 位于 \dot{y}_i 之外，则可用外推法求出相应的 u 。

例 2.21 若已知系统正向模型模糊关系 R 仍为例 2.19 求得的所示，且已知 u 和 y 的量化方法及隶属度函数均同例 2.19。现要求利用插值法计算当 $y=12$ 时所对应的 u 值。

根据例 2.19，已求得当 $u=0, 1, 2, 3, 4, 5$ 时相应的 y 值如表 2.22 和图 2.52 所示。现要求 $y=12$ ，采用线性内插可以求得 $u=3.43$ 。理论值 $u_{\text{理}} = \sqrt{12} = 3.46$ ，可见它们也是比较接近的。

上面介绍了两种模糊模型求逆的方法。显然第一种方法更为简单，它只需一次合成运算。而第二种方法需多次合成运算，然后还要进行插值计算。但第一种方法要求规则是完备的，否则对于未建模的区域，则可能导致完全错误的结果。而插值法对于未建模的区域则可通过外推法得到合适的结果。

例如，对于例 2.19，若缺少数据 $(u, y) = (5, 25)$ ，则求得

$$R(u, y) = \bigcup_{i=1}^5 R_i(u, y) = \begin{bmatrix} 1 & 0.5 & 0 & 0 & 0 & 0 \\ 1 & 0.5 & 0.5 & 0 & 0 & 0 \\ 0.5 & 1 & 0.5 & 0 & 0 & 0 \\ 0.5 & 0.5 & 1 & 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 1 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0.5 & 0 \end{bmatrix}$$

当 $y=25$ 时,按照第一种方法求得 $u=0$ (理论值应 $u=5$),该结果显然是非常错误的,其原因是在 $(u, y)=(5, 25)$ 的附近没有相应的规则。所以它不能获得正确推理结果。若按照第二种方法,参照图 2.52,采用线性外推可求得 $u=7.3$ 。这时虽与理论值有较大误差,但不是错误的结果。这时若采用二次函数外推可得 $u=5$,结果是非常准确的。

3. 控制器设计

在第 2.7.3 节中,结合自动小车的模糊控制介绍了基于语言模型求逆的模糊控制器设计方法。下面给出更加一般的结果,它是自适应模糊控制的一个重要组成部分。

暂时先不考虑自适应控制,画出一般的模糊控制系统如图 2.53 所示。问题是已知控制对象模型 R_p 及期望性能 R 设计模糊控制器。

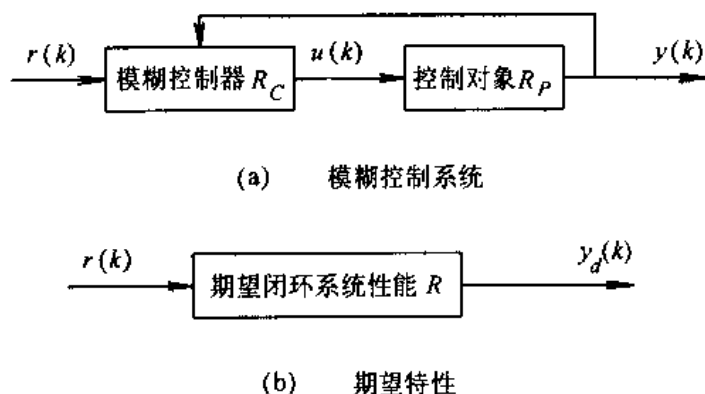


图 2.53 模糊控制系统及期望特性

设已知控制对象的模糊模型为

$$y(k+1) = [\bar{y}(k) \times \bar{u}(k-1) \times u(k)] \circ R_p$$

其中

$$\bar{y}(k) = [y(k) \quad y(k-1) \quad \cdots \quad y(k-n+1)]^T$$

$$\bar{u}(k-1) = [u(k-1) \quad u(k-2) \quad \cdots \quad u(k-m)]^T$$

这里考虑的是最一般的情况。很多情况下可能无 $\bar{u}(k-1)$ 项。

设已知期望的闭环系统性能为

$$y_d(k+1) = [\bar{r}(k) \times \bar{y}(k)] \circ R$$

其中 $\bar{r}(k) = [r(k) \quad r(k-1) \quad \cdots \quad r(k-n+1)]^T$, $r(k)$ 为参考输入。

模糊控制器的计算过程如下:

(1) 首先根据期望的闭环系统特性计算出期望的输出

$$y_d(k+1) = [\bar{r}(k) \times \bar{y}(k)] \circ R$$

(2) 根据控制对象的逆模型计算出控制量

$$\begin{aligned} u(k) &= \lfloor y_d(k+1) \times \bar{y}(k) \times \bar{u}(k-1) \rfloor \circ R_P^{-1} \\ &= \{ \lfloor \bar{r}(k) \times \bar{y}(k) \rfloor \circ R \times \bar{y}(k) \times \bar{u}(k-1) \} \circ R_P^{-1} \\ &= \lfloor \bar{r}(k) \times \bar{y}(k) \times \bar{u}(k-1) \rfloor \circ R_C \end{aligned}$$

可见这一步需要用到模糊模型的求逆,前面已对此进行了专门的讨论,并给出了两种简单实用的方法。

图 2.54 形象地画出了模糊控制器的计算过程。

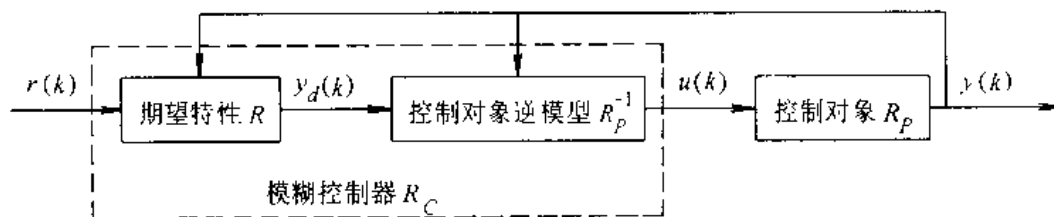


图 2.54 模糊控制器的计算过程

上面以离散模型为例介绍了模糊控制器的计算过程。对于连续模型其计算过程是类似的。例如对于如第 2.7.3 节的自动小车的模糊控制,小车的动力学模型为

$$\dot{y} = (y \times \dot{y} \times u) \circ R_P$$

期望的闭环特性为

$$\ddot{y}_d = (y \times \dot{y}) \circ R$$

根据小车动力学模型的逆模型可以求得模糊控制器为

$$\begin{aligned} u &= (y \times \dot{y} \times \ddot{y}_d) \circ R_P^{-1} = \lfloor y \times \dot{y} \times (y \times \dot{y}) \circ R \rfloor \circ R_P^{-1} \\ &= (y \times \dot{y}) \circ R_C \end{aligned}$$

4. 自适应模糊控制器

将上面讨论的在线模糊模型辨识与控制器设计组合在一起便构成如图 2.48 所示的间接自适应模糊控制。结合图 2.54 可以画出较为具体的结构如图 2.55 所示。

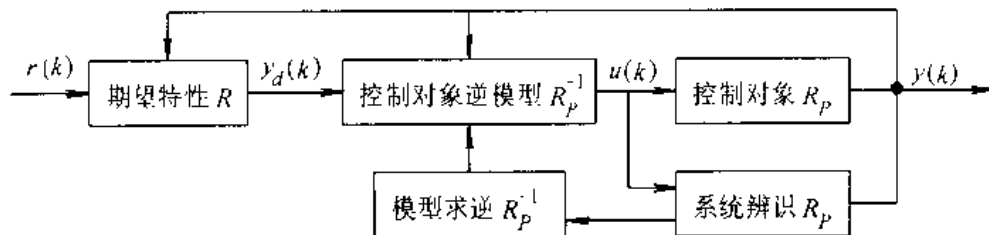


图 2.55 间接自适应模糊控制

在图 2.55 所示的自适应控制结构中包含了两个反馈回路:控制器回路和模型修正回路。控制器回路根据输出量反馈来确定所需的控制量 $u(k)$,以达到期望的系统性能 R 。模型修正回路利用输入输出数据来自适应地修正控制对象模型。为了降低模型对输出噪声的灵敏度,模型修正回路的时间常数必须选取得足够大,它可以通过调整遗忘因子 λ 来

实现。 λ 越大,修正回路的时间常数也越大,这里也有与常规的自适应控制相一致的要求:内回路的响应速度要远远快于外回路的参数调整速度。

两个反馈回路稳定性问题必须加以考虑。前面介绍了模型修正过程的稳定性问题:在一定的条件下,模型的收敛性是能够得到保证的。控制回路的稳定性直接受模型求逆及期望性能规则集的影响。控制器的设计是基于所建模型是准确的假设,因此期望特性的收敛性是影响控制回路稳定性的主要因素,它可以通过前面第 2.7.2 节所介绍的方法来进行分析。

在自适应模糊控制的实现过程中,还有一些问题需要具体加以考虑。

(1) 模型表示所需的存储量与输出精度的折中考虑。这里模糊模型采用模糊规则库及相应的模糊关系来表示,若每个变量量化的分辨率越高,则输出的精度也越高,同时它要求的存储容量也越大。因而两者是矛盾的。所以应适当选择量化的分级数。一个较好的解决方法是采用非线性的量化方法或者采用混合控制的方法。所谓混合控制的方法是指在远离平衡点时采用上述的自适应模糊控制,而在平衡点附近时采用常规的 PID 控制或分辨率更高的自适应模糊控制来进行更精细的调整。

(2) 性能关系矩阵的选取。性能关系矩阵的精度也直接影响控制变量的精度。通常期望性能关系矩阵 R 可以用一个低阶模型来表示,一般取为二阶,这样便于采用模糊相平面法来分析系统的动态响应性能和稳定性。

期望性能的给定必须保证在物理上是可实现的。这一点可通过在性能规则库中选取与状态变量相同的论域来保证。这样对于输出变量导数的任何要求都限制在实际范围。

(3) 模型求逆方法的选取。前面已经对此进行过讨论。基于模糊关系定义的方法要求正向模型的规则集比较完备。它比较适合于参数变化较小的情况。当起始规则很少甚至没有的情况下,基于插值法的求逆方法仍能给出较好的结果。

(4) 一步预报控制。间接自适应控制方法中,在一个采样周期内要在线地完成模型辨识及控制器设计的计算任务,因而可能产生较大的计算延时,即送出控制量比采集到输出量延迟一段时间。当该计算延时与采样周期相比不可忽略时,它使系统的性能明显变差。这时可采用一步预报的控制方式,即首先预报出下一时刻测量值,根据该预报测量值事先计算出下一时刻控制量。这样当到达下一采样时刻时,可立即将控制量送出。

参 考 文 献

- [1] Lee C C. Fuzzy Logic in Control Systems: Fuzzy Logic Controller IEEE Trans. on SMC, 1990, 20(2)
- [2] [日]水本雅晴著,刘凤璞等编译. 模糊数学及其应用. 科学出版社,1988
- [3] 王学慧,田成方. 微机模糊控制理论及其应用. 电子工业出版社,1987
- [4] 李友善,李军. 模糊控制理论及其在过程控制中的应用. 国防工业出版社,1993
- [5] 李上勇,夏承光. 模糊控制和智能控制理论与应用. 哈尔滨工业大学出版社,1990
- [6] Peng W. Analysis and Synthesis is of Fuzzy Intelligent Control Systems. Ph. D Thesis, Hong Kong Polytechnic, 1993

- [7] Harris C J, Moore C G, Brown M. Intelligent Control-Aspects of Fuzzy Logic and Neural Nets. World Scientific, 1993
- [8] Tanaka K, Sugeno M. Stability Analysis and Design of Fuzzy Control Systems. Fuzzy Sets and Systems, 1992, 45
- [9] S. Shao. Fuzzy Self-Organizing Controller and Its Application for Dynamic Processes. Fuzzy Sets and Systems, 1988, 26
- [10] Tanaka K. Stability and Stabilizability of Fuzzy-Neural-Linear Control systems. IEEE Tran. on Fuzzy Systems, 1995, 3(4)
- [11] Feng G, Cao S G *et al.* Design of Fuzzy Control Systems with Guaranteed Stability. Fuzzy Sets and Systems (to appear).

第3章 神经网络控制

本章主要讨论人工神经网络(以后简称神经网络)在系统建模及控制方面的应用。神经网络控制是一种基本上不依赖于模型的控制方法,它比较适用于那些具有不确定性或高度非线性的控制对象,并具有较强的适应和学习功能,因而神经网络控制是智能控制的一个重要分支领域。

本章首先介绍几种可用于控制的神经网络模型,然后介绍它们在系统建模及控制中的应用。最后专门介绍它在机器人控制中的应用。

3.1 概 述

3.1.1 神经元模型

1. 生物神经元模型

人脑是由大量的神经细胞组合而成的,它们之间互相连接。每个神经细胞(也称神经元)具有如图 3.1 所示的结构。

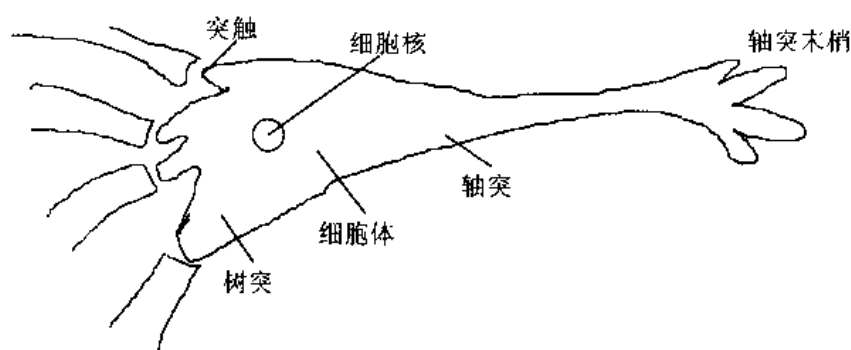


图 3.1 生物神经元模型

由图看出,脑神经元由细胞体、树突和轴突构成。细胞体是神经元的中心,它一般又由细胞核、细胞膜等组成。树突是神经元的主要接受器,它主要用来接受信息。轴突的作用主要是传导信息,它将信息从轴突起点传到轴突末梢,轴突末梢与另一个神经元的树突或细胞体构成一种突触的机构。通过突触实现神经元之间的信息传递。

2. 人工神经元模型

人工神经网络是利用物理器件来模拟生物神经网络的某些结构和功能。图 3.2 是最典型的人工神经元模型。

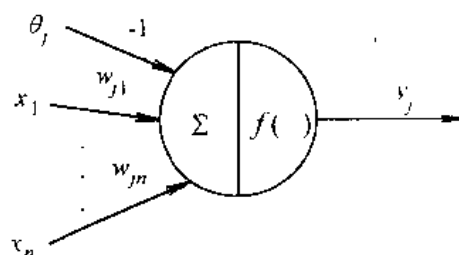


图 3.2 人工神经元模型

该神经元模型的输入输出关系为

$$s_j = \sum_{i=1}^n w_{ji} x_i - \theta_j = \sum_{i=0}^n w_{ji} x_i \quad (x_0 = \theta_j, w_{j0} = -1)$$

$$y_j = f(s_j)$$

其中 θ_j 称为阈值, w_{ji} 称为连接权系数, $f(\cdot)$ 称为输出变换函数, 图 3.3 表示了几种常见的变换函数。

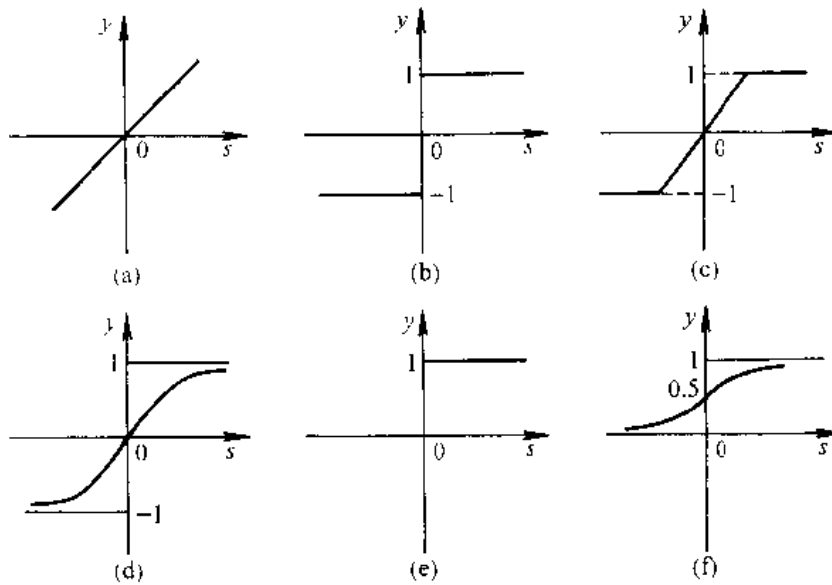


图 3.3 常见的变换函数

在图 3.3 中, 各变换函数的解析表达式分别为

(a) 比例函数

$$y = f(s) = s$$

(b) 符号函数

$$y = f(s) = \begin{cases} 1 & s \geq 0 \\ -1 & s < 0 \end{cases}$$

(c) 饱和函数

$$y = f(s) = \begin{cases} 1 & s \geq \frac{1}{k} \\ ks & -\frac{1}{k} \leq s < \frac{1}{k} \\ -1 & s < -\frac{1}{k} \end{cases}$$

(d) 双曲函数

$$y = f(s) = \frac{1 - e^{-\mu s}}{1 + e^{-\mu s}}$$

(e) 阶跃函数

$$y = f(s) = \begin{cases} 1 & s \geq 0 \\ 0 & s < 0 \end{cases}$$

(f) S 形函数

$$y = f(s) = \frac{1}{1 + e^{-s}}$$

3.1.2 人工神经网络

人工神经网络是一个并行和分布式的信息处理网络结构,该网络结构一般由许多个神经元组成,每个神经元有一个单一的输出,它可以连接到很多其它的神元,其输入有多个连接通路,每个连接通路对应一个连接权系数。

严格说来,神经网络是一个具有如下性质的有向图。

- (1) 对于每个结点有一个状态变量 x_i ;
- (2) 结点 i 到结点 j 有一个连接权系数 w_{ji} ;
- (3) 对于每个结点有一个阈值 θ_j ;

(4) 对于每个结点定义一个变换函数 $f_j[x_i, w_{ji}, \theta_j (i \neq j)]$,最常见的情形为 $f(\sum_i w_{ji} x_i - \theta_j)$ 。

图 3.4 表示了两个典型的神经网络结构,图 3.4(a)为前馈型网络,图 3.4(b)为反馈型网络。

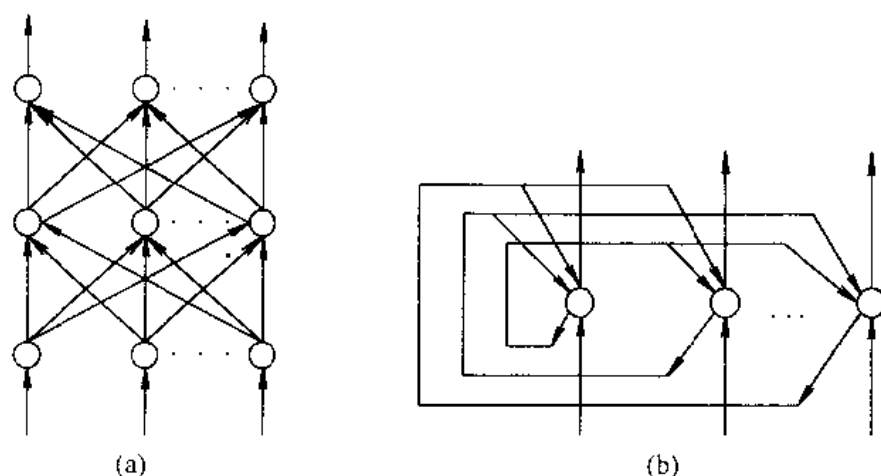


图 3.4 典型的神经网络结构

人工神经网络是生物神经网络的一种模拟和近似。它主要从两个方面进行模拟:一种是从结构和实现机理方面进行模拟,它涉及到生物学、生理学、心理学、物理及化学等许多基础学科。由于生物神经网络的结构和机理相当复杂,现在距离完全认识它们还相差甚远;另外一种是从功能上加以模拟,即尽量使得人工神经网络具有生物神经网络的某些功能特性,如学习、识别、控制等功能。本书着重于后者,主要介绍几种典型的人工神经网络模型,并重点研究它们在系统建模及控制方面的应用。

3.1.3 生物神经网络系统与计算机处理信息的比较

计算机具有快速计算的能力,这是人所远远不能比拟的,而人的学习、决策和识别等方面的能力则远远超过计算机。下面通过对这两者在处理信息方面的比较来说明产生这些差别的原因。

1. 处理速度

计算机处理单个信息的时间约为 ns 级(例如 Cray 计算机约为 4.2ns)。而脑神经元对外部激励的响应时间大约在 ms 级。可见,计算机处理单个信息的时间要比人脑大约快 10^6 倍。

2. 处理顺序

虽然计算机处理单个信息的速度比人脑快很多,但对有些问题(如识别、决策等),计算机却没有人脑快。其根本的原因在于计算机处理信息的顺序是串行的,而人脑处理信息是并行的。正是由于这一点,才使得人脑具有很强的综合处理信息的能力。

3. 处理单元的数目及复杂程度

人脑是一个十分复杂的生物组织,据估计它大约具有 10^{11} — 10^{14} 个神经元,每个神经元大约与 10^3 — 10^4 个其它神经元相连接。如果设想脑神经系统是人工神经网络的最终的模拟对象,那么便需要数量非常巨大的人工神经元处理单元才能使人工神经网络具有类似人脑的高级信息处理的功能。

除了脑神经元的数量十分巨大外,单个脑神经元的构造也是很复杂的,它远远不是如前面所述人工神经元模型所实现的那种简单关系。

4. 知识存储

在计算机中,知识是静态地存储在编有地址的记忆单元中,新的信息破坏老的信息。而在人脑中,知识存储在神经元之间的连接关系中,新的知识用来调整这种连接关系,而不是破坏这种连接关系。一句话,知识在人脑中具有适应性,而在计算机中只是严格的替换关系。由此可以解释为什么人脑具有综合概括的能力,而计算机却不具备。

5. 容错能力

人脑具备较好的容错能力,个别神经元的损坏并不影响整体的性能。而通常的计算机却不具备容错能力,CPU 或存储器的损坏都将导致整体系统的实质性破坏。

6. 运行控制

在计算机中有一个中央处理单元来控制所有的活动和对所有的信息进行存取操作,它实质上产生了信息处理的一个瓶颈,同时也使得一旦控制部件产生故障而导致整个系统的失效。而在脑神经系统中,不存在这样的中央控制单元来控制每一个神经元的活动。每个神经元只受与它相联结的一部分神经元的影响,而不受其它部分神经元的控制和影响。

3.1.4 神经网络的发展概况

1943 年心理学家 W. McCulloch 和数理逻辑学家 W. Pitts 首先提出了一个简单的神经网络模型,其神经元的输入输出关系为

$$y_j = \text{Sgn}(\sum_i w_{ji} x_i - \theta_j)$$

其中输入、输出均为二值量, w_{ji} 为固定的权值。利用该简单网络可以实现一些逻辑关系。虽然该模型过于简单, 但它为进一步的研究打下了基础。

1949 年 D. O. Hebb 首先提出了一种调整神经网络连接权的规则, 通常称为 Hebb 学习规则。其基本思想是, 当两个神经元同时兴奋或同时抑制时, 则它们之间的连接强度便增加。用式子表示即为

$$w_{ij} = \begin{cases} \sum_{k=1}^n x_i^{(k)} x_j^{(k)} & i \neq j \\ 0 & i = j \end{cases}$$

或者 $w_{ij}(k+1) = w_{ij}(k) + x_i^{(k+1)} x_j^{(k+1)}$ 。该学习规则的意义为, 连接权的调整正比于两个神经元活动状态的乘积, 连接权是对称的, 神经元到自身的连接权为零。现在仍有不少神经网络采用这样的学习规则。

1958 年 F. Rosenblatt 等人研究了一种特殊类型的神经网络, 称为“感知机”(perceptron)。他们认为这是生物系统感知外界传感信息的简化模型。该模型主要用于模式分类, 并一度引起人们的广泛兴趣。

1969 年 M. Minsky 和 S. Papert 发表了名为“感知机”的专著。他们在这本专著中指出了简单的线性感知机的功能是有限的, 它无法解决线性不可分的两类样本的分类问题。典型的例子如“异或”计算, 即简单的线性感知机不可能实现“异或”的逻辑关系。要解决这个问题, 必须加入隐层结点。但是对于多层网络, 如何找到有效的学习算法尚是难于解决的问题。因此它使得整个 70 年代神经网络的研究处于低潮。

美国物理学家 J. J. Hopfield 在 1982 和 1984 年发表了两篇神经网络的文章, 引起了很大的反响。他提出了一种反馈互连网, 并定义了一个能量函数, 它是神经元的状态和连接权的函数, 利用该网络可以求解联想记忆和优化计算的问题。该网络后来称为 Hopfield 网, 最典型的例子是应用该网络成功地求解了旅行商最优路径问题。

1986 年 D. E. Rumelhart 和 J. L. McClelland 等人提出了多层前馈网的反向传播算法 (Back Propagation), 以后简称 BP 网络或 BP 算法。该算法解决了感知机所不能解决的问题。

Hopfield 网和反向传播算法的提出使人们看到了神经元网络的前景和希望。1987 年在美国召开了第一届国际神经网络会议, 它掀起了神经网络研究的热潮, 许多研究人员都企图找到神经网络在各自领域的应用。

神经网络控制也是从这个背景下发展起来的, 自 80 年代后期以来, 神经网络控制已取得很大进展, 本章着重介绍在这方面的研究工作和主要成果。

3.2 前馈神经网络

对于如图 3.4 所示的神经网络, 它具有分层的结构。最下面一层是输入层, 中间是隐层, 最上面一层是输出层。其信息从输入层依次向上传递, 直至输出层。这样的结构称为

前馈网络。这是神经网络中的一种典型结构。

3.2.1 感知器网络

感知器是最简单的前馈网络,它主要用于模式分类。也可用在基于模式分类的学习控制和多模态控制中。

1. 单层感知器网络

图 3.5 所示为单层的感知器网络结构。

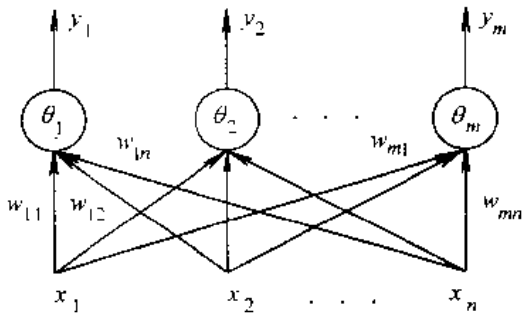


图 3.5 单层感知器网络

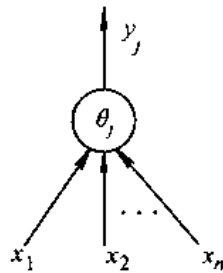


图 3.6 单个神经元的感知器

图中 $\mathbf{x}=[x_1 \ x_2 \cdots x_n]^T$ 是输入特征向量, w_{jn} 是 x_i 到 y_j 的连接权, 输出量 y_j ($j=1, 2, \cdots, m$) 是按照不同特征的分类结果。由于按不同特征的分类是互相独立的, 因而可以取出其中的一个神经元来讨论, 如图 3.6 所示。其输入到输出的变换关系为

$$s_j = \sum_{i=1}^n w_{ji} x_i - \theta_j$$

$$y_j = f(s_j) = \begin{cases} 1 & s_j \geq 0 \\ -1 & s_j < 0 \end{cases}$$

若有 P 个输入样本 x^p ($p=1, 2, \cdots, P$), 经过该感知器的输出 y_j 只有两种可能, 即 1 和 -1, 从而说明它将输入模式分成了两类。若将 x^p ($p=1, 2, \cdots, P$) 看成是 n 维空间的 P 个点, 则该感知器将该 P 个点分成了两类, 它们分属于 n 维空间的两个不同的部分。

为便于说明, 下面以二维空间为例, 如图 3.7 所示, 设图中的“。”和“×”表示输入的

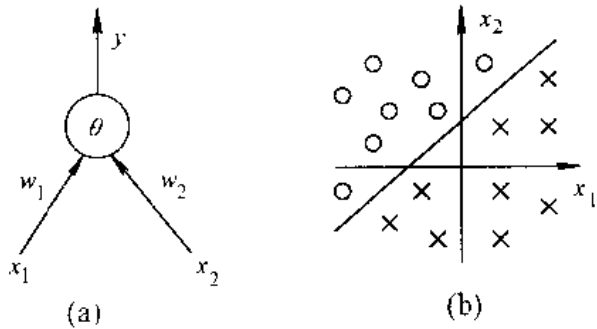


图 3.7 两维输入的感知器

特征向量点,其中“。”和“×”表示具有不同特征的两类向量。现在要求用单个神经元感知器将其分类。

根据感知器的变换关系,可知分界线的方程为

$$w_1x_1 + w_2x_2 - \theta = 0$$

显然,这是一条直线方程。它说明,只有那些线性可分模式类才能用感知器来加以区分。如图 3.8 所示的异或关系,显然它是线性不可分的。因此单层感知器不可能将其正确分类。历史上,Minsky 正是利用这个典型例子指出了感知器的致命弱点,从而导致了 70 年代神经元网络的研究低潮。

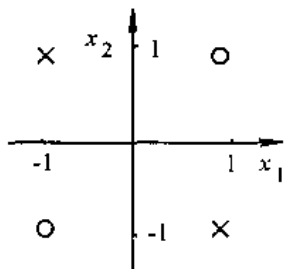


图 3.8 异或关系的线性不可分性

从图 3.7 可以看出,若输入模式是线性可分的,则可以找到无穷多条直线来对其进行正确的分类。现在的问题是,如果已知一组输入样本模式以及它们所属的特征类,如何找出其中一条分界线能够对它们进行正确的分类。对于一般情况其问题可描述为,已知输入输出样本 x_p 和 d_p ($p=1,2,\dots,P$),这里 x_p 和 d_p 表示第 p 组输入向量和期望的输出。问题是如何设计感知器网络的连接权 w_i ($i=1,2,\dots,n$) 和 θ ,以使该网络能实现正确的分类。也就是说,如何根据样本对连接权和阈值进行学习和调整。这里样本相当于“教师”,所以这是一个有监督的学习问题。下面给出一种学习算法。

(1) 随机地给定一组连接权 $w_i(0)$, $k=0$;

(2) 任取其中一组样本 x_p 和 d_p , 计算

$$s = \sum_{i=0}^n w_i x_{pi} \quad (\text{设取 } x_{p0} = 1, w_0 = -\theta)$$

$$y_p = f(s) = \begin{cases} 1 & s \geq 0 \\ -1 & s < 0 \end{cases}$$

(3) 按下式调整连接权

$$w_i(k+1) = w_i(k) + \alpha(d_p - y_p)x_{pi} \quad i = 1, 2, \dots, n$$

其中取 $\alpha > 0$, α 称为学习率。

(4) 在样本集中选取另外一组样本,并让 $k+1 \rightarrow k$, 重复上述(2)~(4)的过程,直到

$$w_i(k+1) = w_i(k) \quad i = 1, 2, \dots, n$$

可以证明,该学习算法收敛的充分必要条件是输入样本是线性可分的。同时学习率 α 的选取也是十分关键的。 α 选取太小,学习太慢; α 太大,学习过程可能出现修正过头的情况,从而产生振荡。

2. 多层感知器网络

根据上面讨论,对于如图 3.8 所示的线性不可分的输入模式,只用单层感知器网络不可能对其实现正确的区分,这时可采用如图 3.9 所示的多层感

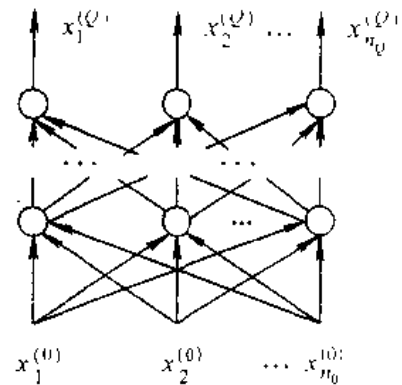


图 3.9 多层感知器网络

知器网络。其中第 0 层为输入层,有 n_0 个神经元;第 Q 层为输出层,有 n_Q 个输出,中间层为隐层。该多层感知器网络的输入输出变换关系为

$$s_i^{(q)} = \sum_{j=0}^{n_{q-1}} w_{ij}^{(q)} x_j^{(q-1)} \quad (x_0^{(q-1)} = \theta_i^{(q)}, w_{i0}^{(q)} = -1)$$

$$x_i^{(q)} = f(s_i^{(q)}) = \begin{cases} 1 & s_i^{(q)} \geq 0 \\ -1 & s_i^{(q)} < 0 \end{cases}$$

$$i = 1, 2, \dots, n_q \quad j = 1, 2, \dots, n_{q-1} \quad q = 1, 2, \dots, Q$$

这时每一层相当于一个单层感知器网络,如对于第 q 层,它形成一个 $n_q - 1$ 维的超平面,它对于该层的输入模式进行线性分类,但是由于多层的组合,最终可实现对输入模式的较复杂的分类。

例如对于如图 3.8 所示的异或关系,可采用如图 3.10 所示的多层感知器网络来实现对它的正确分类。具体做法是:

(1) 利用上述学习算法,设计连接权系数 $w_{11}^{(1)}$ 和 $w_{12}^{(1)}$,以使得其分界线为图 3.11(a) 中的 L_1 。即 L_1 的直线方程为

$$w_{11}^{(1)} x_1^{(0)} + w_{12}^{(1)} x_2^{(0)} - \theta_1^{(1)} = 0$$

且相应于 P_2 的输出为 1,相应于 P_1 和 P_3 和 P_4 的输出为 -1。

(2) 设计连接权系数 $w_{21}^{(1)}$ 和 $w_{22}^{(1)}$,以使得其分界线为图 3.11(a) 中的 L_2 ,且使得相应于 P_1, P_2 和 P_3 的输出为 1,相应于 P_4 的输出为 -1。

(3) 在 $x_1^{(1)}$ 和 $x_2^{(1)}$ 平面中(见图 3.11(b)),这时只有三个点 Q_1, Q_2 和 Q_3 ,括弧中标出了所对应的第一层的输入模式。 Q_1, Q_2 和 Q_3 是第二层(即神经元 $x^{(2)}$)的输入模式。现在只要设计连接权系数 $w_1^{(2)}$ 和 $w_2^{(2)}$,以使得其分界线为图 3.11(b) 中的 L_3 ,即可将 Q_2 与

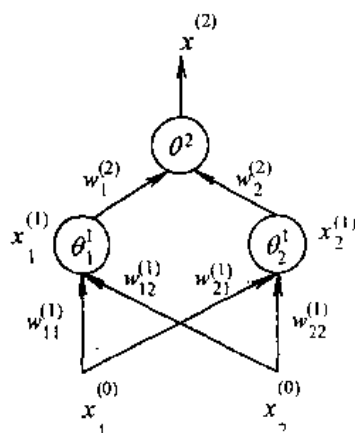


图 3.10 实现异或关系的多层感知器网络

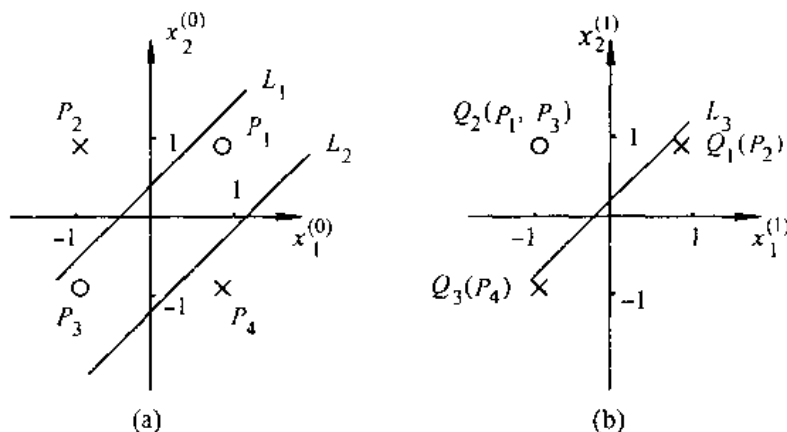


图 3.11 实现异或关系多层感知器网络的模式类划分

(Q_1, Q_2) 区分开来,也即将 (P_1, P_3) 与 (P_2, P_4) 区分开来,从而正确地实现了异或关系。

可见,适当地设计多层感知器网络可以实现任意形状的划分。

3.2.2 BP 网络

前面介绍的感知器网络中神经元的变换函数采用的是如图 3.3(b)所示的符号函数,因此输出的是二值量。它主要用于模式分类。这里所要介绍的多层前馈网具有如图 3.9 相同的结构,这时神经元的变换函数采用如图 3.3(f)的 S 型函数,因此输出量是 0 到 1 之间的连续量,它可实现从输入到输出的任意的非线性映射。由于连接权的调整采用的是反向传播(Back Propagation)的学习算法,因此该网络也称为 BP 网络。

在图 3.9 所示的多层前馈网络中,第 1 层为输入层,第 Q 层为输出层,中间各层为隐层。设第 q 层($q=1,2,\dots,Q$)的神经元个数为 n_q ,输入到第 q 层的第 i 个神经元的连接权系数为 $w_{ij}^{(q)}$ ($i=1,2,\dots,n_q; j=1,2,\dots,n_{q-1}$)。该网络的输入输出变换关系为

$$s_i^{(q)} = \sum_{j=0}^{n_{q-1}} w_{ij}^{(q)} x_j^{(q-1)} \quad (x_0^{(q-1)} = \theta_i^{(q)}, w_{i0}^{(q)} = -1)$$

$$x_i^{(q)} = f(s_i^{(q)}) = \frac{1}{1 + e^{-w_i^{(q)}}}$$

$$i = 1, 2, \dots, n_q \quad j = 1, 2, \dots, n_{q-1} \quad q = 1, 2, \dots, Q$$

设给定 P 组输入输出样本 $\mathbf{x}_p^{(0)} = [x_{p1}^{(0)} \ x_{p2}^{(0)} \ \dots \ x_{p,n_0}^{(0)}]^T$, $\mathbf{d}_p = [d_{p1} \ d_{p2} \ \dots \ d_{p,n_Q}]^T$ ($p=1, 2, \dots, P$) 利用该样本集首先对 BP 网络进行训练,也即对网络的连接权系数进行学习和调整,以使该网络实现给定的输入输出映射关系。经过训练的 BP 网络,对于不是样本集中的输入也能给出合适的输出。该性质称为泛化(generalization)功能。从函数拟合的角度,它说明 BP 网络具有插值功能。下面介绍连接权系数的学习方法。

设取拟合误差的代价函数为

$$E = \frac{1}{2} \sum_{p=1}^P \sum_{i=1}^{n_Q} (d_{pi} - x_{pi}^{(Q)})^2 = \sum_{p=1}^P E_p$$

即

$$E_p = \frac{1}{2} \sum_{i=1}^{n_Q} (d_{pi} - x_{pi}^{(Q)})^2$$

问题是如何调整连接权系数以使代价函数 E 最小。优化计算的方法很多,比较典型的是采用一阶梯度法,即最速下降法。下面具体介绍这种方法。

一阶梯度法寻优的关键是计算优化目标函数(即本问题中的误差代价函数)E 对寻优参数的一阶导数。下面我们从输出层开始来依次计算

$$\partial E / \partial w_{ij}^{(q)} \quad (q = Q, Q-1, \dots, 1)$$

由于

$$\frac{\partial E}{\partial w_{ij}^{(q)}} = \sum_{p=1}^P \frac{\partial E_p}{\partial w_{ij}^{(q)}}$$

所以下面着重讨论 $\partial E_p / \partial w_{ij}^{(q)}$ 的计算

对于第 Q 层有

$$\frac{\partial E_p}{\partial w_{ij}^{(Q)}} = \frac{\partial E_p}{\partial x_{pi}^{(Q)}} \frac{\partial x_{pi}^{(Q)}}{\partial s_{pi}^{(Q)}} \frac{\partial s_{pi}^{(Q)}}{\partial w_{ij}^{(Q)}} = - (d_{pi} - x_{pi}^{(Q)}) f'(s_{pi}^{(Q)}) x_{pj}^{(Q-1)} = - \delta_{pi}^{(Q)} x_{pj}^{(Q-1)}$$

其中

$$\delta_{pi}^{(Q)} = - \frac{\partial E_p}{\partial x_{pi}^{(Q)}} = (d_{pi} - x_{pi}^{(Q)}) f'(s_{pi}^{(Q)})$$

$x_{pi}^{(Q)}$, $s_{pi}^{(Q)}$ 及 $x_{pj}^{(Q-1)}$ 表示利用第 p 组输入样本所算得的结果。

对于第 $Q-1$ 层有

$$\begin{aligned} \frac{\partial E_p}{\partial w_{ij}^{(Q-1)}} &= \frac{\partial E_p}{\partial x_{pi}^{(Q-1)}} \frac{\partial x_{pi}^{(Q-1)}}{\partial w_{ij}^{(Q-1)}} = \left(\sum_{k=1}^{n_Q} \frac{\partial E_p}{\partial s_{pk}^{(Q)}} \frac{\partial s_{pk}^{(Q)}}{\partial x_{pi}^{(Q-1)}} \right) \frac{\partial x_{pi}^{(Q-1)}}{\partial s_{pi}^{(Q-1)}} \frac{\partial s_{pi}^{(Q-1)}}{\partial w_{ij}^{(Q-1)}} \\ &= \left(\sum_{k=1}^{n_Q} - \delta_{pk}^{(Q)} w_{ki}^{(Q)} \right) f'(s_{pi}^{(Q-1)}) x_{pj}^{(Q-2)} = - \delta_{pi}^{(Q-1)} x_{pj}^{(Q-2)} \end{aligned}$$

其中

$$\delta_{pi}^{(Q-1)} = - \frac{\partial E_p}{\partial x_{pi}^{(Q-1)}} = \left(\sum_{k=1}^{n_Q} \delta_{pk}^{(Q)} w_{ki}^{(Q)} \right) f'(s_{pi}^{(Q-1)})$$

显然,它是反向递推计算的公式,即首先计算出 $\delta_{pk}^{(Q)}$,然后再由上式递推计算出 $\delta_{pi}^{(Q-1)}$ 。依次类推,可继续反向递推计算出 $\delta_{pi}^{(q)}$ 和 $\partial E_p / \partial w_{ij}^{(q)}$, ($q=Q-2, \dots, 1$)。从上式看出,在 $\delta_{pi}^{(q)}$ 的表达式中包含了导数项 $f'(s_{pi}^{(q)})$, 由于假定 $f(\cdot)$ 为 S 形函数,所以其导数可求得如下:

$$\begin{aligned} x_{pi}^{(q)} &= f(s_{pi}^{(q)}) = \frac{1}{1 + e^{-\mu s_{pi}^{(q)}}} \\ f'(s_{pi}^{(q)}) &= \frac{\mu e^{-\mu s_{pi}^{(q)}}}{(1 + e^{-\mu s_{pi}^{(q)}})^2} = \mu f(s_{pi}^{(q)}) [1 - f(s_{pi}^{(q)})] = \mu x_{pi}^{(q)} (1 - x_{pi}^{(q)}) \end{aligned}$$

最后可归纳出 BP 网络的学习算法如下:

$$W_{ij}^{(q)}(k+1) = w_{ij}^{(q)}(k) + \alpha D_{ij}^{(q)}(k+1), \quad \alpha > 0$$

$$D_{ij}^{(q)} = \sum_{p=1}^P \delta_{pi}^{(q)} x_{pj}^{(q-1)}$$

$$\delta_{pi}^{(q)} = \left(\sum_{k=1}^{n_{q+1}} \delta_{pk}^{(q+1)} w_{ki}^{(q+1)} \right) \mu x_{pi}^{(q)} (1 - x_{pi}^{(q)})$$

$$\delta_{pi}^{(Q)} = (d_{pi} - x_{pi}^{(Q)}) \mu x_{pi}^{(Q)} (1 - x_{pi}^{(Q)})$$

$$q = Q, Q-1, \dots, 1 \quad i = 1, 2, \dots, n_q \quad j = 1, 2, \dots, n_{q-1}$$

由于该算法是反向递推(Back Propagation)计算的,因而通常称该多层前馈网络为 BP 网络。该网络实质上是对任意非线性映射关系的一种逼近,由于采用的是全局逼近的方法,因而 BP 网络具有较好的泛化能力。

从以上的讨论看出,对于给定的样本集,目标函数 E 是全体连接权系数 $w_{ij}^{(q)}$ 的函数。因此,要寻优的参数 $w_{ij}^{(q)}$ 个数比较多,也就是说,目标函数 E 是关于连接权的一个非常复杂的超曲面,这就给寻优计算带来一系列的问题。其中一个最大的问题是收敛速度慢。由于待寻优的参数太多,必然导致收敛速度慢的缺点。第二个严重缺陷是局部极值问题,即 E 的超曲面可能存在多个极值点。按照上面的寻优算法,它一般收敛到初值附近的局部极值。

总括起来,BP 网络的主要优点是:

- 只要有足够多的隐层和隐结点,BP 网络可以逼近任意的非线性映射关系;
- BP 网络的学习算法属于全局逼近的方法,因而它具有较好的泛化能力。

它的主要缺点是:

- 收敛速度慢;
- 局部极值;
- 难以确定隐层和隐结点的个数。

从原理上,只要有足够多的隐层和隐结点,即可实现复杂的映射关系,但是如何根据特定的问题来具体确定网络的结构尚无很好的方法,仍需要凭借经验和试凑。

BP 网络能够实现输入输出的非线性映射关系,但它并不依赖于模型。其输入与输出之间的关联信息分布地存储于连接权中。由于连接权的个数很多,个别神经元的损坏只对输入输出关系有较小的影响,因此 BP 网络显示了较好的容错性。

BP 网络由于其很好的逼近非线性映射的能力,因而它可应用于信息处理、图象识别、模型辨识、系统控制等多个方面。对于控制方面的应用,其很好的逼近特性和泛化能力是一个很好的性质。而收敛速度慢却是一个很大的缺点,这一点难以满足具有适应功能的实时控制的要求。

3.2.3 BP 网络学习算法的改进

前面提到,BP 网络的一个严重的缺点是收敛太慢,它影响了该网络在许多方面的实际应用。为此,许多人对 BP 网络的学习算法进行了广泛的研究,提出了许多改进的算法,下面介绍典型的几种。

1. 引入动量项

上述标准 BP 算法实质上是一种简单的最速下降静态寻优算法,在修正 $w(k)$ 时,只是按 k 时刻的负梯度方式进行修正,而没有考虑以前积累的经验,即以前时刻的梯度方向,从而常常使学习过程发生振荡,收敛缓慢。为此,有人提出了如下的改进算法。

$$w(k+1) = w(k) + \alpha[(1-\eta)D(k) + \eta D(k-1)]$$

其中 $w(k)$ 既可表示单个的连接权系数,也可表示连接权向量(其元素为连接权系数)。 $D(k) = -\partial E / \partial w(k)$ 为 k 时刻的负梯度。 $D(k-1)$ 是 $k-1$ 时刻的负梯度。 α 为学习率, $\alpha > 0$ 。 η 为动量项因子, $0 \leq \eta < 1$ 。

该方法所加入的动量项实质上相当于阻尼项,它减小了学习过程的振荡趋势,改善了收敛性,这是目前应用比较广泛的一种改进算法。

2. 变尺度法

标准的 BP 学习算法所采用的是一阶梯度法,因而收敛较慢。若采用二阶梯度法,则可以大大改善收敛性。二阶梯度法的算法为

$$w(k+1) = w(k) - \alpha[\nabla^2 E(k)]^{-1} \nabla E(k)$$

其中

$$\nabla E(k) = \frac{\partial E}{\partial w(k)} \quad \nabla^2 E(k) = \frac{\partial^2 E}{\partial w^2(k)} \quad 0 < \alpha \leq 1$$

虽然二阶梯度法具有比较好的收敛性,但是它需要计算 E 对 w 的二阶导数,这个计算量是很大的。所以一般不直接采用二阶梯度法,而常常采用变尺度法或共轭梯度法,它们具有如二阶梯度法收敛较快的优点,而又无需直接计算二阶梯度。下面具体给出变尺度法的算法。

$$w(k+1) = w(k) + \alpha H(k)D(k)$$

$$H(k) = H(k-1) - \frac{\Delta w(k)\Delta w^T(k)}{\Delta w^T(k)\Delta D(k)} - \frac{H(k-1)\Delta D(k)\Delta D^T(k)H(k-1)}{\Delta D^T(k)H(k-1)\Delta D(k)}$$

$$\Delta w(k) = w(k) - w(k-1)$$

$$\Delta D(k) = D(k) - D(k-1)$$

3. 变步长法

一阶梯度法寻优收敛较慢的一个重要原因是 α (学习率) 不好选择。 α 选得太小,收敛太慢,若 α 选得太大,则有可能修正过头,导致振荡甚至发散。下面给出的变步长法即是针对这个问题而提出的。

$$w(k+1) = w(k) + \alpha(k)D(k)$$

$$\alpha(k) = 2^\lambda \alpha(k-1)$$

$$\lambda = \text{sgn}[D(k)D(k-1)]$$

这里 w 表示某个连接权系数。上面的算法说明,当连续两次迭代其梯度方向相同时,表明下降太慢,这时可使步长加倍;当连续两次迭代其梯度方向相反时,表明下降过头,这时可使步长减半。当需要引入动量项时,上述算法的第二项可修改为

$$w(k+1) = w(k) + \alpha(k)[(1-\eta)D(k) + \eta D(k-1)]$$

在使用该算法时,由于步长在迭代过程中自适应进行调整,因此对于不同的连接权系数实际采用了不同的学习率,也就是说误差代价函数 E 在超曲面上在不同的方向按照各自比较合理的步长向极小点逼近。

3.2.4 神经网络的训练

前面介绍了感知器和 BP 两种前馈型网络,它们主要用于模式分类和函数估计,它们可应用于许多方面。基于神经网络的系统建模和控制是其中的一个重要方面。

神经网络主要有以下一些特点。

(1) 具有自适应功能

它主要是根据所提供的数据,通过学习和训练,找出和输出之间的内在联系,从而求得问题的解答,而不是依靠对问题的先验知识和规则,因而它具有很好的适应性。

(2) 具有泛化功能

它能够处理那些未经训练过的数据,而获得相应于这些数据的合适的解答。同样,它能够处理那些有噪声或不完全的数据,从而显示了很好的容错能力。对于许多实际问题来说,泛化能力是非常有用的,因为现实世界所获得的数据常常受到噪声的污染或残缺不全。

(3) 非线性映射功能

现实的问题常常是非常复杂的,各个因数之间互相影响,呈现出复杂的非线性关系,

神经网络为处理这些问题提供了有用的工具。

(4) 高度并行处理

神经网络的处理是高度并行的,因此用硬件实现的神经网络的处理速度可远远高于通常计算机的处理速度。

与常规的计算机程序相比较,神经网络主要基于所测量的数据对系统进行建模、估计和逼近,它可应用于如分类、预测及模式识别等众多方面。例如,函数映射是功能建模的一个典型例子。

传统的计算机程序也可完成类似的任务,在某些方面它们可以互相替代。然而更主要的是它们各有所长。传统的计算机程序比较适合于那些需要高精度的数值计算或者需要符号处理的那些任务。例如财务管理和计算,它比较适合于采用计算机程序,而不适合于采用神经网络。对于那些几乎没有规则,数据不完全或者多约束优化问题,则适合于用神经网络。例如用神经网络来控制一个工业过程便是这样的例子,对于这种情况很难定义规则,历史数据很多而且充满噪声,准确的计算是毫无必要的。

某些情况下应用神经网络会存在严重的缺点。当所给数据不充分或不存可学习的映射关系时,神经网络可能找不到满意的解。其次,有时很难估价神经网络给出的结果。神经网络中的连接权系数是千万次数据训练后的结果,对它的意义很难给出明显的解释,它对输出结果的影响也是非常复杂的。神经网络的训练是很慢的,而且有时需要付出很高的代价。这一方面是由于需要收集、分析和处理大量的训练数据,同时还需要相当的经验来选择合适的参数。

神经网络在实际应用时的执行时间也是需要加以检验的。执行时间取决于连接权的个数,它大体与网络结点个数的平方成正比。因此网络结点的稍许增加便可能引起执行时间的很大增长。对于有些应用问题尤其是控制,太大的执行时间则可能阻碍它的实时应用。这种情况下必须采用专用的硬件。

总之,应根据实际问题的特点来确定是采用神经网络还是常规的计算机程序。这两者可以结合起来使用,例如,神经网络可用作一个大的应用程序中的一个组成部分,其作用类似于可调用的一个函数,应用程序将一组数据传给神经网络,神经网络将结果返回给应用程序。

下面讨论训练神经网络的具体步骤和几个实际问题。

(1) 产生数据样本集

为了成功地开发出神经网络,产生数据样本集是第一步,也是十分重要和关键的一步。这里包括原始数据的收集、数据分析、变量选择以及数据的预处理,只有经过这些步骤后,才能对神经网络进行有效的学习和训练。

首先要在大量的原始测量数据中确定出最主要的输入模式。例如,若两个输入具有很强的相关性,则只需取其中一个作为输入,这就需要对原始数据进行统计分析,检验它们之间的相关性。又如工业过程可能记录了成百上千个压力、温度和流量数据。这时就需要对它们进行相关分析,找出其中一、二个最主要的量作为输入。

在确定了最重要的输入量后,需进行尺度变换和预处理。尺度变换常常将它们变换到 $[-1, 1]$ 或 $[0, 1]$ 的范围。在进行尺度变换前必须先检查是否存在异常点(或称野点),这

些点必须剔除。通过对数据的预处理分析还可以检验其是否存在周期性、固定变化趋势或其它关系。对数据的预处理就是要使得经变换后的数据对于神经网络更容易学习和训练。例如在过程控制中,采用温度的增量或导数比用温度值本身更能说明问题,也更容易找出变量之间的实质联系。在进行数据预处理时主要用到信号处理或特征抽取技术。如计算数据的和、差、倒数、乘幂、求根、对数、平均、滑动平均以及富立叶变换等。甚至于神经网络本身也可以作为数据预处理的工具,为另一个神经网络准备数据。

对于一个复杂问题应该选择多少数据,这也是一个很关键的问题。系统的输入输出关系就包含在这些数据样本中。所以一般说来,取的数据越多,学习和训练的结果便越能正确反映输入输出关系。但是选太多的数据将增加收集、分析数据以及网络训练所付出的代价。当然,选太少的数据则可能得不到正确的结果。事实上数据的多少取决于许多因素,如网络的大小、网络测试的需要以及输入输出的分布等。其中网络大小最关键。通常较大的网络需要较多的训练数据。一个经验规则是:训练模式应是连接权总数的 5 至 10 倍。

在神经网络训练完成后,需要有另外的测试数据来对网络加以检验,测试数据应是独立的数据集合。最简单的方法是:将收集到的可用数据随机地分成两部分,譬如说其中三分之二用于网络的训练,另外三分之一用于将来的测试,随机选取的目的是为了尽量减小这两部分数据的相关性。

影响数据大小的另一个因数是输入模式和输出结果的分布,对数据预先加以分类可以减少所需的数据量。相反,数据稀薄不匀甚至互相覆盖则势必要增加数据量。

(2) 确定网络的类型和结构

在训练神经网络之前,首先要确定所选用的网络类型。前面我们只介绍了两种类型的前馈型网络,问题相对比较简单。若主要用于模式分类,尤其是线性可分的情况,则可采用较为简单的感知器网络,若主要用于函数估计,则可应用 BP 网络。实际上,神经网络的类型很多,需根据问题的性质和任务的要求来合适地选择网络类型。一般是从已有的网络类型中选用一种比较简单而又能满足要求的网络,若新设计一个网络类型来满足问题的要求往往比较困难。

在网络的类型确定后,剩下的问题是选择网络的结构和参数。以 BP 网络为例,需选择网络的层数、每层的结点数、初始权值、阈值、学习算法、数值修改频度、结点变换函数及参数、学习率及动量项因子等参数。这里有些项的选择有一些指导原则,但更多的是靠经验和试凑。

对于具体问题若确定了输入和输出变量后,网络输入层和输出层的结点个数也便随之确定了。对于隐层的层数可首先考虑只选择一个隐层。剩下的问题是如何选择隐层的结点数。其选择原则是:在能正确反映输入输出关系的基础上,尽量选取较少的隐层结点数,而使网络尽量简单。具体选择可有如下两种方法:

a. 先设置较少的结点,对网络进行训练,并测试网络的逼近误差(后面还将介绍训练和测试的具体方法),然后逐渐增加结点数,直到测试的误差不再有明显的减小为止。

b. 先设置较多的结点,在对网络进行训练时,采用如下的误差代价函数

$$E_f = \frac{1}{2} \sum_{p=1}^P \sum_{i=1}^{n_q} (d_{pi} - x_{pi}^{(Q)})^2 + \epsilon \sum_{q=1}^Q \sum_{i=1}^{n_q} \sum_{j=1}^{n_{q-1}} |w_{ij}^{(q)}|$$

$$=E + \epsilon \sum_{q,i,j} |w_{ij}^{(q)}|$$

其中 E 仍与以前的定义相同,它表示输出误差的平方和。引入第二项的作用相当于引入一个“遗忘”项,其目的是为了使训练后的连接权系数尽量小。可以求得这时 E_f 对 $w_{ij}^{(q)}$ 的梯度为

$$\frac{\partial E_f}{\partial w_{ij}^{(q)}} = \frac{\partial E}{\partial w_{ij}^{(q)}} + \epsilon \operatorname{sgn}(w_{ij}^{(q)})$$

利用该梯度可以求得相应的学习算法。利用该学习算法,在训练过程中只有那些确实必要的连接权才予以保留,而那些不很必要的连接将逐渐衰减为零。最后可去掉那些影响不大的连接权和相应的结点,从而得到一个适当规模的网络结构。

若采用上述任一方法选择得到的隐层结点数太多。这时可考虑采用二个隐层。为了达到相同的映射关系,采用二个隐层的结点总数常常可比只用一个隐层时少。

(3) 训练和测试

最后一步是对网络进行训练和测试,在训练过程中对训练样本数据需要反复地使用。对所有样本数据正向运行一次并反传修改连接权一次称为一次训练(或一次学习),这样的训练需要反复地进行下去直至获得合适的映射结果。通常训练一个网络需要成百上千次。

特别应该注意的一点是,并非训练的次数越多,越能得到正确的输入输出的映射关系。训练网络的目的在于找出蕴含在样本数据中的输入和输出之间的本质联系,从而对于未经训练的输入也能给出合适的输出,即具备泛化功能。由于所收集的数据都是包含噪声的,训练的次数过多,网络将包含噪声的数据都记录了下来,在极端情况下,训练后的网络可以实现相当于查表的功能。但是对于新的输入数据却不能给出合适的输出,也即并不具备很好的泛化功能。网络的性能主要用它的泛化能力来衡量,它并不是用对训练数据的拟合程度来衡量,而是要用一组独立的数据来加以测试和检验。在用测试数据检验时,保持连接权系数不改变,只用该数据作为网络的输入,正向运行该网络,检验输出的均方误差。实际操作时应该训练和测试交替进行,即每训练一次,同时用测试数据测试一遍,画出均方误差随训练次数的变化曲线,如图 3.12 所示。

从误差曲线可以看出,在用测试数据检验时,均方误差开始逐渐减小,当训练次数再增加时,测试检验误差反而增加。误差曲线上极小点所对应的即为恰当的训练次数,若再训练即为“过度训练”了。

对于网络隐层结点数的选择如果采用试验法,也必须将训练与测试相结合,最终也用测试误差来衡量网络的性能。均方误差与隐层结点数也有与图 3.12 相类似的关系,因此也不是结点数越多越好。

网络的结点数对网络的泛化能力有很大影响,结点数太多,它倾向于记住所有的训练

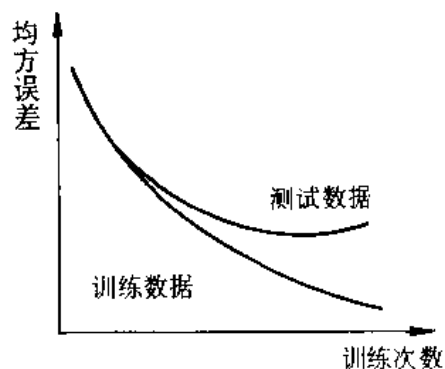


图 3.12 均方误差曲线

数据,包括噪声的影响,反而降低了泛化能力;而结点数太少,它不能拟合样本数据,因而也谈不上有较好的泛化能力。选择结点数原则是:选择尽量少的结点数以实现尽量好的泛化能力。

在用试验法选择其它参数时也必须最终检验测试数据的误差。例如初始权值的选择,一般可用随机法产生。为避免局部极值问题,可选取多组初始权值。最后选用最好的一种,这里也是靠检验测试数据误差来进行比较。

3.3 反馈神经网络

本节讨论如图 3.5 所示的反馈型神经网络。反馈网络是一种动态网络,它需要工作一段时间才能达到稳定。该网络主要用于联想记忆和优化计算。由于该网络是首先由 Hopfield 提出的,因此通常称它为 Hopfield 网。根据网络的输出是离散量或是连续量,Hopfield 网络也分为离散和连续的两种。下面分别对它们进行讨论。

3.3.1 离散 Hopfield 网络

1. 网络的结构和工作方式

离散 Hopfield 网络的结构如图 3.13 所示。

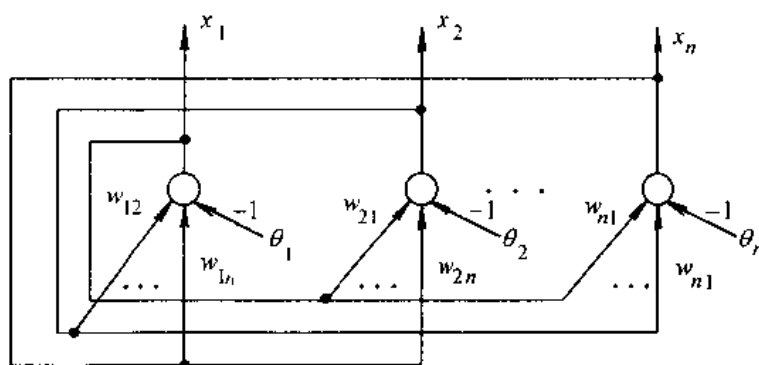


图 3.13 离散 Hopfield 网络

可以看出它是一个单层网络,共有 n 个神经元结点,每个结点输出均连接到其它神经元的输入,同时所有其它神经元的输出均连到该神经元的输入。对于每一个神经元结点,其工作方式仍同以前一样,即

$$\begin{cases} s_i = \sum_{j=1}^n w_{ij}x_j - \theta_i \\ x_i = f(s_i) \end{cases}$$

其中 $w_{ii}=0$, θ_i 为阈值, $f(\cdot)$ 是变换函数。对于离散 Hopfield 网络, $f(\cdot)$ 通常取为二值函数,即

$$f(s) = \begin{cases} 1 & s \geq 0 \\ -1 & s < 0 \end{cases}$$

或

$$f(s) = \begin{cases} 1 & s \geq 0 \\ 0 & s < 0 \end{cases}$$

整个网络有如下两种工作方式：

(1) 异步方式。每次只有一个神经元结点进行状态的调整计算，其它结点的状态均保持不变，即

$$\begin{cases} x_i(k+1) = f\left(\sum_{j=1}^n w_{ij}x_j(k) - \theta_i\right) \\ x_j(k+1) = x_j(k), \quad j \neq i \end{cases}$$

其调整次序可以随机选定，也可按规定的次序进行。

(2) 同步方式。所有的神经元结点同时调整状态，即

$$x_i(k+1) = f\left(\sum_{j=1}^n w_{ij}x_j(k) - \theta_i\right) \quad \forall i$$

上述同步计算方式也可写成如下的矩阵形式：

$$\mathbf{x}(k+1) = f(W\mathbf{x}(k) - \boldsymbol{\theta})$$

其中 $\mathbf{x} = [x_1 \ x_2 \ \cdots \ x_n]^T$ 和 $\boldsymbol{\theta} = [\theta_1 \ \theta_2 \ \cdots \ \theta_n]^T$ 是向量， W 是由 w_{ij} 所组成的 $n \times n$ 矩阵， $f(s)$ 是向量函数，它表示 $f(s) = [f(s_1) \ f(s_2) \ \cdots \ f(s_n)]^T$ 。

该网络是动态的反馈网络，其输入是网络的状态初值

$$\mathbf{x}(0) = [x_1(0) \ x_2(0) \ \cdots \ x_n(0)]^T$$

输出是网络的稳定状态 $\lim_{k \rightarrow \infty} \mathbf{x}(k)$ 。

2. 稳定性和吸引子

从上述工作过程可以看出，离散 Hopfield 网络实质上是一个离散的非线性动力学系统。因此如果系统是稳定的，则它可以从任一初态收敛到一个稳定状态；若系统是不稳定的，由于网络结点输出点只有 1 和 -1（或 1 和 0）两种状态，因而系统不可能出现无限发散，只可能出现限幅的自持振荡或极限环。

若将稳态视为一个记忆样本，那么初态朝稳态的收敛过程便是寻找记忆样本的过程。初态可认为是给定样本的部分信息，网络改变的过程可认为是从部分信息找到全部信息，从而实现了联想记忆的功能。

若将稳态与某种优化计算的目标函数相对应，并作为目标函数的极小点。那么初态朝稳态的收敛过程便是优化计算过程。该优化计算是在网络演变过程中自动完成的。

(1) 稳定性

定义 3.1： 若网络的状态 \mathbf{x} 满足 $\mathbf{x} = f(W\mathbf{x} - \boldsymbol{\theta})$ ，则称 \mathbf{x} 为网络的稳定点或吸引子。

定理 3.1： 对于离散 Hopfield 网络，若按异步方式调整状态，且连接权矩阵 W 为对称阵，则对于任意初态，网络都最终收敛到一个吸引子。

证明： 定义网络的能量函数为

$$E(k) = -\frac{1}{2}\mathbf{x}^T(k)W\mathbf{x}(k) + \mathbf{x}^T(k)\theta$$

由于神经元结点的状态只能取 1 和 -1 (或 1 和 0) 两种状态, 因此上述定义的能量函数 $E(k)$ 是有界的。令 $\Delta E(k) = E(k+1) - E(k)$, $\Delta \mathbf{x}(k) = \mathbf{x}(k+1) - \mathbf{x}(k)$, 则

$$\begin{aligned}\Delta E(k) &= E(k+1) - E(k) \\ &= -\frac{1}{2}[\mathbf{x}(k) + \Delta \mathbf{x}(k)]^T W [\mathbf{x}(k) + \Delta \mathbf{x}(k)] + [\mathbf{x}(k) + \Delta \mathbf{x}(k)]^T \theta \\ &\quad - \left[-\frac{1}{2}\mathbf{x}^T(k)W\mathbf{x}(k) + \mathbf{x}^T(k)\theta \right] \\ &= -\Delta \mathbf{x}^T(k)W\mathbf{x}(k) - \frac{1}{2}\Delta \mathbf{x}^T(k)W\Delta \mathbf{x}(k) + \Delta \mathbf{x}^T(k)\theta \\ &= -\Delta \mathbf{x}^T(k)[W\mathbf{x}(k) - \theta] - \frac{1}{2}\Delta \mathbf{x}^T(k)W\Delta \mathbf{x}(k)\end{aligned}$$

由于假定为异步工作方式, 因此可设第 k 时刻只有第 i 个神经元调整状态, 即 $\Delta \mathbf{x}(k) = [0 \cdots 0 \Delta x_i(k) 0 \cdots 0]^T$ 代入上式则有

$$\Delta E(k) = -\Delta x_i(k) \left[\sum_{j=1}^n w_{ij}x_j(k) - \theta_i \right] - \frac{1}{2}\Delta x_i^2 w_{ii}$$

令 $s_i(k) = \sum_{j=1}^n w_{ij}x_j(k) - \theta_i$, 则

$$\begin{aligned}\Delta E(k) &= -\Delta x_i(k)[s_i(k) + \frac{1}{2}\Delta x_i(k)w_{ii}] \\ &= -\Delta x_i(k)s_i(k) \quad (w_{ii} = 0)\end{aligned}$$

设神经元结点取 1 和 -1 两种状态, 则

$$x_i(k+1) = f[s_i(k)] = \begin{cases} 1 & s_i(k) \geq 0 \\ -1 & s_i(k) < 0 \end{cases}$$

下面考虑 $\Delta x_i(k)$ 可能出现的各种情况:

$$\text{a. } x_i(k) = -1, x_i(k+1) = f[s_i(k)] = 1$$

这时有 $\Delta x_i(k) = 2, s_i(k) \geq 0$, 从而 $\Delta E(k) \leq 0$

$$\text{b. } x_i(k) = 1, x_i(k+1) = f[s_i(k)] = -1$$

这时有 $\Delta x_i(k) = -2, s_i(k) < 0$, 从而 $\Delta E(k) < 0$

$$\text{c. } x_i(k+1) = x_i(k)$$

这时有 $\Delta x_i(k) = 0$, 从而 $\Delta E(k) = 0$

可见, 在任何情况下均有 $\Delta E(k) \leq 0$, 由于 $E(k)$ 有下界, 所以 $E(k)$ 将收敛到一常数。

下面需考察 $E(k)$ 收敛到常数时是否对应于网络的吸引子。根据上述分析, 当 $\Delta E(k) = 0$ 时, 相应于以下两种情况之一:

$$\text{a. } x_i(k+1) = x_i(k) = 1 \text{ 或 } x_i(k+1) = x_i(k) = -1$$

$$\text{b. } x_i(k) = -1, x_i(k+1) = 1, s_i(k) = 0$$

对于情况 a, 表明 x_i 已进入稳定态; 对于情况 b, 网络继续演变时 $x_i = 1$ 也将不会再变化, 因为若 x_i 由 1 再变回 -1, 则有 $\Delta E < 0$, 它与 $E(k)$ 已收敛到常数相矛盾。所以网络最终将收敛到吸引子。

上述分析时假设 $w_{ii}=0$, 实际上不难看出, 当 $w_{ii}>0$ 时上述结论仍成立, 而且收敛过程将更快。

上而证明时假设神经元结点取 1 和 -1 两种状态, 不难验证当 x 取 1 和 0 两种状态时, 上述结论也是成立的。

定理 3.2: 对于离散 Hopfield 网络, 若按同步方式调整状态, 且连接权矩阵 W 为非负定对称阵, 则对于任意初态, 网络都最终收敛到一个吸引子。

证明: 前已求得

$$\begin{aligned}\Delta E(k) &= E(k+1) - E(k) \\ &= -\Delta \mathbf{x}^T(k)[W\mathbf{x}(k) - \theta] - \frac{1}{2}\Delta \mathbf{x}^T(k)W\Delta \mathbf{x}(k) \\ &= -\Delta \mathbf{x}^T(k)\mathbf{s}(k) - \frac{1}{2}\Delta \mathbf{x}^T(k)W\Delta \mathbf{x}(k) \\ &= -\sum_{i=1}^n \Delta x_i(k)s_i(k) - \frac{1}{2}\Delta \mathbf{x}^T(k)W\Delta \mathbf{x}(k)\end{aligned}$$

前已证得, $\forall i$ 有 $-\Delta x_i(k)s_i(k) \leq 0$, 因此只要 W 为非负定阵即有 $\Delta E(k) \leq 0$, 也即 $E(k)$ 最终将收敛到一个常数值, 并按照如上而同样的分析可说明网络将最终收敛到一个吸引子。

可见对于同步方式, 它对连接权矩阵 W 的要求更高了。若不满足 W 为非负定对称阵的要求, 则网络可能出现自持振荡即极限环。

由于异步工作方式比同步方式有更好的稳定性能, 实用时较多采用异步工作方式。异步方式的主要缺点是失去了神经网络并行处理的优点。

(2) 吸引子的性质

a. 若 \mathbf{x} 是网络的一个吸引子, 且 $\forall i, \theta_i = 0, \sum_{j=1}^n w_{ij}x_j \neq 0$, 则 $-\mathbf{x}$ 也一定是该网络的吸引子。

证明: 由于 \mathbf{x} 是吸引子, 即 $\mathbf{x} = f[W\mathbf{x}]$, 从而有 $f[W(-\mathbf{x})] = f[-W\mathbf{x}] = -f[W\mathbf{x}] = -\mathbf{x}$, 即 $-\mathbf{x}$ 也是该网络的吸引子。

b. 若 $\mathbf{x}^{(a)}$ 是网络的吸引子, 则与 $\mathbf{x}^{(a)}$ 的海明距离 $d_H(\mathbf{x}^{(a)}, \mathbf{x}^{(b)}) = 1$ 的 $\mathbf{x}^{(b)}$ 一定不是吸引子, 海明距离定义为两个向量中不相同的元素的个数。

证明: 不失一般性, 设 $x_1^{(a)} \neq x_1^{(b)}, x_i^{(a)} = x_i^{(b)} (i=2, 3, \dots, n)$ 。因为 $w_{11}=0$, 所以有

$$x_1^{(a)} = f\left[\sum_{j=2}^n w_{1j}x_j^{(a)} - \theta_1\right] = f\left[\sum_{j=2}^n w_{1j}x_j^{(b)} - \theta_1\right] \neq x_1^{(b)}$$

所以 $\mathbf{x}^{(b)}$ 一定不是网络的吸引子。

推论: 若 $\mathbf{x}^{(a)}$ 是网络的吸引子, 且 $\forall i, \theta_i = 0, \sum_{j=1}^n w_{ij}x_j^{(a)} \neq 0$ 则 $d_H(\mathbf{x}^{(a)}, \mathbf{x}^{(b)}) = n-1$ 的 $\mathbf{x}^{(b)}$ 一定不是吸引子。

证明: 若 $d_H(\mathbf{x}^{(a)}, \mathbf{x}^{(b)}) = n-1$, 则 $d_H(-\mathbf{x}^{(a)}, \mathbf{x}^{(b)}) = 1$ 。根据性质 a, $\mathbf{x}^{(a)}$ 是网络的吸引子, $-\mathbf{x}^{(a)}$ 也是网络的吸引子, 根据性质 b, $\mathbf{x}^{(b)}$ 一定不是吸引子。

(3) 吸引域

为了能够实现正确的联想记忆, 对于每个吸引子应该有一定的吸引范围, 这个吸引范围

便称为吸引域。下面给出较严格的定义。

定义 3.2: 若 $\mathbf{x}^{(a)}$ 是吸引子, 对于异步方式, 若存在一个调整次序可以从 \mathbf{x} 演变到 $\mathbf{x}^{(a)}$, 则称 \mathbf{x} 弱吸引到 $\mathbf{x}^{(a)}$; 若对于任意调整次序都可以从 \mathbf{x} 演变到 $\mathbf{x}^{(a)}$, 则称 \mathbf{x} 强吸引到 $\mathbf{x}^{(a)}$ 。

定义 3.3: 对所有 $\mathbf{x} \in R(\mathbf{x}^{(a)})$ 均有 \mathbf{x} 弱(强)吸引到 $\mathbf{x}^{(a)}$, 则称 $R(\mathbf{x}^{(a)})$ 为 $\mathbf{x}^{(a)}$ 的弱(强)吸引域。

对于同步方式, 由于无调整次序问题, 所以相应的吸引域也无强弱之分。

对于异步方式, 对同一个状态, 若采用不同的调整次序, 有可能弱吸引到不同的吸引子。

3. 连接权的设计

为了保证 Hopfield 网络在异步方式工作时能稳定收敛, 连接权矩阵 W 应是对称的。若要保证同步方式收敛, 则要求 W 为非负定阵, 这个要求比较高。因而设计 W 一般只保证异步方式收敛。另外一个要求是对于给定的样本必须是网络的吸引子, 而且还要有一定的吸引域, 这样才能正确实现联想记忆功能。为了实现上述功能, 通常采用 Hebb 规则来设计连接权。

设给定 m 个样本 $\mathbf{x}^{(k)} (k=1, 2, \dots, m)$, 并设 $\mathbf{x} \in \{-1, 1\}^n$ 则按 Hebb 规则设计的连接权为

$$w_{ij} = \begin{cases} \sum_{k=1}^m x_i^{(k)} x_j^{(k)} & i \neq j \\ 0 & i = j \end{cases}$$

或

$$\begin{cases} w_{ij}(k) = w_{ij}(k-1) + x_i^{(k)} x_j^{(k)} & k = 1, 2, \dots, m \\ w_{ij}(0) = 0 & w_{ii} = 0 \end{cases}$$

写成矩阵形式则为

$$\begin{aligned} W &= [\mathbf{x}^{(1)} \mathbf{x}^{(2)} \dots \mathbf{x}^{(m)}] \begin{bmatrix} \mathbf{x}^{(1)T} \\ \mathbf{x}^{(2)T} \\ \vdots \\ \mathbf{x}^{(m)T} \end{bmatrix} - mI \\ &= \sum_{k=1}^m \mathbf{x}^{(k)} \mathbf{x}^{(k)T} - mI = \sum_{k=1}^m (\mathbf{x}^{(k)} \mathbf{x}^{(k)T} - I) \end{aligned}$$

其中 I 为单位矩阵。

当网络结点状态为 1 和 0 两种状态即 $\mathbf{x} \in \{0, 1\}^n$ 时, 相应的连接权为

$$w_{ij} = \begin{cases} \sum_{k=1}^m (2x_i^{(k)} - 1)(2x_j^{(k)} - 1) & i \neq j \\ 0 & i = j \end{cases}$$

或

$$\begin{cases} w_{ij}(k) = w_{ij}(k-1) + (2x_i^{(k)} - 1)(2x_j^{(k)} - 1) & k = 1, 2, \dots, m \\ w_{ij}(0) = 0 & w_{ii} = 0 \end{cases}$$

写成矩阵形式则为

$$W = \sum_{k=1}^m (2\mathbf{x}^{(k)} - \mathbf{b})(2\mathbf{x}^{(k)} - \mathbf{b})^T - mI$$

其中 $\mathbf{b} = [1 \ 1 \ \cdots \ 1]^T$ 。

显然,上面所设计的连接权矩阵满足对称性的要求。下面进一步分析所给样本是否为网络的吸引子,这一点是十分重要的。下面以 $\mathbf{x} \in \{-1, 1\}^n$ 的情况为例进行分析。

若 m 个样本 $\mathbf{x}^{(k)} (k=1, 2, \cdots, m)$ 是两两正交的,即

$$\begin{cases} \mathbf{x}^{(i)T} \mathbf{x}^{(j)} = 0 & i \neq j \\ \mathbf{x}^{(i)T} \mathbf{x}^{(i)} = n \end{cases}$$

则有

$$\begin{aligned} W\mathbf{x}^{(k)} &= \left(\sum_{i=1}^m \mathbf{x}^{(i)} \mathbf{x}^{(i)T} - mI \right) \mathbf{x}^{(k)} = \sum_{i=1}^m \mathbf{x}^{(i)} \mathbf{x}^{(i)T} \mathbf{x}^{(k)} - m\mathbf{x}^{(k)} \\ &= n\mathbf{x}^{(k)} - m\mathbf{x}^{(k)} = (n - m)\mathbf{x}^{(k)} \end{aligned}$$

可见,只要满足 $n - m > 0$,便有

$$f[W\mathbf{x}^{(k)}] = f[(n - m)\mathbf{x}^{(k)}] = \mathbf{x}^{(k)}$$

也即 $\mathbf{x}^{(k)}$ 是网络的吸引子。

若 m 个样本 $\mathbf{x}^{(k)} (k=1, 2, \cdots, m)$ 不是两两相交,且设向量之间的内积为: $\mathbf{x}^{(i)T} \mathbf{x}^{(j)} = \beta_{ij}$, 显然 $\beta_{ii} = n, (i=1, 2, \cdots, m)$ 。则有

$$W\mathbf{x}^{(k)} = \sum_{i=1}^m \mathbf{x}^{(i)} \mathbf{x}^{(i)T} \mathbf{x}^{(k)} - m\mathbf{x}^{(k)} = (n - m)\mathbf{x}^{(k)} + \sum_{\substack{i=1 \\ i \neq k}}^m \mathbf{x}^{(i)} \beta_{ik}$$

取其中第 j 个元素

$$[W\mathbf{x}^{(k)}]_j = (n - m)x_j^{(k)} + \sum_{\substack{i=1 \\ i \neq k}}^m x_j^{(i)} \beta_{ik}$$

若能使得 $\forall j$ 有

$$n - m > \left| \sum_{\substack{i=1 \\ i \neq k}}^m x_j^{(i)} \beta_{ik} \right|$$

则 $\mathbf{x}^{(k)}$ 是网络的吸引子。上式右端可进一步化为

$$\left| \sum_{\substack{i=1 \\ i \neq k}}^m x_j^{(i)} \beta_{ik} \right| \leq \sum_{\substack{i=1 \\ i \neq k}}^m |\beta_{ik}| \leq (m - 1)\beta_m$$

其中 $\beta_m \triangleq |\beta_{ik}|_{\max}$ 。进而若能使得 $n - m > (m - 1)\beta_m$, 即

$$m < \frac{n + \beta_m}{1 + \beta_m}$$

则可以保证所有的样本均为网络的吸引子。

若 m 个样本满足

$$\alpha n \leq d_H(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \leq (1 - \alpha)n$$

其中 $i, j = 1, 2, \cdots, m, i \neq j, 0 < \alpha < 0.5$, 则有

$$|\beta_{ij}| \leq n - 2\alpha n = \beta_m$$

从而得出 m 个样本均为网络吸引子的条件为

$$m < \frac{2n(1-\alpha)}{1+n(1-2\alpha)}$$

注意,上式仅为充分条件。当不满足上述条件时,需要具体检验才能确定。

4. 记忆容量

所谓记忆容量是指:在网络结构参数一定的条件下,要保证联想功能的正确实现,网络所能存储的最大的样本数。也就是说,给定网络结点数 n ,样本数 m 最大可为多少。这些样本向量不仅本身应为网络的吸引子,而且应有一定的吸引域,这样才能实现联想记忆的功能。

记忆容量不仅与结点数 n 有关,它还与连接权的设计有关,适当地设计连接权可以提高网络的记忆容量。记忆容量还与样本本身的性质有关,对于用 Hebb 规则设计连接权的网络,如果输入样本是正交的,则可以获得最大的记忆容量。实际问题的样本不可能都是正交的,所以在研究记忆容量时通常都假设样本向量是随机的。

记忆容量还与要求的吸引域大小有关,要求的吸引域越大,则记忆容量便越小。一个样本向量 $\mathbf{x}^{(k)}$ 的吸引域可以看成是以该向量为中心的球体。在该球体中的向量 $\mathbf{x}^{(i)}$ 满足 $d_H(\mathbf{x}^{(i)}, \mathbf{x}^{(k)}) \leq \alpha n$ ($0 \leq \alpha \leq 0.5$),称 α 为吸引半径。

对于给定的网络,严格的分析并确定其记忆容量并不是一件很容易的事情。Hopfield 曾提出了一个数量范围,即

$$m \leq 0.15n$$

按照样本为随机分布的假设所作的理论分析表明,当 $n \rightarrow \infty$ 时,其记忆容量为

$$m \leq \frac{(1-2\alpha)^2 n}{2 \ln n}$$

其中 α 为要求的吸引半径。

上面提到,当样本为两两正交时可以有最大的记忆容量。对于一般的记忆样本,可以通过改进连接权的设计来提高记忆容量。下面介绍其中的一种方法。

设给定 m 个样本向量 $\mathbf{x}^{(k)}$ ($k=1, 2, \dots, m$),首先组成如下的 $n \times (m-1)$ 阶矩阵

$$A = [\mathbf{x}^{(1)} - \mathbf{x}^{(m)}, \mathbf{x}^{(2)} - \mathbf{x}^{(m)}, \dots, \mathbf{x}^{(m-1)} - \mathbf{x}^{(m)}]$$

对 A 进行奇异值分解

$$A = U \Sigma V^T$$

其中

$$\Sigma = \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix}$$

$$S = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_r)$$

U 是 $n \times n$ 正交阵, V 是 $(m-1) \times (m-1)$ 正交阵, U 可表示成

$$U = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_r, \mathbf{u}_{r+1}, \dots, \mathbf{u}_n]$$

则 $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_r$ 是对应于非零奇异值 $\sigma_1, \sigma_2, \dots, \sigma_r$ 的左奇异向量,且组成了 A 的值域空间的正交基; $\mathbf{u}_{r+1}, \dots, \mathbf{u}_n$ 是 A 的值域的正交补空间的正交基。

按如下方法组成连接权矩阵 W 和阈值向量 θ 。

$$W = \sum_{k=1}^r u_k u_k^T$$

$$\theta = Wx^{(m)} - x^{(m)}$$

显然,按上述方法求得的连接权矩阵是对称的。因而可保证异步工作方式的稳定性,下面进一步证明给定的样本向量 $x^{(k)} (k=1,2,\cdots,m)$ 都是吸引子。

由于 u_1, u_2, \cdots, u_r 是 A 的值域空间的正交基,所以 A 中的任一向量 $x^{(k)} - x^{(m)} (k=1, 2, \cdots, m-1)$ 均可表示为 u_1, u_2, \cdots, u_r 的线性组合,即

$$x^{(k)} - x^{(m)} = \sum_{i=1}^r \alpha_i u_i$$

由于 U 为正交阵,所以 u_1, u_2, \cdots, u_r 为互相正交的单位向量,从而对任一向量 $u_i (i=1, 2, \cdots, r)$ 有

$$Wu_i = \sum_{k=1}^r u_k u_k^T u_i = u_i$$

进而有

$$W(x^{(k)} - x^{(m)}) = W \sum_{i=1}^r \alpha_i u_i = \sum_{i=1}^r \alpha_i (Wu_i)$$

$$= \sum_{i=1}^r \alpha_i u_i = x^{(k)} - x^{(m)}$$

对于任一样本向量 $x^{(k)} (k=1, 2, \cdots, m-1)$, 有

$$Wx^{(k)} - \theta = Wx^{(k)} - Wx^{(m)} + x^{(m)} = W(x^{(k)} - x^{(m)}) + x^{(m)} = x^{(k)}$$

从而有

$$f(Wx^{(k)} - \theta) = f(x^{(k)}) = x^{(k)}$$

对于第 m 个样本 $x^{(m)}$ 有

$$Wx^{(m)} - \theta = Wx^{(m)} - Wx^{(m)} + x^{(m)} = x^{(m)}$$

从而有

$$f(Wx^{(m)} - \theta) = f(x^{(m)}) = x^{(m)}$$

以上推证过程说明,按照这种方法设计的连接权矩阵,可以使得所有的样本 $x^{(k)} (k=1, 2, \cdots, m)$ 均为网络的吸引子。而并不要求它们两两正交,也就是说,按此设计提高了网络的记忆容量。

5. 举例

设离散 Hopfield 网络的结构如图 3.13 所示。其中 $n=4, \theta_i=0 (i=1, 2, 3, 4), m=2$, 两个样本为

$$x^{(1)} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad x^{(2)} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}$$

首先根据 Hebb 规则求得连接权矩阵为

$$W = \mathbf{x}^{(1)}\mathbf{x}^{(1)\top} + \mathbf{x}^{(2)}\mathbf{x}^{(2)\top} - 2I = \begin{bmatrix} 0 & 2 & 2 & 2 \\ 2 & 0 & 2 & 2 \\ 2 & 2 & 0 & 2 \\ 2 & 2 & 2 & 0 \end{bmatrix}$$

这里 $d_H(\mathbf{x}^{(1)}, \mathbf{x}^{(2)})=4$, 相当于 $\alpha=0$, 显然它不满足上面给出的充分条件。 $\mathbf{x}^{(1)}$ 和 $\mathbf{x}^{(2)}$ 是否是网络的吸引子需具体加以检验:

$$f(W\mathbf{x}^{(1)}) = f\begin{bmatrix} 6 \\ 6 \\ 6 \\ 6 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \mathbf{x}^{(1)} \quad f(W\mathbf{x}^{(2)}) = f\begin{bmatrix} -6 \\ -6 \\ -6 \\ -6 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} = \mathbf{x}^{(2)}$$

可见, 两个样本 $\mathbf{x}^{(1)}$ 和 $\mathbf{x}^{(2)}$ 均为网络的吸引子。事实上, 由于 $\mathbf{x}^{(2)} = -\mathbf{x}^{(1)}$, 只要其中一个是吸引子, 那么另一个也必为吸引子。

下面再考察这两个吸引子是否具有一定的吸引能力, 即是否具备联想记忆的功能。

(1) 设 $\mathbf{x}(0) = \mathbf{x}^{(3)} = [-1 \ 1 \ 1 \ 1]^\top$, 显然它比较靠近 $\mathbf{x}^{(1)}$ 。下面用异步方式按 1, 2, 3, 4 的调整次序来演变网络:

$$x_1(1) = f\left(\sum_{j=1}^n w_{1j}x_j(0)\right) = f(6) = 1$$

$$x_2(1) = x_2(0) = 1 \quad x_3(1) = x_3(0) = 1 \quad x_4(1) = x_4(0) = 1$$

即 $\mathbf{x}(1) = [1 \ 1 \ 1 \ 1]^\top = \mathbf{x}^{(1)}$ 。可见, 只需异步方式调整一步即收敛到 $\mathbf{x}^{(1)}$ 。

(2) 设 $\mathbf{x}(0) = \mathbf{x}^{(4)} = [1 \ -1 \ -1 \ -1]^\top$, 显然它比较靠近 $\mathbf{x}^{(2)}$ 。下面仍用异步方式按 1, 2, 3, 4 的调整次序演变网络:

$$x_1(1) = f\left(\sum_{j=1}^n w_{1j}x_j(0)\right) = f(-6) = -1$$

$$x_2(1) = x_2(0) = -1 \quad x_3(1) = x_3(0) = -1 \quad x_4(1) = x_4(0) = -1$$

即 $\mathbf{x}(1) = [-1 \ -1 \ -1 \ -1]^\top = \mathbf{x}^{(2)}$ 。可见只调整一步便收敛到 $\mathbf{x}^{(2)}$ 。

(3) 设 $\mathbf{x}(0) = \mathbf{x}^{(5)} = [1 \ 1 \ -1 \ -1]^\top$, 这时它与 $\mathbf{x}^{(1)}$ 和 $\mathbf{x}^{(2)}$ 的海明距离均为 2。若按 1, 2, 3, 4 的次序调整网络可得

$$x_1(1) = f\left(\sum_{j=1}^n w_{1j}x_j(0)\right) = f(-2) = -1$$

$$x_i(1) = x_i(0) \quad i = 2, 3, 4$$

即 $\mathbf{x}(1) = [-1 \ 1 \ -1 \ -1]^\top$

$$x_2(2) = f\left(\sum_{j=1}^n w_{2j}x_j(1)\right) = f(-6) = -1$$

$$x_i(2) = x_i(1) \quad i = 1, 3, 4$$

即 $\mathbf{x}(2) = [-1 \ -1 \ -1 \ -1]^\top = \mathbf{x}^{(2)}$ 。可见此时 $\mathbf{x}^{(5)}$ 收敛到了 $\mathbf{x}^{(2)}$ 。

若按 3, 4, 1, 2 的次序调整网络可得

$$x_3(1) = f\left(\sum_{j=1}^n w_{3j}x_j(0)\right) = f(2) = 1$$

$$x_i(1) = x_i(0) \quad i = 1, 2, 4$$

即 $\mathbf{x}(1) = [1 \ 1 \ 1 \ -1]^T$

$$x_4(2) = f\left(\sum_{j=1}^n w_{4j}x_j(1)\right) = f(6) = 1$$

$$x_i(2) = x_i(1) \quad i = 1, 2, 3$$

即 $\mathbf{x}(2) = [1 \ 1 \ 1 \ 1]^T = \mathbf{x}^{(1)}$ 。可见此时 $\mathbf{x}^{(5)}$ 收敛到了 $\mathbf{x}^{(1)}$ 。

从上面具体计算可以看出,对于不同的调整次序 $\mathbf{x}^{(5)}$ 既可弱收敛到 $\mathbf{x}^{(1)}$ 也可弱收敛到 $\mathbf{x}^{(2)}$ 。

下面对该例应用同步方式进行计算,仍取 $\mathbf{x}(0)$ 为 $\mathbf{x}^{(3)}$, $\mathbf{x}^{(4)}$ 和 $\mathbf{x}^{(5)}$ 三种情况。

$$(1) \quad \mathbf{x}(0) = \mathbf{x}^{(3)} = [-1 \ 1 \ 1 \ 1]^T$$

$$\mathbf{x}(1) = f[W\mathbf{x}(0)] = f(W\mathbf{x}^{(3)}) = f \begin{bmatrix} 6 \\ 2 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\mathbf{x}(2) = f[W\mathbf{x}(1)] = f \begin{bmatrix} 6 \\ 6 \\ 6 \\ 6 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

可见此时 $\mathbf{x}^{(3)}$ 收敛到了 $\mathbf{x}^{(1)}$ 。

$$(2) \quad \mathbf{x}(0) = \mathbf{x}^{(4)} = [1 \ -1 \ -1 \ -1]^T$$

$$\mathbf{x}(1) = f[W\mathbf{x}(0)] = f \begin{bmatrix} -6 \\ -2 \\ -2 \\ -2 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}$$

$$\mathbf{x}(2) = f[W\mathbf{x}(1)] = f \begin{bmatrix} -6 \\ -6 \\ -6 \\ -6 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}$$

可见此时 $\mathbf{x}^{(4)}$ 收敛到了 $\mathbf{x}^{(2)}$ 。

$$(3) \quad \mathbf{x}(0) = \mathbf{x}^{(5)} = [1 \ 1 \ -1 \ -1]^T$$

$$\mathbf{x}(1) = f[W\mathbf{x}(0)] = f \begin{bmatrix} -2 \\ -2 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$$

$$\mathbf{x}(2) = f[W\mathbf{x}(1)] = f \begin{bmatrix} 2 \\ 2 \\ -2 \\ -2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

可见,它将在两个状态间跳跃,产生极限环为2的自持振荡。若根据前面的稳定性分析,由于此时连接权矩阵 W 不是非负定阵,所以出现了振荡。

3.3.2 连续 Hopfield 网络

1. 网络的结构和工作方式

连续的 Hopfield 网络也是单层的反馈网络,其结构仍如图 3.13 所示,对于每一个神经元结点,其工作方式为

$$\begin{cases} s_i = \sum_{j=1}^n w_{ij}x_j - \theta_i \\ \frac{dy_i}{dt} = -\frac{1}{\tau}y_i + s_i \\ x_i = f(y_i) \end{cases}$$

这里,同样假定 $w_{ij}=w_{ji}$,它与离散的 Hopfield 网络相比,这里多了中间一个式子,该式是一阶微分方程,相当于一阶惯性环节。 s_i 是该环节的输入, y_i 是该环节的输出。对于离散的 Hopfield 网络,中间的式子也可看成为 $y_i=s_i$ 。它们之间的另一个差别是第三个式子一般不再是二值函数,而一般取 S 形函数,即当 $x_i \in (-1,1)$ 时取

$$x_i = f(y_i) = \frac{1 - e^{-\mu y_i}}{1 + e^{-\mu y_i}}$$

当 $x_i \in (0,1)$ 时,取

$$x_i = f(y_i) = \frac{1}{1 + e^{-\mu y_i}}$$

它们都是连续的单调上升的函数,如图 3.14 所示。这里 θ_i 表示阈值。其它文献中 θ_i 前通

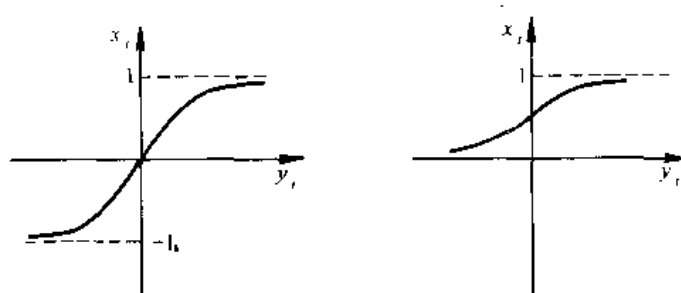


图 3.14 变换函数 $x_i = f(y_i)$

常取正号。为使它与离散 Hopfield 网络表示一致,这里取为负号。

Hopfield 利用模拟电路设计了一个连续 Hopfield 网络的电路模型。图 3.15 表示了其中由运算放大器电路实现的一个结点的模型。

根据图 3.15 可以列出如下的电路方程:

$$\begin{cases} C_i \frac{du_i}{dt} + \frac{u_i}{R_i} + I_i = \sum_{j=1}^n \frac{V_j - u_i}{R_{ij}} \\ V_i = f(u_i) \end{cases}$$

经整理得

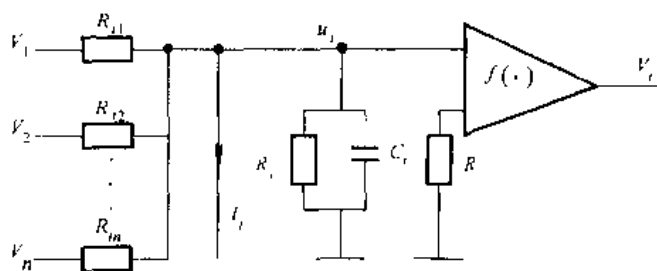


图 3.15 连续 Hopfield 网络的一个神经元结点的电路模型

$$\begin{cases} \frac{du_i}{dt} = -\frac{1}{R'_i C_i} u_i + \sum_{j=1}^n \frac{1}{R_{ij} C_i} V_j - \frac{I_i}{C_i} \\ V_i = f(u_i) \end{cases}$$

其中

$$\frac{1}{R'_i} = \frac{1}{R_i} + \sum_{j=1}^n \frac{1}{R_{ij}}$$

若令 $x_i = V_i, y_i = u_i, \tau = R'_i C_i, w_{ij} = \frac{1}{R_{ij} C_i}, \theta_i = \frac{I_i}{C_i}$

则上式化为

$$\begin{cases} \frac{dy_i}{dt} = -\frac{1}{\tau} y_i + \sum_{j=1}^n w_{ij} x_j - \theta_i \\ x_i = f(y_i) \end{cases}$$

可以看出,连续 Hopfield 网络实质上是一个连续的非线性动力学系统,它可用一组非线性微分方程来描述。当给定初始状态 $x_i(0)$ ($i=1, 2, \dots, n$),通过求解非线性微分方程组即可求得网络状态的运动轨迹。若系统是稳定的,则它最终可收敛到一个稳定状态。若用图 3.15 所示的硬件来实现,则这个求解非线性微分方程的过程将由该电路自动完成,其求解速度是非常快的。

2. 稳定性

定义连续 Hopfield 网络的能量函数为

$$\begin{aligned} E &= -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i x_j + \sum_{i=1}^n x_i \theta_i + \sum_{i=1}^n \frac{1}{\tau_i} \int_0^{x_i} f^{-1}(\eta) d\eta \\ &= -\frac{1}{2} \mathbf{x}^T \mathbf{W} \mathbf{x} + \mathbf{x}^T \boldsymbol{\theta} + \sum_{i=1}^n \frac{1}{\tau_i} \int_0^{x_i} f^{-1}(\eta) d\eta \end{aligned}$$

该能量函数的表达式与离散 Hopfield 网络的定义是完全相同的。对于离散 Hopfield 网络,由于 $f(\cdot)$ 是二值函数,所以第三项的积分项为零。由于 $x_i \in (-1, 1)$ 或 $x_i \in (0, 1)$, 因此上述定义的能量函数 E 是有界的。因此只需证得 $dE/dt \leq 0$ 即可说明系统是稳定的

$$\frac{dE}{dt} = \sum_{i=1}^n \frac{\partial E}{\partial x_i} \frac{dx_i}{dt}$$

根据上述 E 的表达式可以求得

$$\begin{aligned}
\frac{\partial E}{\partial x_i} &= - \sum_{j=1}^n w_{ij} x_j + \theta_i + \frac{1}{\tau_i} f^{-1}(x_i) \\
&= - \sum_{j=1}^n w_{ij} x_j + \theta_i + \frac{1}{\tau_i} y_i \\
&= - \frac{dy_i}{dt}
\end{aligned}$$

代入上式得

$$\begin{aligned}
\frac{dE}{dt} &= \sum_{i=1}^n \left(- \frac{dy_i}{dt} \frac{dx_i}{dt} \right) = - \sum_{i=1}^n \left(\frac{dy_i}{dx_i} \frac{dx_i}{dt} \frac{dx_i}{dt} \right) \\
&= - \sum_{i=1}^n \left(\frac{dy_i}{dx_i} \left(\frac{dx_i}{dt} \right)^2 \right)
\end{aligned}$$

前面已假设 $x_i = f(y_i)$ 是单调上升函数, 如图 3.14 所示。显然它的反函数 $y_i = f^{-1}(x_i)$ 也为单调上升函数, 即有 $dy_i/dx_i > 0$ 。同时 $(dx_i/dt)^2 \geq 0$ 。因而有

$$\frac{dE}{dt} \leq 0 \quad (\text{仅当所有 } x_i \text{ 均为常数时才取等号})$$

根据李雅普诺夫稳定性理论, 该网络系统一定是渐近稳定的。即随着时间的演变, 网络状态总是朝 E 减小的方向运动, 一直到 E 取得极小值, 这时所有的 x_i 变为常数, 也即网络收敛到稳定状态。

3. 求解 TSP 问题举例

连续 Hopfield 网络主要用来进行优化计算。因此, 如何设计连接权系数及其它参数需根据具体问题来加以确定。下面以 Hopfield 神经网络应用于 TSP (Travelling Salesman Problem) 为例加以说明。

TSP 问题是人工智能中的一个难题: 推销员要到 n 个城市去推销产品, 要求推销员每个城市都要去到, 且只能去一次, 如何规划路线才能使所走的路程最短。这是一个典型的组合优化问题。下面要解决的问题是如何恰当地描述该问题, 使其适合于用 Hopfield 网络来求解。正是由于 Hopfield 成功地求解了 TSP 问题, 才使得人们对神经网络再次引起了广泛的兴趣。

设使用 n^2 个神经元结点组成如下的方阵排列 (以 $n=5$ 为例)。

市名 \ 路径	1	2	3	4	5
A	0	1	0	0	0
B	0	0	0	1	0
C	1	0	0	0	0
D	0	0	0	0	1
E	0	0	1	0	0

每个神经元采用如下的 S 形变换函数

$$x_n = \frac{1}{1 + e^{-\mu x_n}}$$

其中 $\alpha \in \{A, B, C, D, E\}, i \in \{1, 2, 3, 4, 5\}$ 。这里取较大的 μ , 以使 S 形函数比较陡峭, 从而稳态时 x_α 能够趋于 1 或趋于 0。

在方阵中 A, B, C, D, E 表示城市名称, 1, 2, 3, 4, 5 表示路径顺序。为了保证每个城市只去一次, 方阵每行只能有一个元素为 1, 其余为零。为了在某一时刻只能经一个城市, 方阵中每列也只能有一个元素为 1, 其余为零。为使每个城市必须经一次, 方阵中 1 的个数总和必须为 n 。对于所给方阵, 其相应的路径顺序为: $C \rightarrow A \rightarrow E \rightarrow B \rightarrow D \rightarrow C$, 所走的距离为 $d = d_{CA} + d_{AE} + d_{EB} + d_{BD} + d_{DC}$ 。

根据路径最短的要求及上述约束条件可以写出总的能量函数为

$$E = \rho_1 \sum_{\alpha} \sum_{\beta \neq \alpha} \sum_i d_{\alpha\beta} x_{\alpha} x_{\beta, i+1} + \frac{\rho_2}{2} \sum_i \sum_{\alpha} \sum_{\beta \neq \alpha} x_{\alpha} x_{\beta} + \frac{\rho_3}{2} \sum_{\alpha} \sum_i \sum_{j \neq i} x_{\alpha} x_{\alpha_j} + \frac{\rho_4}{2} \left(\sum_{\alpha} \sum_i x_{\alpha} - n \right)^2 + \sum_{\alpha} \sum_i \frac{1}{\tau_{\alpha}} \int_0^{x_{\alpha}} f^{-1}(\eta) d\eta$$

其中第一项反映了路径的总长度, 例如当 $\alpha = C, \beta = A, i = 1$, 且神经网络状态如上面方阵排列时, 有 $d_{\alpha\beta} x_{\alpha} x_{\beta, i+1} = d_{CA} x_{C1} x_{A2} d_{CA}$, 注意此处若 $i+1 > n$ 时, 用 1 代 $i+1$; 第二项反映了“方阵的每一列只能有一个元素为 1”的要求; 第三项反映了“方阵的每一行只能有一个元素为 1”的要求; 第四项反映了“方阵中 1 的个数总和为 n ”的要求; 第五项是 Hopfield 网络本身的要求; $\rho_1, \rho_2, \rho_3, \rho_4$ 是各项的加权系数。

根据上式可以求得

$$\frac{\partial E}{\partial x_{\alpha}} = \rho_1 \sum_{\beta \neq \alpha} d_{\alpha\beta} x_{\beta, i+1} + \rho_2 \sum_{\beta \neq \alpha} x_{\beta} + \rho_3 \sum_{j \neq i} x_{\alpha_j} + \rho_4 \left(\sum_{\alpha} \sum_i x_{\alpha} - n \right) + \frac{1}{\tau_{\alpha}} f^{-1}(x_{\alpha})$$

根据前面关于稳定性的分析, 应有如下关系成立

$$\frac{dy_{\alpha}}{dt} = - \frac{\partial E}{\partial x_{\alpha}}$$

代入上式得

$$\frac{dy_{\alpha}}{dt} = - \frac{1}{\tau_{\alpha}} y_{\alpha} - \rho_1 \sum_{\beta \neq \alpha} d_{\alpha\beta} x_{\beta, i+1} - \rho_2 \sum_{\beta \neq \alpha} x_{\beta} - \rho_3 \sum_{j \neq i} x_{\alpha_j} - \rho_4 \left(\sum_{\alpha} \sum_i x_{\alpha} - n \right)$$

令

$$\begin{cases} w_{\alpha, \beta} = - \rho_1 d_{\alpha\beta} \delta_{j, i+1} - \rho_2 \delta_{ij} (1 - \delta_{\alpha\beta}) - \rho_3 \delta_{\alpha\beta} (1 - \delta_{ij}) - \rho_4 \\ \theta_{\alpha} = - \rho_4 n \end{cases}$$

其中 $\delta_{\alpha\beta}, \delta_{ij}$ 是离散 δ 函数, 即

$$\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

从而可写出标准的 Hopfield 网络形式为

$$\begin{cases} \frac{dy_{\alpha}}{dt} = - \frac{1}{\tau_{\alpha}} y_{\alpha} + \sum_{\beta} \sum_j w_{\alpha, \beta} x_{\beta_j} - \theta_{\alpha} \\ x_{\alpha} = f(y_{\alpha}) \end{cases}$$

据此可构成连续的 Hopfield 网络,并适当地给定 $x_n(0)$ (若无先验知识,可随机给定),运行该神经网络,其稳态解即为所要求的解。

3.3.3 Boltzmann 机

Boltzmann 机是一种随机神经网络,也是一种反馈型神经网络,它在很多方面与离散 Hopfield 网类似。Boltzmann 机可用于模式分类、预测、组合优化及规划等方面。

1. 网络的结构和工作方式

在结构上,Boltzmann 机是单层的反馈网络,形式上与离散 Hopfield 网一样,具有对称的连接权系数,即 $w_{ij}=w_{ji}$ 且 $w_{ii}=0$ 。但功能上,Boltzmann 机可以看成是多层网络,其中一部分结点是输入结点,一部分是输出结点,还有一部分是隐结点。隐结点不与外界发生联系,它主要用来实现输入与输出之间的高阶联系。

Boltzmann 机是一个随机神经网络,这是与 Hopfield 网络的最大区别。图 3.16 表示了其中一个神经元结点的示意图。

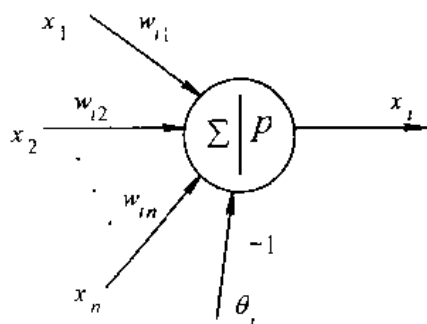


图 3.16 Boltzmann 机的单个神经元

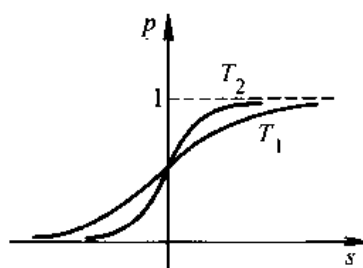


图 3.17 S 形曲线随温度的变化

它的工作方式为

$$\begin{cases} s_i = \sum_{j=1}^n w_{ij}x_j - \theta_i \\ p_i = \frac{1}{1 + e^{-s_i/T}} \end{cases}$$

其中 p_i 是 $x_i=1$ 的概率。

显然 $x_i=0$ (或 $x_i=-1$) 的概率为

$$1 - p_i = \frac{e^{-s_i/T}}{1 + e^{-s_i/T}}$$

显然 p_i 是 S 形函数。 T 越大,曲线越平坦; T 越小,曲线越陡峭。参数 T 通常称为“温度”。图 3.17 表示 S 形曲线随参数 T 变化的情况。当 $T \rightarrow 0$ 时,S 形函数便趋于二值函数,随机神经网络便退化为确定性网络,即与 Hopfield 网络具有同样的工作方式。

与 Hopfield 网一样,对 Boltzmann 机也定义网络的能量函数如下:

$$E = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij}x_i x_j + \sum_{i=1}^n x_i \theta_i = -\frac{1}{2} \mathbf{x}^T \mathbf{W} \mathbf{x} + \mathbf{x}^T \boldsymbol{\theta}$$

Boltzmann 机也有两种类型的工作方式:同步方式和异步方式。下面只考虑异步工作

方式。

若考虑第 i 个神经元的状态发生变化, 根据前面讨论 Hopfield 网络时所作的推导, 有

$$\Delta E_i = -\Delta x_i \left(\sum_{j=1}^n w_{ij} x_j - \theta_i \right) = -\Delta x_i s_i$$

这时 $x_i = 1$ 的概率

$$p_i = \frac{1}{1 + e^{-s_i/T}}$$

若 $s_i > 0$, 则 $p_i > 0.5$, 即有较大的概率取 $x_i = 1$ 。若原来 $x_i = 1$, 则 $\Delta x_i = 0, \Delta E_i = 0$; 若原来 $x_i = 0$, 则 $\Delta x_i > 0$, 而此时 $s_i > 0$, 所以 $\Delta E_i < 0$ 。

如若 $s_i < 0$, 则有较大的概率取 $x_i = 0$, 若原为 $x_i = 0$, 则 $\Delta E_i = 0$; 若原来 $x_i = 1$, 则 $\Delta x_i < 0$, 而此时 $s_i < 0$, 所以这时 $\Delta E_i < 0$ 。

不管以上何种情况, 随着系统状态的演变, 从概率的意义上, 系统的能量总是朝小的方向变化, 所以系统最后总能稳定到能量的极小点附近。由于这是随机网络, 在能量极小点附近, 系统也不会停止在某一个固定的状态。

由于神经元状态按概率取值, 因此以上分析只是从概率意义上说网络的能量总的趋势是朝着减小的方向演化, 但在有些步神经元状态可能按小概率取值, 从而使能量增加, 在有些情况下这对跳出局部极值是有好处的。这也是 Boltzmann 机与 Hopfield 网另一个不同之处。

为了有效地演化到网络能量函数的全局极小点, 通常采用模拟退火的方法。即开始采用较高的温度 T , 此时各状态出现概率的差异不大, 比较容易跳出局部极小点进入到全局极小点附近, 然后再逐渐减小温度 T , 各状态出现概率的差别逐渐拉大, 从而一方面可较准确地运动到能量的极小点, 同时阻止它跳出该最小点。

根据前面的结果, 当 x_i 由 1 变为 0 时, $\Delta x_i = -1$

则

$$\Delta E_i = E|_{x_i=0} - E|_{x_i=1} = \sum_{j=1}^n w_{ij} x_j - \theta_i = s_i$$

设 $x_i = 1$ (其它状态不变) 的概率为 p_1 , 相应的能量函数为 E_1 , $x_i = 0$ (其它状态不变) 的概率为 p_0 , 相应的能量函数为 E_0 。

$$p_1 = \frac{1}{1 + e^{-s_i/T}} = \frac{1}{1 + e^{-\Delta E_i/T}}$$

$$p_0 = 1 - p_1 = \frac{e^{-\Delta E_i/T}}{1 + e^{-\Delta E_i/T}}$$

显然有

$$\frac{p_0}{p_1} = e^{-\Delta E_i/T} = e^{-(E_0 - E_1)/T} = \frac{e^{-E_0/T}}{e^{-E_1/T}}$$

推广之, 容易得到, 对于网络中任意两个状态 α 和 β 出现的概率与它们的能量 E_α 和 E_β 之间也满足

$$\frac{p_\alpha}{p_\beta} = e^{-(E_\alpha - E_\beta)/T} = \frac{e^{-E_\alpha/T}}{e^{-E_\beta/T}}$$

这正好是 Boltzmann 分布。也就是该网络称为 Boltzmann 机的由来。

从以上结果可以看出, Boltzmann 机处于某一状态的概率主要取决于在此状态下的能量, 能量越低, 概率越大; 同时此概率还取决于温度参数 T , T 越大, 不同状态出现概率的差异便越小。较容易跳出能量的局部极小点而到达全局的极小点; T 越小时情形正相反。这也就是采用模拟退火方法寻求全局最优的原因所在。

与 Hopfield 网相似, Boltzmann 机的实际运行也分为两个阶段: 第一阶段是学习和训练阶段, 即根据学习样本对网络进行训练, 将知识分布地存储于网络的连接权中; 第二阶段是工作阶段, 即根据输入运行网络得到合适的输出, 这一步实质上是按照某种机制将知识提取出来。

2. 网络的学习和训练

网络学习的目的是通过给出一组学习样本, 经学习后得到 Boltzmann 机各种神经元之间的连接权 w_{ij} 。

设 Boltzmann 机共有 n 个结点, 其中 m 个是输入和输出结点, 其余 $r=n-m$ 个是隐结点。设 x_a 表示 m 维输入和输出结点向量, y_β 表示 r 维结点向量, $x_a \wedge y_\beta$ 表示整个网络的状态向量。同时设 $p^+(x_a)$ 表示学习样本出现的概率, 即输入输出结点约束为 x_a 时的概率。 x_a 最多可为 2^m 个, 所以 $p^+(x_a)$ 也为 2^m 个。对于学习样本数少于 2^m 个时, 可取其余未给定样本的概率为 0。设 $p^+(x_a)$ 均为已知, 且有

$$\sum_{a=1}^{2^m} p^+(x_a) = 1$$

设 $p^-(x_a)$ 表示系统无约束时出现 x_a 的概率, 对网络进行学习的目的便是调整连接权, 以使得 $p^-(x_a)$ 尽可能与 $p^+(x_a)$ 相一致。定义

$$G = \sum_{a=1}^{2^m} p^-(x_a) \ln \left[\frac{p^+(x_a)}{p^-(x_a)} \right]$$

为使 $p^-(x_a)$ 与 $p^+(x_a)$ 相一致的度量, G 是一个非负量, 仅当 $p^+(x_a) = p^-(x_a)$ 时才有 $G = 0$ 。因此下面的问题变为寻求连接权系数, 以使 G 取极小值。根据上式可以求得

$$\frac{\partial G}{\partial w_{ij}} = - \sum_{a=1}^{2^m} \frac{p^+(x_a)}{p^-(x_a)} \frac{\partial p^-(x_a)}{\partial w_{ij}}$$

根据上述假定有

$$p^-(x_a) = \sum_{\beta=1}^{2^r} p^-(x_a \wedge y_\beta) = \frac{\sum_{\beta=1}^{2^r} e^{-E_{a\beta}/T}}{\sum_{\lambda=1}^{2^m} \sum_{\mu=1}^{2^r} e^{-E_{\lambda\mu}/T}}$$

其中 $E_{a\beta}$ 表示网络状态为 $x_a \wedge y_\beta$ 时的能量, $E_{\lambda\mu}$ 表示网络状态为 $x_\lambda \wedge y_\mu$ 的能量, 即 (设神经元阈值 $\theta_i = 0$)

$$E_{a\beta} = - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i^{a\beta} x_j^{a\beta}$$

$$E_{\lambda\mu} = - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i^{\lambda\mu} x_j^{\lambda\mu}$$

其中 $x_i^{a\beta}$ 表示 n 维状态向量 $x_a \wedge y_\beta$ 的第 i 个分量, $x_i^{\lambda\mu}$ 表示 n 维状态向量 $x_\lambda \wedge y_\mu$ 的第 i 个分量, 进而求得

$$\begin{aligned}\frac{\partial p^-(x_a)}{\partial w_{ij}} &= \left[-\frac{1}{T} \sum_{\beta=1}^{2^r} e^{-E_{a\beta}/T} (-x_i^{a\beta} x_j^{a\beta}) \sum_{\lambda=1}^{2^m} \sum_{\mu=1}^{2^r} e^{-E_{\lambda\mu}/T} \right. \\ &\quad \left. + \frac{1}{T} \sum_{\lambda=1}^{2^m} \sum_{\mu=1}^{2^r} e^{-E_{\lambda\mu}/T} (-x_i^{\lambda\mu} x_j^{\lambda\mu}) \sum_{\beta=1}^{2^r} e^{-E_{a\beta}/T} \right] / \left(\sum_{\lambda=1}^{2^m} \sum_{\mu=1}^{2^r} e^{-E_{\lambda\mu}/T} \right)^2 \\ &= \frac{1}{T} \left[\sum_{\beta=1}^{2^r} p^-(x_a \wedge y_\beta) x_i^{a\beta} x_j^{a\beta} - p^-(x_a) \sum_{\lambda=1}^{2^m} \sum_{\mu=1}^{2^r} p^-(x_\lambda \wedge y_\mu) x_i^{\lambda\mu} x_j^{\lambda\mu} \right]\end{aligned}$$

将上式代入前面 $\partial G / \partial w_{ij}$ 的表达式中得

$$\begin{aligned}\frac{\partial G}{\partial w_{ij}} &= -\frac{1}{T} \left[\sum_{a=1}^{2^m} \sum_{\beta=1}^{2^r} \frac{p^+(x_a)}{p^-(x_a)} p^-(x_a \wedge y_\beta) x_i^{a\beta} x_j^{a\beta} \right. \\ &\quad \left. - \left(\sum_{a=1}^{2^m} p^+(x_a) \right) \sum_{\lambda=1}^{2^m} \sum_{\mu=1}^{2^r} p^-(x_\lambda \wedge y_\mu) x_i^{\lambda\mu} x_j^{\lambda\mu} \right]\end{aligned}$$

由于

$$\begin{aligned}p^+(x_a \wedge y_\beta) &= p^+(y_\beta | x_a) p^+(x_a) \\ p^-(x_a \wedge y_\beta) &= p^-(y_\beta | x_a) p^-(x_a) \\ p^+(y_\beta | x_a) &= p^-(y_\beta | x_a)\end{aligned}$$

所以

$$\frac{p^+(x_a)}{p^-(x_a)} p^-(x_a \wedge y_\beta) = p^+(x_a \wedge y_\beta)$$

又已知 $\sum_{a=1}^{2^m} p^+(x_a) = 1$, 代入前面式子得

$$\begin{aligned}\frac{\partial G}{\partial w_{ij}} &= -\frac{1}{T} \left[\sum_{a=1}^{2^m} \sum_{\beta=1}^{2^r} p^+(x_a \wedge y_\beta) x_i^{a\beta} x_j^{a\beta} - \sum_{\lambda=1}^{2^m} \sum_{\mu=1}^{2^r} p^-(x_\lambda \wedge y_\mu) x_i^{\lambda\mu} x_j^{\lambda\mu} \right] \\ &= -\frac{1}{T} (p_{ij}^+ - p_{ij}^-)\end{aligned}$$

其中 p_{ij}^+ 表示网络受到学习样本的约束且系统达到平衡态时第 i 和第 j 个神经元同时为 1 的概率, p_{ij}^- 表示系统为自由态且达到平衡时第 i 和第 j 个神经元同时为 1 的概率。

最后归纳出 Boltzmann 机网络学习的步骤如下。

(1) 随机设定网络的连接权初值 $w_{ij}(0)$

(2) 按照已知的概率 $p^+(x_a)$, 依次给定学习样本, 在学习样本的约束下按照模拟退火程序运行网络直至到达平衡状态, 统计出各 p_{ij}^+ 。在无约束条件下按同样的步骤并同样的次数运行网络, 统计出各 p_{ij}^- 。

(3) 按下述公式修改 w_{ij} ,

$$w_{ij}(k+1) = w_{ij}(k) + \alpha(p_{ij}^+ - p_{ij}^-) \quad \alpha > 0$$

重复上述步骤, 直到 $p_{ij}^+ - p_{ij}^-$ 小于一定的容限。

3.4 局部逼近神经网络

神经网络可以有各种分类方法。例如前面介绍了前馈神经网络和反馈神经网络,这主要是从网络的结构来划分的。神经网络控制主要应用神经网络的函数逼近功能。若从这个角度,神经网络可分为全局逼近神经网络和局部逼近神经网络。如果网络的一个或多个连接权系数或自适应可调参数在输入空间的每一点对任何一个输出都有影响,则称该神经网络为全局逼近网络。前面介绍的多层前馈网络是全局逼近网络的典型例子。对于每个输入输出数据对,网络的每一个连接权均需进行调整,从而导致全局逼近网络学习速度很慢的缺点。这个缺点对于控制来说常常是不可容忍的。若对输入空间的某个局部区域,只有少数几个连接权影响网络的输出,则称该网络为局部逼近网络。对于每个输入输出数据对,只有少量的连接权需要进行调整,从而使局部逼近网络具有学习速度快的优点,这一点对于控制来说是至关重要的。CMAC、B样条、RBF以及某些模糊神经网络是局部逼近神经网络的典型例子,下面对它们进行具体的介绍。

3.4.1 CMAC 神经网络

CMAC 网络是 J. S. Albus 于 1975 年最先提出来的,它是小脑模型关节控制器(Cerebellar Model Articulation Controller)的简称。它是仿照小脑如何控制肢体运动的原理而建立的神经网络模型,因此 CMAC 最初主要用来求解机械手的关节运动,其后进一步将它用于机器人控制、模式识别、信号处理以及自适应控制等,其中 W. T. Miller 等人已将其成功地应用于机械手的实时动态轨迹跟踪。

1. CMAC 的结构及工作原理

CMAC 由一个固定的非线性输入层和一个可调的线性输出层所组成。其结构如图 3.18 所示

设 CMAC 所要逼近的函数映射关系为

$$y = f(x)$$

其中 $x = [x_1 \ x_2 \cdots x_n]^T$, $y = [y_1 \ y_2 \cdots y_r]^T$ 。在 CMAC 中用如图所示的两个映射来实现。

(1) $S: x \rightarrow A$, 即 $\alpha = S(x)$

这是图 3.18 中输入层所实现的功能。其中 $\alpha = [\alpha_1 \ \alpha_2 \cdots \alpha_m]^T$ 是 m 维相联空间 A 中的向量。 α 的元素只取 1 或 0 两个值。对于某个特定的 x , 只有少数元素为 1, 大部分元素为 0。可见 $\alpha = S(x)$ 实现的是一个特定的非线性映射。该非线性映射是在设计网络时便确定了的。输入空间中的一个点对应于 α 中的几个元素为 1, 也即对应相联空间 A 中的一个局部区域。后面还将详细讨论该非线性映射的具体实现。

(2) $P: A \rightarrow y$, 即 $y = P(\alpha) = W\alpha$

这是图 3.18 中输出层所实现的功能。其中

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ \vdots & & & \vdots \\ w_{r1} & w_{r2} & \vdots & w_{rm} \end{bmatrix} \quad \alpha = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_m \end{bmatrix}$$

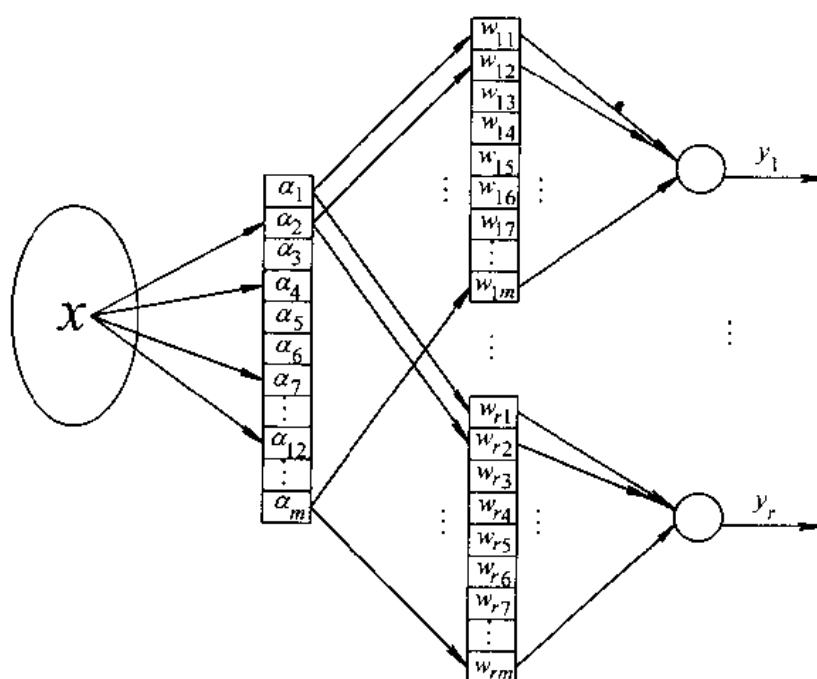


图 3.18 CMAC 的原理结构

可见输出层实现的是线性映射。其中连接权 w_{ij} ($i=1,2,\dots,r; j=1,2,\dots,m$)，是可以调整的参数。对于第 i 个输出，则有

$$y_i = P_i(\alpha) = \sum_{j=1}^m w_{ij} \alpha_j \quad i = 1, 2, \dots, r$$

类似于 BP 网络的误差反传算法,CMAC 神经网络的连接权学习算法为

$$w_{ij}(k+1) = w_{ij}(k) + \beta(y_i^d - y_i)\alpha_j/\alpha^T\alpha$$

其中 β 为学习率。可以证明当 $0 < \beta < 2$ 时可保证该迭代学习算法的收敛性(下面介绍 B 样条神经网络时还要对此详加讨论)。 y_i^d 和 y_i 分别表示第 i 个输出分量的期望值和实际值。由于相联向量 α 中只有少量几个元素为 1,其余均为零,因此在一次数据训练中只有少量的连接权需要调整。正是由于这个特点,才使得 CMAC 神经网络具有比较快的学习速度。同时,由于输入空间中的一个点对应相联空间中的一个局部区域,当输入空间中的两个点比较靠近时,它们所对应的相联空间中的局部区域也比较靠近,而且互相有重叠。正是由于这个特点,才使得 CMAC 神经网络具有局部泛化能力。

2. CMAC 输入层非线性映射的实现

当输入向量 $x = [x_1 \ x_2 \ \dots \ x_n]^T$ 时,它映射为相联空间 A 中的向量 α ,设 A^* 表示 A 中非零元素的集合,则对于图 3.18 有 $A^* = \{\alpha_2, \alpha_4, \alpha_7, \alpha_{12}\}$ 。CMAC 的输出则为这些非零元素的加权和。因此可将输入向量 x 看成为地址,输出向量 y 可看成该地址中的内容,对于任意输入 x ,若要改变其内容 y ,只需改变与 A^* 相关的连接权。

对于一般的函数映射,当输入相近时,输出也比较接近,当输入的距离较远时,相应的输出是互不相关的。为了实现这样的要求,当两个输入向量 x_i 和 x_j 距离较近时,交集 $A_i^* \cap A_j^*$ 应较大;相反,当 x_i 和 x_j 距离较远时,交集 $A_i^* \cap A_j^*$ 应比较小或为空集。输入空间

的距离用如下的海明距离来表示

$$H_{ij} = \sum_{k=1}^n |x_{ik} - x_{jk}|$$

其中 x_{ik} 和 x_{jk} 分别表示输入向量 x_i 和 x_j 的分量。也就是说,当 H_{ij} 较小时, $A_i^* \wedge A_j^*$ 应较大; H_{ij} 较大时, $A_i^* \wedge A_j^*$ 应较小或为空。为了实现这一点,必须要求相联空间的元素个数(用 $|A_p|$ 表示)应远远大于 A^* 中的元素个数(用 $|A^*|$ 表示)。通常选 $|A_p| = 100|A^*|$ 。

对于上面的选择是否能够保证对于输入空间的每个点均存在唯一的映射 $x \rightarrow A$, 是需要加以检验的。假设输入向量 x 的每个分量均可取 q 个不同的值, 则输入总共可能有 q^n 个不同的模式。设 $u = |A^*|$, $v = |A_p|/|A^*|$, 则 A^* 可能的组合为

$$C_{uv}^u = \frac{(uv)!}{u!(uv-u)!} > \frac{(uv-u)^u}{u!} > \frac{(uv-u)^u}{u^u} = (v-1)^u$$

若对上面的选择有 $v=100$, 只需 $q^n < 99^{|A^*|}$, 则一定能够保证存在唯一的映射 $x \rightarrow A$ 。

为了实现上面的要求,CMAC 输入层的非线性映射可进一步分解为如下的两步:

$$x \rightarrow M, M \rightarrow A$$

对于输入向量 x 的每个分量 $x_i (i=1, 2, \dots, n)$, 首先按下面的方法将其转换成二进制变量 m_i 。

(1) m_i 必须有且只有一个区间段的位值均为 1, 其余位值为 0。作为举例, 表 3.1 列出了一组向量 x 转换为 m 的结果, 其中 μ_i 位当 $3 \leq x \leq 6$ 时其位值为 1, 其余情况为 0。

表 3.1 $x \rightarrow m$ 映射举例

x	m											
	μ_0	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8	μ_9	μ_{10}	μ_{11}
1	1	1	1	1	0	0	0	0	0	0	0	0
2	0	1	1	1	1	0	0	0	0	0	0	0
3	0	0	1	1	1	1	0	0	0	0	0	0
4	0	0	0	1	1	1	1	0	0	0	0	0
5	0	0	0	0	1	1	1	1	0	0	0	0
6	0	0	0	0	0	1	1	1	1	0	0	0
7	0	0	0	0	0	0	1	1	1	1	0	0
8	0	0	0	0	0	0	0	1	1	1	1	0
9	0	0	0	0	0	0	0	0	1	1	1	1

(2) 在任何一个 m_i 中位值为 1 的个数(记为 $|m_i^*|$)均等于 $|A^*|$, 即 $|m_i^*| = |A^*|$ 。例如在表 3.1 中, $|m_i^*| = 4$ 。

(3) 将 m^* 中位下标名字与 x 的对应关系列成表, 如表 3.2 所示。其中只要保持同一下标名字在同一列中的位置不变, 其余次序可以是任意的。

表 3.2 $x \rightarrow m$ 映射的简化形式

x	m^*
1	a, b, c, d
2	e, b, c, d
3	e, f, c, d
4	e, f, g, d
5	e, f, g, h
6	i, f, g, h
7	i, j, g, h
8	i, j, k, h
9	i, j, k, l

对于如表 3.2 所示的一维情况,输入 x_i 与 x_j 的距离 $H_{ij} < |A_i^*|$ 时它与 $A_i^* \wedge A_j^*$ 之间有如下关系:

$$H_{ij} = |A_i^*| - |A_i^* \wedge A_j^*|$$

例如,若 $x_1=1, x_2=3$, 则 $A_1^* \wedge A_2^* = \{c, d\}$, $|A_1^*|=4$, 从而有

$$H_{12} = |A_1^*| - |A_1^* \wedge A_2^*| = 2$$

上面以一维情况为例说明了 $x \rightarrow M$ 映射方法,对于多维情况,则有

$$x \rightarrow M = \begin{cases} x_1 \rightarrow m_1^* \\ x_2 \rightarrow m_2^* \\ \vdots \\ x_n \rightarrow m_n^* \end{cases}$$

以二维情况为例,设 $x = [x_1 \ x_2]^T$, 并设 $1 \leq x_1 \leq 5$ 和 $1 \leq x_2 \leq 7$, 并选 $|A^*|=4$ 。首先按上面介绍的方法,实现如下两个映射: $x_1 \rightarrow m_1^*$ 和 $x_2 \rightarrow m_2^*$, 结果如表 3.3 所示。

表 3.3 二维向量 $x \rightarrow m^*$ 的映射

x_1	m_1^*
1	A, B, C, D
2	E, B, C, D
3	E, F, C, D
4	E, F, G, D
5	E, F, G, H
x_2	m_2^*
1	a, b, c, d
2	e, b, c, d
3	e, f, c, d
4	e, f, g, d
5	e, f, g, h
6	i, f, g, h
7	i, j, g, h

当输入向量超过一维时,将每个 m_i^* 的元素相应组合起来便可求得 A^* 。例如,若 $x_1=2$ 和 $x_2=4$,根据表 3.3 有: $m_1^*=\{E,B,C,D\}$ 和 $m_2^*=\{e,f,g,d\}$ 。于是对于 $x=[2\ 4]^T$ 得到 $A^*=\{Ee,Bf,Cg,Dd\}$ 。表 3.4 给出了所有可能的情况。

表 3.4 二维向量 $x \rightarrow A^*$ 的映射

$A^* \backslash x_1 \backslash x_2$	x_1	A,B,C,D	E,B,C,D	E,F,C,D	E,F,G,D	E,F,G,H
		1	2	3	4	5
a,b,c,d	1	Aa,Bb,Cc,Dd	Ea,Bb,Cc,Dd	Ea,Fb,Cc,Dd	Ea,Fb,Gc,Dd	Ea,Fb,Gc,Hd
e,b,c,d	2	Ae,Bb,Cc,Dd	Ee,Bb,Cc,Dd	Ee,Fb,Cc,Dd	Ee,Fb,Gc,Dd	Ee,Fb,Gc,Hd
e,f,c,d	3	Ae,Bf,Cc,Dd	Ee,Bf,Cc,Dd	Ee,Ff,Cc,Dd	Ee,Ff,Gc,Dd	Ee,Ff,Gc,Hd
e,f,g,d	4	Ae,Bf,Cg,Dd	Ee,Bf,Cg,Dd	Ee,Ff,Cg,Dd	Ee,Ff,Gg,Dd	Ee,Ff,Gg,Hd
e,f,g,h	5	Ae,Bf,Cg,Dh	Ee,Bf,Cg,Dh	Ee,Ff,Cg,Dh	Ee,Ff,Gg,Dh	Ee,Ff,Gg,Hh
i,f,g,h	6	Ai,Bf,Cg,Dh	Ei,Bf,Cg,Dh	Ei,Ff,Cg,Dh	Ei,Ff,Gg,Dh	Ei,Ff,Gg,Hh
i,j,g,h	7	Ai,Bj,Cg,Dh	Ei,Bj,Cg,Dh	Ei,Fj,Cg,Dh	Ei,Fj,Gg,Dh	Ei,Fj,Gg,Hh

根据表 3.4 可以看到,对于二维的情况也存在与一维情况同样的结果:输入向量越靠近,即 H_{ij} 越小, $|A_i^* \cap A_j^*|$ 便越大。例如 $x_1=[3\ 5]^T$ 和 $x_2=[3\ 2]^T$, $H_{12}=3$, $A_1^* \cap A_2^* = \{Ee\}$, $|A^*|=4$,从而有

$$H_{12}=3=|A^*|-|A_1^* \cap A_2^*|$$

检验表 3.4 可以发现,虽然上面的关系并不总成立,但是当 x_1 与 x_2 的距离增加时,绝不会出现 $|A_i^* \cap A_j^*|$ 也增加的情况。

上面所介绍的实现 $x \rightarrow A$ 的映射的方法可以推广至更高维的情况。而且 x_1 与 x_2 接近, $|A_i^* \cap A_j^*|$ 便越大,这个关系也总是成立的。

3. CMAC 网络存储的哈希编码

按照上面介绍的方法,当输入向量维数很高时,相联空间 A 的维数将很大,例如,若 x 的每个分量取 q 个不同的值,则 A 的维数将是 q^n ,若 $q=50$, $n=10$,则需要 50^{10} 的存储单元,这是一个太大的数目,实际上是不可能实现的。可见 CMAC 神经网络所需容量随输入维数呈指数增长,这是它的一个很大的缺点。事实上,在相联空间 A 中只有很少元素为 1,绝大多数元素为 0,因此 A 是一个稀疏矩阵。另外对于相距较远的输入向量,它们所对应 A^* 若有很小的概率产生重叠也不会产生太大的影响。鉴于此,可以将维数很大的相联空间 A 映射到一个维数少得多的空间 A_p 。哈希编码(Hash-coding)可用来实现这样的映射。

哈希编码是计算机中压缩稀疏矩阵的一个常用技术。当在一个大的存储区域稀疏地存储一些数据时,可以通过哈希编码将其压缩存储到一个小的存储区域。在这里就是要将稀疏地存储于 A 中的 A^* 通过哈希编码将其压缩存储到小的存储区域 A_p 中。具体可通过一个产生伪随机数的程序来实现,将大存储区域 A 的地址作为该伪随机数产生程序的变量,产生的随机数限制在一个较小的整数范围,该随机数便作为小存储区域 A_p 的地址。

因此 $A \rightarrow A_p$ 是一个多对少的映射。

哈希编码的这种多对少的映射特性可能导致在大存储区域的不同数据映射为小存储区域的同一地址。这个现象称为碰撞。若用程序产生的随机数的随机性能较好,则可尽量减少这种碰撞现象,但要完全避免是不可能的。

对于 CMAC 神经网络来说,这种碰撞现象称为学习的互相干扰,虽然不希望出现这个问题,但它也并不是一个十分严重的问题。它与相邻输入所产生的 A^* 重叠本质上是一样的,通过数据样本的重复训练可以解决这个问题。

另外哈希编码所产生碰撞现象的概率也是很小的。例如,若 $|A_p| = 2000$, $|A^*| = 20$, 则对于同一输入向量 x 所对应的 A^* 的不同元素映射到 A_p 中同一地址的概率为

$$\frac{1}{2000} + \frac{2}{2000} + \cdots + \frac{19}{2000} \approx 0.1$$

可见这个概率是很小的。另外当输入向量 x_1 和 x_2 相距较远,有 $A_1^* \cap A_2^* = \phi$, 但经过哈希编码后,可能出现 $A_{p1}^* \cap A_{p2}^* \neq \phi$, 即 A_{p1}^* 与 A_{p2}^* 有重叠,这是另外一种碰撞现象,也是不希望的,例如对于上面的例子 ($|A_p| = 2000$, $|A^*| = 20$), 可以求得

$$\text{prob}(|A_{p1}^* \cap A_{p2}^*| = 0) = 0.818$$

$$\text{prob}(|A_{p1}^* \cap A_{p2}^*| = 1) = 0.165$$

$$\text{prob}(|A_{p1}^* \cap A_{p2}^*| = 2) = 0.016$$

$$\text{prob}(|A_{p1}^* \cap A_{p2}^*| \geq 3) = 0.001$$

可见,对于原先无重叠的情况经哈希编码后出现重叠的概率是很小的,即使出现也只是极少量元素的重叠,它对 CMAC 的性能影响很小。

对于一个实际问题应恰当地选取 $|A^*|$ 和 $|A_p|$, $|A^*|$ 选得较小可使计算量减小,且学习速度加快,但同时也减小了泛化能力。 $|A_p|$ 太小,则增加了上面所述碰撞现象的概率,但太大则要求太多的存储容量。因而 $|A^*|$ 和 $|A_p|$ 的选择应折中考虑。

CMAC 神经网络输入层的非线性映射是事先确定的,训练时只需局部地调整输出层的连接权,而这些连接权与输出只是简单的线性关系,因此在输入数据充分激发的情况下,可保证学习算法的收敛性及快速的收敛速度。从而使该网络尤其适于自适应建模与控制。

3.4.2 B 样条神经网络

上面介绍的 CMAC 神经网络,其输出可表示为(单输出)

$$y = \sum_{j=1}^m w_j \alpha_j(x)$$

其中 $\alpha_j(x)$ 表示输入层非线性映射所得相联空间向量 α 的第 j 个分量, w_j 是与此分量相关联的连接权。上面式子也可换种角度来理解,对单输入情况, $\alpha_j(x)$ ($j=1, 2, \dots, m$) 可以看成一系列矩形基函数,两个输入时 $\alpha_j(x)$ 可看成是长方体基函数等等, w_j 则可看成是与第 j 个基函数 $\alpha_j(x)$ 相关联的连接权。例如对于如表 3.1 所示的映射,其相应的基函数可如图 3.19 所示。

由于 CMAC 神经网络是一种类似感知器的相联记忆网络,因此这样的基函数也称感

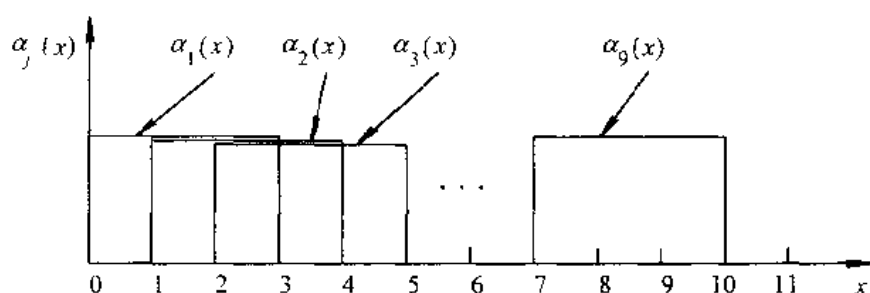
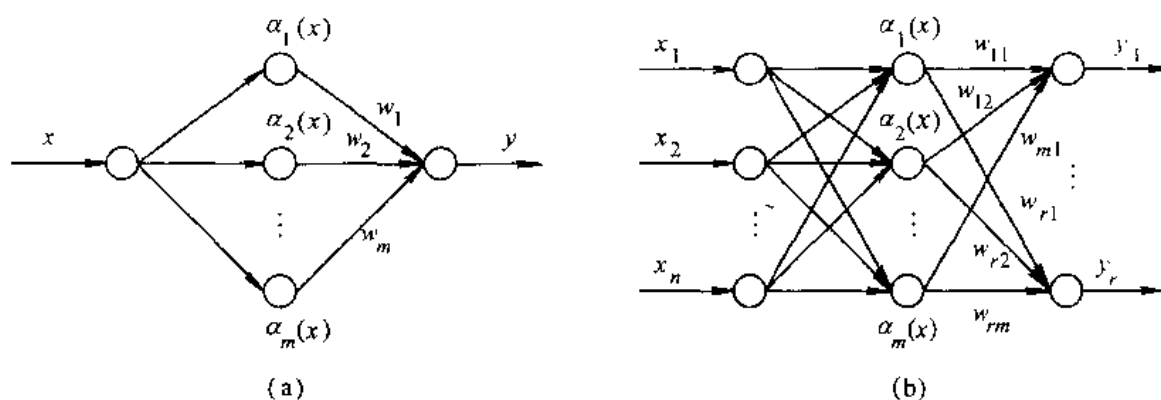


图 3.19 一维输入的矩形基函数

受域(receptive field)函数。若采用这样的观点,CMAC 神经网络也可如图 3.20 所示。



(a) 单输入单输出 (b) 多输入多输出

图 3.20 基于基函数解释的 CMAC 神经网络

由于 CMAC 神经网络的基函数形状非常简单,因而它具有实现容易、学习速度快的优点。然而也正是由于这一点,它的输出只能用台阶形函数来逼近一个光滑函数,因而逼近的精度不高。为了提高逼近精度就必须提高分辨率,而提高分辨率更进一步增加了对存储容量的要求,它们是互相矛盾的。同时,显然 CMAC 也不能学习所逼近函数的导数。下面要介绍的 B 样条神经网络便是针对 CMAC 这些缺点而设计的,有时也将它们称为 BMAC 神经网络。

1. B 样条神经网络的基函数

B 样条神经网络是基于样条函数插值的原理而设计的神经网络。因此它的基函数是由一些局部的多项式组成。例如上述的 CMAC 即可看成是由一阶基函数组成的最简单的 B 样条神经网络。CMAC 的基函数可看成是局部的零阶多项式。

(1) 输入空间的分割

为了定义 B 样条神经网络的基函数,需要首先定义对输入空间进行分割的方法。

设输入向量为 $\mathbf{x} = [x_1 \ x_2 \ \cdots \ x_n]^T$, 且 $x_i \in I_i$, I_i 为一有限区间, 定义为

$$I_i = \{x_i; x_i^{\min} \leq x_i \leq x_i^{\max}\}$$

对该区间进行如下的分割:

$$x_i^{\min} < \lambda_{i,1} \leq \lambda_{i,2} \leq \dots \leq \lambda_{i,m_i} < x_i^{\max}$$

其中 $\lambda_{i,j}$ 称为 x_i 的第 j 个内节点, 同样也可定义 $\lambda_{i,j}$ 的外节点:

$$\dots \lambda_{i,-1} \leq \lambda_{i,0} \leq x_i^{\min} \quad x_i^{\max} \leq \lambda_{i,m_i+1} \leq \lambda_{i,m_i+2} \dots$$

一般情况下, 所有左边的外节点均置于 x_i^{\min} , 所有右边的外节点均置于 x_i^{\max} 。若两节点在同一位置, 则称该节点为重节点。所有这些节点将整个区间 I_i 分为如下 m_i+1 个子区间 $I_{i,j}$ ($0 \leq j \leq m_i$)

$$I_{i,j} = x_i : \begin{cases} x_i \in [\lambda_{i,j}, \lambda_{i,j+1}) & j = 0, 1, \dots, m_i - 1 \\ x_i \in [\lambda_{i,j}, \lambda_{i,j+1}] & j = m_i \end{cases}$$

(2) 单变量基函数

首先考虑一维输入时基函数的构成, 设 $N_j^k(x)$ 为定义在 $\lambda_{j-kd}, \lambda_{j-kd-1}, \dots, \lambda_j$ 上的 k 阶基函数, 从而有感受域函数 $\alpha_j(x) = N_j^k(x)$ 。这里 d 称扩展系数。若这些节点重合在一起, 则 $N_j^k(x) = 0$, 否则它用如下的递阶关系进行计算

$$N_j^k(x) = \left(\frac{x - \lambda_{j-kd}}{\lambda_{j-d} - \lambda_{j-kd}} \right) N_{j-d}^{k-1}(x) + \left(\frac{\lambda_j - x}{\lambda_j - \lambda_{j-(k-1)d}} \right) N_j^{k-1}(x)$$

$$N_j^1(x) = \begin{cases} 1 & x \in [\lambda_{j-d}, \lambda_j) \\ 0 & x \notin [\lambda_{j-d}, \lambda_j) \end{cases}$$

上面定义的基函数具有如下的三条性质

- 正定性: $N_j^k(x) > 0, \quad x \in [\lambda_{j-d}, \lambda_j)$
- 紧密性: $N_j^k(x) = 0, \quad x \notin [\lambda_{j-d}, \lambda_j)$
- 归一性: $\frac{1}{d} \sum_j N_j^k(x) = 1, \quad x \in I$

图 3.21 画出了当 $d=2, k=1, 2, 3$ 时基函数的图形。可以看出它们的形状类似于模糊集合的隶属度函数。不难验证, 对于单节点情况, 当 $k \geq 2$ 时, $N_j^k(x)$ 以及它的直到 $(k-2)$ 阶导数在 x 的整个区间上都是连续的。

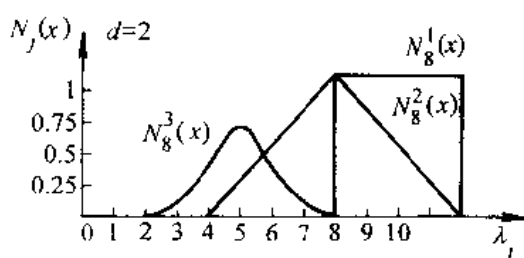


图 3.21 B 样条神经网络的基函数

根据上面的基函数的表达式, 可以求得该基函数的各阶导数为

$$\frac{d^r}{dx^r} [N_j^k(x)] \triangleq {}^{(r)}N_j^k(x) = \frac{{}^{(r-1)}N_{j-d}^{k-1}(x)}{\lambda_{j-d} - \lambda_{j-kd}} - \frac{{}^{(r-1)}N_j^{k-1}(x)}{\lambda_j - \lambda_{j-(k-1)d}}$$

$${}^{(0)}N_j^1(x) = \begin{cases} 1, & x \in [\lambda_{j-d}, \lambda_j) \\ 0, & x \notin [\lambda_{j-d}, \lambda_j) \end{cases}$$

B 样条神经网络具有如图 3.20 相同的结构, 若对于单输入单输出则有

$$y = \sum_{j=1}^m w_j a_j(x) = \sum_{j=1}^m w_j N_j^k(x)$$

从而很容易求得神经网络输出对输入的导数为

$$\frac{dy}{dx^i} \triangleq y^{(i)} = \sum_{j=1}^m w_j^{(i)} N_j^k(x)$$

这个导数信号对于控制是很有用的。

(3) 多变量基函数

设输入向量 $x \in R^n$, 则定义多维变量基函数为

$$\alpha_{j_1 j_2 \dots j_n}(x) = \prod_{i=1}^n N_{i, j_i}^{k_i}(x_i)$$

其中 $N_{i, j_i}^{k_i}(x_i)$ 表示 $x_i (i=1, 2, \dots, n)$ 的第 $j_i (j_i=1, 2, \dots, m_i)$ 个单变量基函数, 并设 x_i 的基函数的阶数为 k_i , 扩展系数为 d_i 。多维变量基函数的总数为

$$m = \prod_{i=1}^n m_i$$

可见, 多维变量基函数是每个坐标分量基函数可能的组合相乘的结果。

不难看出, 多维变量基函数也满足如下的性质

- 正定性: $\alpha_{j_1 j_2 \dots j_n}(x) > 0, \quad \forall i \quad x_i \in [\lambda_{i, j_i - k_i d_i}, \lambda_{i, j_i})$
- 紧密性: $\alpha_{j_1 j_2 \dots j_n}(x) = 0, \quad \exists i \quad x_i \in [\lambda_{i, j_i - k_i d_i}, \lambda_{i, j_i})$
- 归一性: $\frac{1}{d} \sum_{j_1=1}^{m_1} \sum_{j_2=1}^{m_2} \dots \sum_{j_n=1}^{m_n} \alpha_{j_1 j_2 \dots j_n}(x) = 1, d = \prod_{i=1}^n d_i$
- 连续性:
 设 $k = \min\{k_1, k_2, \dots, k_n\}$, 则当 $k \geq 2$ 时, $\alpha_{j_1 j_2 \dots j_n}(x)$ 以及它的直到 $(k-2)$ 阶导数在整个输入空间均连续。

图 3.22 展示出了 3 个二维变量基函数的例子, 其中 $k_1 = k_2 = k$ 分别等于 1, 2, 3。

根据以上分析可以看出, 多维 B 样条神经网络需要占用

$$m = \prod_{i=1}^n m_i$$

个存储单元, 它随着输入维数的增加而呈指数增长。所以一般也需采用如前面介绍的哈希编码方法来压缩存储量。

2. B 样条神经网络的工作原理及学习算法

B 样条神经网络的结构仍如图 3.20 所示。从图看出, 该网络从输入到输出的映射可分为如下两步:

(1) $X \rightarrow \alpha(x)$

这是图 3.20 中输入层所实现的功能, 其中 $\alpha(x) = [\alpha_1(x) \ \alpha_2(x) \ \dots \ \alpha_m(x)]^T$ 是 m 维相联空间 A 中的向量, $\alpha_j(x)$ 表示对于给定的输入 x 所对应的第 j 个基函数的输出值。由于基函数的紧密性和正定性, $\alpha(x)$ 中只有

$$\rho = \prod_{i=1}^n k_i d_i$$

个元素非零, 且其值在 0 与 1 之间, 其余元素均为零。若令 $k_i = k$ 和 $d_i = 1 (i=1, 2, \dots, n)$,

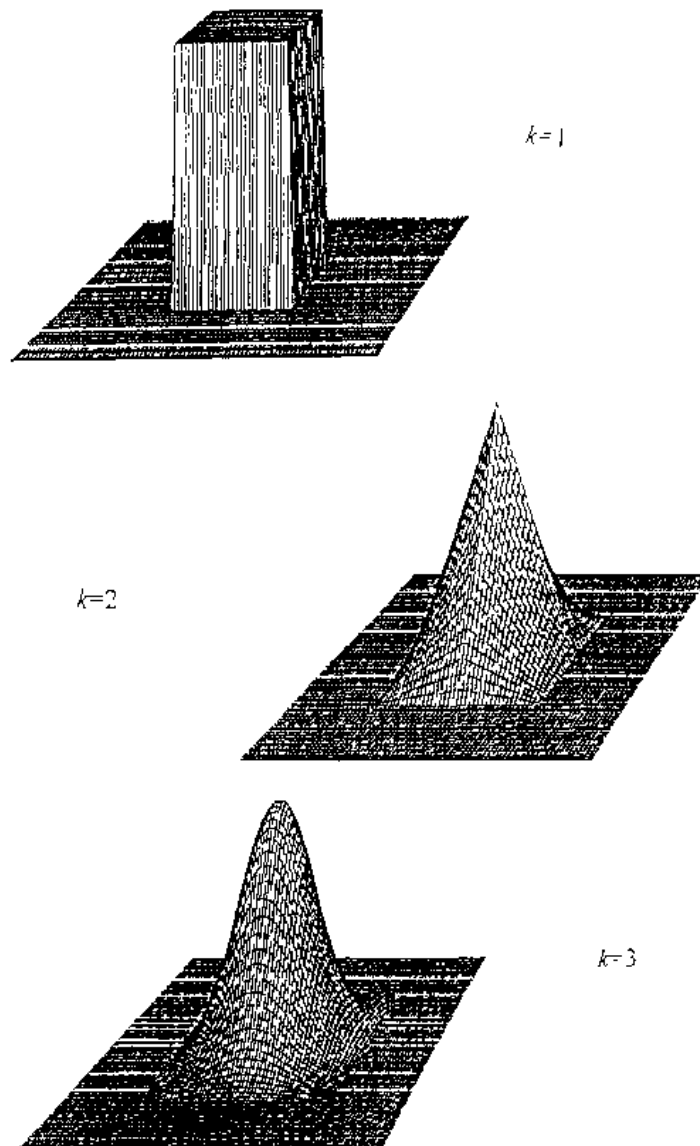


图 3.22 二维变量的基函数

则 $\rho = k^n$ 。可见, $\alpha(x)$ 中只有少数元素非零, 大部分元素为 0。根据前面关于基函数的定义, $\alpha(x)$ 是可以很容易计算出来的, 其中 $\alpha(x)$ 中的每一个非零元素, 是由 n 个单变量基函数值相乘的结果。

可见, $\alpha(x)$ 实现的是一个特定的非线性映射, 该非线性映射是设计网络时便确定了的, 即与此有关的参数 m_i, k_i 和 $d_i = 1 (i = 1, 2, \dots, n)$ 需要事先设计。

$$(2) \alpha(x) \rightarrow y \quad y = W\alpha(x)$$

这是图 3.20 中输出层所实现的功能, 其中

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ \vdots & & & \vdots \\ w_{r1} & w_{r2} & \cdots & w_{rm} \end{bmatrix} \quad \alpha(x) = \begin{bmatrix} \alpha_1(x) \\ \alpha_2(x) \\ \cdots \\ \alpha_m(x) \end{bmatrix}$$

可见输出层是线性映射。其中连接权 $w_{ij}(i=1,2,\cdots,r;j=1,2,\cdots,m)$ 是可以调整的参数,对于第 i 个输出有

$$y_i = \sum_{j=1}^m w_{ij} \alpha_j(x) \quad i=1,2,\cdots,r$$

若令 $w_i = [w_{i1} \ w_{i2} \ \cdots \ w_{im}]$ 表示矩阵 W 的第 i 行,则上式也可表示为

$$y_i = w_i \alpha(x)$$

这里采用与 CMAC 神经网络相同的连接权学习算法,即

$$w_{ij}(l+1) = w_{ij}(l) + \beta[y_i^d - y_i(l)]\alpha_j(x)/\alpha^T(x)\alpha(x)$$

也可写成如下的向量形式

$$w_i(l+1) = w_i(l) + \beta[y_i^d - y_i(l)]\alpha^T(x)/\alpha^T(x)\alpha(x)$$

其中 y_i^d 表示第 i 个输出量的期望值, $y_i(l)$ 表示第 i 个输出量第 l 次计算的的实际输出值, β 是学习率。前面在介绍 CMAC 神经网络时已给出,当 $0 < \beta < 2$ 时可确保迭代学习算法的收敛性。下面对这一点加以具体的说明。

令 $e_i(l) = y_i^d - y_i(l) = y_i^d - w_i(l)\alpha(x)$, 则

$$\begin{aligned} \Delta e_i(l) &= e_i(l+1) - e_i(l) = [y_i^d - y_i(l+1)] - [y_i^d - y_i(l)] = -[y_i(l+1) - y_i(l)] \\ &= -[w_i(l+1) - w_i(l)]\alpha(x) = -\Delta w_i(l)\alpha(x) \end{aligned}$$

上面的向量形式连接权学习算法可改写为

$$\Delta w_i(l) = \beta e_i(l)\alpha^T(x)/\alpha^T(x)\alpha(x)$$

将该式代入上面的式子可得

$$\Delta e_i(l) = -\beta e_i(l)\alpha^T(x)\alpha(x)/\alpha^T(x)\alpha(x) = -\beta e_i(l)$$

上式可进一步变为

$$e_i(l+1) = (1-\beta)e_i(l)$$

若要求上面迭代过程是稳定的,即

$$\lim_{l \rightarrow \infty} e_i(l) = 0$$

则根据离散系统的稳定性条件,必须要求 $|1-\beta| < 1$,即要求 $0 < \beta < 2$,这也就是前面给出的结论,为了使 $e_i(l)$ 单调衰减,通常只取 $0 < \beta < 1$ 。

学习率 β 在 0 到 1 之间的具体选取还需要折中地加以考虑:选取较大的 β 值可使收敛加快,从而更适合时变系统的建模,而选取较小的 β 值可以使得所建模型对测量和建模噪声较不敏感。

3.4.3 径向基函数神经网络

径向基函数神经网络也称 RBF(Radial Basis Function)神经网络,它也是一种局部逼近的神经网络。

RBF 神经网络也具有如图 3.20 所示的同样的网络结构。它与 CMAC 及 B 样条神经网络不同之处仅在于基函数的选取。

这里所选取的基函数为

$$\alpha_j(x) = \Psi_j(\|x - c_j\|/\sigma_j)$$

其中 c_j 是第 j 个基函数的中心点, σ_j 是一个可以自由选择参数, 它决定了该基函数围绕中心点的宽度, $\|x - c_j\|$ 是向量 $x - c_j$ 的范数, 它通常表示 x 和 c_j 之间的距离, Ψ_j 是一个径向对称的函数, 它在 c_j 处有一个唯一的最大值, 随着 $\|x - c_j\|$ 的增大, Ψ_j 迅速衰减到零。对于给定的输入 $x \in R^n$, 只有一小部分中心靠近 x 的处理单元被激活。

前面介绍的 B 样条基函数可以看成是一种 RBF 基函数, 但最常用的 RBF 基函数是高斯(Gaussian)基函数, 即

$$\alpha_j(x) = \Psi_j(\|x - c_j\|/\sigma_j) = e^{-\frac{\|x - c_j\|^2}{\sigma_j^2}}$$

如图 3.20 所示, RBF 神经网络也由两层组成: 输入层实现从 $x \rightarrow \alpha_j(x) = \Psi_j(\|x - c_j\|/\sigma_j)$ 的非线性映射, 输出层实现从 $\alpha_j(x)$ 到 y 的线性映射, 即

$$y = W\alpha(x)$$

或

$$y_i = \sum_{j=1}^m w_{ij} \alpha_j(x) \quad i = 1, 2, \dots, r$$

其连接权的学习算法仍同前, 即

$$w_{ij}(l+1) = w_{ij}(l) + \beta[y_i^d - y_i(l)]\alpha_j(x)/\alpha^T(x)\alpha(x)$$

由于 $\alpha_j(x)$ 为高斯函数, 因而对任意 x 均有 $\alpha_j(x) > 0$, 从而失去局部调整权值的优点。而事实上当 x 远离 c_j 时, $\alpha_j(x)$ 已非常小, 因此可作为 0 对待。因此实际上只当 $\alpha_j(x)$ 大于某一数值(例如 0.05)时才对相应的权值 w_{ij} 进行修改。经这样处理后 RBF 神经网络也同样具备局部逼近网络学习收敛快的优点。

上面学习算法中的学习率 β 的选取仍同前, 即选取 $0 < \beta < 2$ 时可确保迭代学习算法的收敛性。而实际上通常只取 $0 < \beta < 1$ 。

上述采用高斯基函数的具有如下一些优点:

- 表示形式简单, 即使对于多变量输入也不增加太多的复杂性;
- 径向对称;
- 光滑性好, 任意阶导数均存在;
- 由于该基函数表示简单且解析性好, 因而便于进行理论分析。

高斯基函数也具有如 B 样条基函数的正定性, 而不具备紧密性。这是该基函数的一个主要缺点。但采用如前面所述的近似处理($\alpha_j(x)$ 小于某一小数时即取其为 0), 可一定程度上克服该缺点。

3.5 模糊神经网络

通过前面各节的讨论可知, 神经网络具有并行计算、分布式信息存贮、容错能力强以

及具备自适应学习功能等一系列优点。正是由于这些优点,神经网络的研究受到广泛的关注并吸引了许多研究工作者的兴趣。但一般说来,神经网络不适于表达基于规则的知识,因此在对神经网络进行训练时,由于不能很好地利用已有的经验知识,常常只能将初始权值取为零或随机数,从而增加了网络的训练时间或者陷入非要求的局部极值。这应该说是神经网络的一个不足。

另一方面,模糊逻辑也是一种处理不确定性、非线性和其它不适定问题(ill-posed problem)的有力工具。它比较适合于表达那些模糊或定性的知识,其推理方式比较类似于人的思维模式。以上这些都是模糊逻辑的显著优点。但是一般说来模糊系统缺乏自学习和自适应能力。虽然第 2.8 节介绍了模糊自适应控制,但可以看出,要设计和实现模糊系统的自适应控制是比较困难的。

基于上述讨论可以想见,若能将模糊逻辑与神经网络适当地结合起来,吸取两者的长处,则可组成比单独的神经网络系统或单独的模糊系统性能更好的系统。下面介绍用神经网络来实现模糊系统的两种结构。

3.5.1 基于标准模型的模糊神经网络

在模糊系统中,模糊模型的表示主要有两种:一种是模糊规则的后件是输出量的某一模糊集合,如 NB, PB 等,这是最经常碰到的情况,因而称它为模糊系统的标准模型表示;另一种是模糊规则的后件是输入语言变量的函数,典型的情况是输入变量的线性组合。由于该方法是 Takagi 和 Sugeno 首先提出来的,因此通常称它为模糊系统的 Takagi-Sugeno 模型。下面首先讨论基于标准模型的模糊神经网络。

1. 模糊系统的标准模型

在第 2 章中已经介绍过,对于多输入多输出(MIMO)的模糊规则可以分解为多个多输入单输出(MISO)的模糊规则。因此,不失一般性,下面只讨论 MISO 模糊系统。

图 3.23 所示为一基于标准模型的 MISO 模糊系统的原理结构图。其中 $x \in R^n, y \in R$ 。如果该模糊系统的输出作用于一个控制对象,那么它的作用便是一个模糊逻辑控制器。否则,它可用于模糊逻辑决策系统、模糊逻辑诊断系统等其它方面。

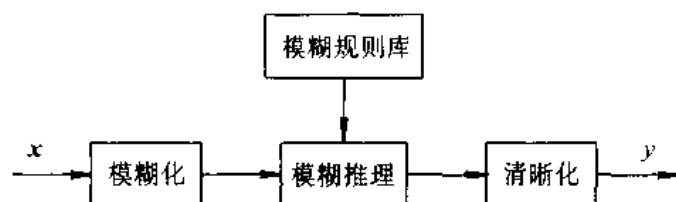


图 3.23 基于标准模型的模糊系统原理结构图

设输入向量 $x = [x_1 \ x_2 \ \cdots \ x_n]^T$, 每个分量 x_i 均为模糊语言变量, 并设

$$T(x_i) = \{A_1^i, A_2^i, \dots, A_{m_i}^i\} \quad i = 1, 2, \dots, n$$

其中 $A_j^i (j=1, 2, \dots, m_i)$ 是 x_i 的第 j 个语言变量值, 它是定义在论域 U_i 上的一个模糊集合。相应的隶属度函数为 $\mu_{A_j^i}(x_i) (i=1, 2, \dots, n; j=1, 2, \dots, m_i)$ 。

输出量 y 也为模糊语言变量且 $T(y) = \{B^1, B^2, \dots, B^{m_y}\}$ 。其中 $B^j (j=1, 2, \dots, m_y)$ 是 y

的第 j 个语言变量值,它是定义在论域 U_j 上的模糊集合。相应的隶属度函数为 $\mu_{B^j}(y)$ 。

设描述输入输出关系的模糊规则为:

R_i : 如果 x_1 是 A_1^i and x_2 是 $A_2^i \cdots$ and x_n 是 A_n^i 则 y 是 B^i

其中 $i=1,2,\cdots,m$, m 表示规则总数, $m \leq m_1 m_2 \cdots m_n$ 。

若输入量采用单点模糊集合的模糊化方法,则对于给定的输入 x ,可以求得对于每条规则的适用度为

$$\alpha_i = \mu_{A_1^i}(x_1) \wedge \mu_{A_2^i}(x_2) \cdots \wedge \mu_{A_n^i}(x_n)$$

或

$$\alpha_i = \mu_{A_1^i}(x_1) \mu_{A_2^i}(x_2) \cdots \mu_{A_n^i}(x_n)$$

通过模糊推理可得对于每一条模糊规则的输出量模糊集合 B_i 的隶属度函数为

$$\mu_{B_i}(y) = \alpha_i \wedge \mu_{B^i}(y)$$

或

$$\mu_{B_i}(y) = \alpha_i \mu_{B^i}(y)$$

从而输出量总的模糊集合为

$$B = \bigcup_{i=1}^m B_i$$

$$\mu_B(y) = \bigvee_{i=1}^m \mu_{B_i}(y)$$

若采用加权平均的清晰化方法,则可求得输出的清晰化量为

$$y = \frac{\int_{U_y} y \mu_B(y) dy}{\int_{U_y} \mu_B(y) dy}$$

由于计算上式的积分很麻烦,实际计算时通常用下面的近似公式

$$y = \frac{\sum_{i=1}^m y_{c_i} \mu_{B_i}(y_{c_i})}{\sum_{i=1}^m \mu_{B_i}(y_{c_i})}$$

其中 y_{c_i} 是使 $\mu_{B_i}(y)$ 取最大值的点,它一般也就是隶属度函数的中心点。显然

$$\mu_{B_i}(y_{c_i}) = \max_y \mu_{B_i}(y) = \alpha_i$$

从而输出量的表达式可变为

$$y = \sum_{i=1}^m y_{c_i} \bar{\alpha}_i$$

其中

$$\bar{\alpha}_i = \frac{\alpha_i}{\sum_{i=1}^m \alpha_i}$$

2. 模糊神经网络的结构

根据上面给出的模糊系统的模糊模型,可设计出如图 3.24 所示的模糊神经网络结

构。图中所示为 MIMO 系统,它是上面所讨论的 MISO 情况的简单推广。

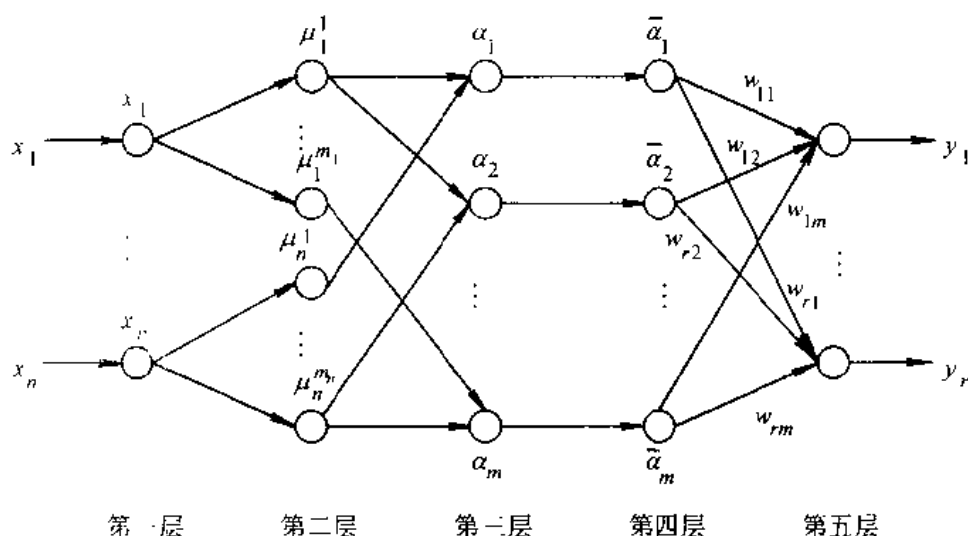


图 3.24 基于标准模型的模糊神经网络结构

图中第一层为输入层。该层的各个结点直接与输入向量的各分量 x_i 连接,它起着将输入值 $\mathbf{x}=[x_1 \ x_2 \ \cdots \ x_n]^T$ 传送到下一层的作用。该层的结点数 $N_1=n$ 。

第二层每个结点代表一个语言变量值,如 NB, PS 等。它的作用是计算各输入分量属于各语言变量值模糊集合的隶属度函数 μ_i^j ,其中

$$\mu_i^j \triangleq \mu_{A_i^j}(x_i)$$

$i=1,2,\cdots,n, j=1,2,\cdots,m_i$ 。 n 是输入量的维数, m_i 是 x_i 的模糊分割数。例如,若隶属函数采用高斯函数表示的铃形函数,则

$$\mu_i^j = e^{-\frac{(x_i - c_{ij})^2}{\sigma_{ij}^2}}$$

其中 c_{ij} 和 σ_{ij} 分别表示隶属函数的中心和宽度。该层的结点总数 $N_2 = \sum_{i=1}^n m_i$ 。

第三层的每个结点代表一条模糊规则,它的作用是用来匹配模糊规则的前件,计算出每条规则的适用度。即

$$\alpha_j = \min\{\mu_{i_1}^{j_1}, \mu_{i_2}^{j_2}, \cdots, \mu_{i_n}^{j_n}\}$$

或

$$\alpha_j = \mu_{i_1}^{j_1} \mu_{i_2}^{j_2} \cdots \mu_{i_n}^{j_n}$$

其中 $i_1 \in \{1,2,\cdots,m_1\}, i_2 \in \{1,2,\cdots,m_2\}, \cdots, i_n \in \{1,2,\cdots,m_n\}, j=1,2,\cdots,m, m = \prod_{i=1}^n m_i$ 。该层的结点总数 $N_3=m$ 。对于给定的输入,只有在输入点附近的那些语言变量值才有较大的隶属度值,远离输入点的语言变量值的隶属度或者很小(高斯隶属度函数)或者为 0(三角形隶属度函数)。当隶属度函数很小(例如小于 0.05)时近似取为 0。因此在 α_j 中只有少量结点输出非 0,而多数结点的输出为 0,这一点与前而介绍的局部逼近网络是类似的。

第四层的结点数与第三层相同,即 $N_4 = N_3 = m$,它所实现的是归一化计算,即

$$\bar{\alpha}_j = \alpha_j / \sum_{j=1}^m \alpha_j, j = 1, 2, \dots, m$$

第五层是输出层,它所实现的是清晰化计算,即

$$y_i = \sum_{j=1}^m w_{ij} \bar{\alpha}_j, j = 1, 2, \dots, r$$

与前面所给出的标准模糊模型的清晰化计算相比较,这里的 w_{ij} 相当于 y_i 的第 j 个语言值隶属函数的中心值,上式写成向量形式则为

$$y = W\bar{\alpha}$$

其中

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_r \end{bmatrix} \quad W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{r1} & w_{r2} & \cdots & w_{rm} \end{bmatrix} \quad \bar{\alpha} = \begin{bmatrix} \bar{\alpha}_1 \\ \bar{\alpha}_2 \\ \vdots \\ \bar{\alpha}_m \end{bmatrix}$$

3. 学习算法

假设各输入分量的模糊分割数是预先确定的,那么需要学习的参数主要是最后一层的连接权 $w_{ij} (i=1, 2, \dots, r; j=1, 2, \dots, m)$, 以及第二层的隶属度函数的中心值 c_{ij} 和宽度 $\sigma_{ij} (i=1, 2, \dots, n; j=1, 2, \dots, m_i)$ 。

上面所给出的模糊神经网络本质上也是一种多层前馈网络,所以可以仿照 BP 网络用误差反传的方法来设计调整参数的学习算法。为了导出误差反传的迭代算法,需要对每个神经元的输入输出关系加以形式化地描述。

设图 3.25 表示模糊神经网络中第 q 层第 j 个结点。其中结点的纯输入 $= f^{(q)}(x_1^{(q-1)}, x_2^{(q-1)}, \dots, x_{n_{q-1}}^{(q-1)}; w_{j1}^{(q)}, w_{j2}^{(q)}, \dots, w_{jn_{q-1}}^{(q)})$, 结点的输出 $= x_j^{(q)} = g^{(q)}(f^{(q)})$ 。对于一般的神经元结点,通常有

$$f^{(q)} = \sum_{i=1}^{n_{q-1}} w_{ji}^{(q)} x_i^{(q-1)}$$

$$x_j^{(q)} = g^{(q)}(f^{(q)}) = \frac{1}{1 + e^{-\mu_j f^{(q)}}}$$

而对于图 3.24 所示的模糊神经网络,其神经元结点的输入输出函数则具有较为特殊的形式。下面具体给出它的每一层的结点函数。

第一层: $f_i^{(1)} = x_i^{(0)} = x_i$ $x_i^{(1)} = g_i^{(1)} = f_i^{(1)}$
 $i = 1, 2, \dots, n$

第二层: $f_{ij}^{(2)} = -\frac{(x_i^{(1)} - c_{ij})^2}{\sigma_{ij}^2}$

$$x_{ij}^{(2)} = \mu_i = g_{ij}^{(2)} = e^{f_{ij}^{(2)}} = e^{-\frac{(x_i - c_{ij})^2}{\sigma_{ij}^2}}$$

$$i = 1, 2, \dots, n \quad j = 1, 2, \dots, m_i$$

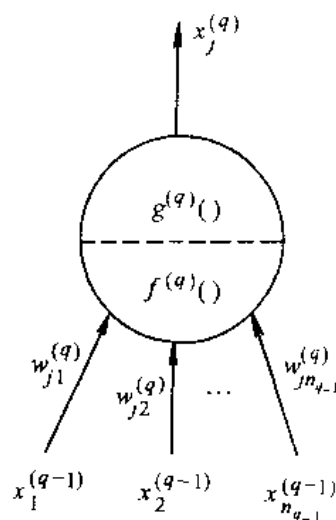


图 3.25 单个神经元结点的基本结构

第三层: $f_j^{(3)} = \min \{x_{i_1}^{(2)}, x_{i_2}^{(2)}, \dots, x_{i_n}^{(2)}\} = \min \{\mu_i^1, \mu_i^2, \dots, \mu_i^n\}$ 或者

$$f_j^{(3)} = x_{i_1}^{(2)} x_{i_2}^{(2)} \cdots x_{i_n}^{(2)} = \mu_i^1 \mu_i^2 \cdots \mu_i^n \quad x_j^{(3)} = \alpha_j = g_j^{(3)} = f_j^{(3)}$$

$$j=1, 2, \dots, m \quad m = \prod_{i=1}^n m_i$$

第四层: $f_j^{(4)} = x_j^{(3)} / \sum_{i=1}^m x_i^{(3)} = \alpha_j / \sum_{i=1}^m \alpha_i \quad x_j^{(4)} = \bar{\alpha}_j = g_j^{(4)} = f_j^{(4)}$

$$j=1, 2, \dots, m$$

第五层: $f_i^{(5)} = \sum_{j=1}^m w_{ij} x_j^{(4)} = \sum_{j=1}^m w_{ij} \bar{\alpha}_j \quad x_i^{(5)} = y_i = g_i^{(5)} = f_i^{(5)} \quad i = 1, 2, \dots, r$

设取误差代价函数为

$$E = \frac{1}{2} \sum_{i=1}^r (y_{di} - y_i)^2$$

其中 y_{di} 和 y_i 分别表示期望输出和实际输出。下面给出误差反传算法来计算 $\frac{\partial E}{\partial w_{ij}}, \frac{\partial E}{\partial c_{ij}}$ 和 $\frac{\partial E}{\partial \sigma_{ij}}$, 然后利用一阶梯度寻优算法来调节 w_{ij}, c_{ij} 和 σ_{ij} 。

首先计算

$$\delta_i^{(5)} \triangleq - \frac{\partial E}{\partial f_i^{(5)}} = - \frac{\partial E}{\partial y_i} = y_{di} - y_i$$

进而求得

$$\frac{\partial E}{\partial w_{ij}} = - \frac{\partial E}{\partial f_i^{(5)}} \frac{\partial f_i^{(5)}}{\partial w_{ij}} = - \delta_i^{(5)} x_j^{(4)} = - (y_{di} - y_i) \bar{\alpha}_j$$

再计算

$$\delta_j^{(4)} \triangleq - \frac{\partial E}{\partial f_j^{(4)}} = - \sum_{i=1}^r \frac{\partial E}{\partial f_i^{(5)}} \frac{\partial f_i^{(5)}}{\partial g_j^{(4)}} \frac{\partial g_j^{(4)}}{\partial f_j^{(4)}} = \sum_{i=1}^r \delta_i^{(5)} w_{ij}$$

$$\delta_j^{(3)} \triangleq - \frac{\partial E}{\partial f_j^{(3)}} = - \frac{\partial E}{\partial f_j^{(4)}} \frac{\partial f_j^{(4)}}{\partial g_j^{(3)}} \frac{\partial g_j^{(3)}}{\partial f_j^{(3)}} = \delta_j^{(4)} \sum_{i=1}^m x_i^{(3)} / \left(\sum_{i=1}^m x_i^{(3)} \right)^2 = \delta_j^{(4)} \sum_{i=1}^m \alpha_i / \left(\sum_{i=1}^m \alpha_i \right)^2$$

$$\delta_{ij}^{(2)} \triangleq - \frac{\partial E}{\partial f_{ij}^{(2)}} = - \sum_{k=1}^m \frac{\partial E}{\partial f_k^{(3)}} \frac{\partial f_k^{(3)}}{\partial g_{ij}^{(2)}} \frac{\partial g_{ij}^{(2)}}{\partial f_{ij}^{(2)}} = \sum_{k=1}^m \delta_k^{(3)} S_{ij} e^{f_{ij}^{(2)}} = \sum_{k=1}^m \delta_k^{(3)} S_{ij} e^{-\frac{(x_i - c_{ij})^2}{\sigma_{ij}^2}}$$

当 $f^{(3)}$ 采用取小运算时, 则当 $g_{ij}^{(2)} = \mu_i^k$ 是第 k 个规则结点输入的最小值时

$$S_{ij} = \frac{\partial f_k^{(3)}}{\partial g_{ij}^{(2)}} = \frac{\partial f_k^{(3)}}{\partial \mu_i^k} = 1$$

否则

$$S_{ij} = \frac{\partial f_k^{(3)}}{\partial g_{ij}^{(2)}} = \frac{\partial f_k^{(3)}}{\partial \mu_i^k} = 0$$

当 $f^{(3)}$ 采用相乘运算时, 则当 $g_{ij}^{(2)} = \mu_i^k$ 是第 k 个规则结点的一个输入时

$$S_{ij} = \frac{\partial f_k^{(3)}}{\partial g_{ij}^{(2)}} = \frac{\partial f_k^{(3)}}{\partial \mu_i^k} = \prod_{\substack{j=1 \\ j \neq i}}^n \mu_j^k$$

否则

$$S_{ij} = \frac{\partial f_k^{(3)}}{\partial g_{ij}^{(2)}} = \frac{\partial f_k^{(3)}}{\partial \mu_i'} = 0$$

从而可得所求一阶梯度为

$$\begin{aligned}\frac{\partial E}{\partial c_{ij}} &= \frac{\partial E}{\partial f_{ij}^{(2)}} \frac{\partial f_{ij}^{(2)}}{\partial c_{ij}} = -\delta_{ij}^{(2)} \frac{2(x_i - c_{ij})}{\sigma_{ij}^2} \\ \frac{\partial E}{\partial \sigma_{ij}} &= \frac{\partial E}{\partial f_{ij}^{(2)}} \frac{\partial f_{ij}^{(2)}}{\partial \sigma_{ij}} = -\delta_{ij}^{(2)} \frac{2(x_i - c_{ij})^2}{\sigma_{ij}^3}\end{aligned}$$

在求得所需的一阶梯度后,最后可给出参数调整的学习算法为

$$w_{ij}(k+1) = w_{ij}(k) - \beta \frac{\partial E}{\partial w_{ij}} \quad i = 1, 2, \dots, r \quad j = 1, 2, \dots, m$$

$$c_{ij}(k+1) = c_{ij}(k) - \beta \frac{\partial E}{\partial c_{ij}} \quad i = 1, 2, \dots, n \quad j = 1, 2, \dots, m,$$

$$\sigma_{ij}(k+1) = \sigma_{ij}(k) - \beta \frac{\partial E}{\partial \sigma_{ij}} \quad i = 1, 2, \dots, n \quad j = 1, 2, \dots, m,$$

其中 $\beta > 0$ 为学习率。

该模糊神经网络也和 BP 网络及 CMAC 等一样,本质上也是实现从输入到输出的非线性映射。它和 BP 网络一样,结构上都是多层前馈网,学习算法都是通过误差反传的方法;它和 CMAC 等一样,都属于局部逼近网络。下面通过一个非线性函数映射的例子来说明该网络的性能及它与标准 CMAC 的比较。

4. 举例

设有如下的二维非线性函数

$$f(x_1, x_2) = \sin(\pi x_1) \cos(\pi x_2)$$

其中 $x_1 \in [-1, 1], x_2 \in [-1, 1]$ 。现用上面给出的模糊神经网络来实现该非线性映射。

设将输入量 x_1 和 x_2 均分为 8 个模糊等级,它们对应于从 NL 到 PL 的 8 个模糊语言名称,即 $m_1 = m_2 = 8$ 。取各个模糊等级的隶属度函数,如图 3.26 所示。这里假设隶属度函

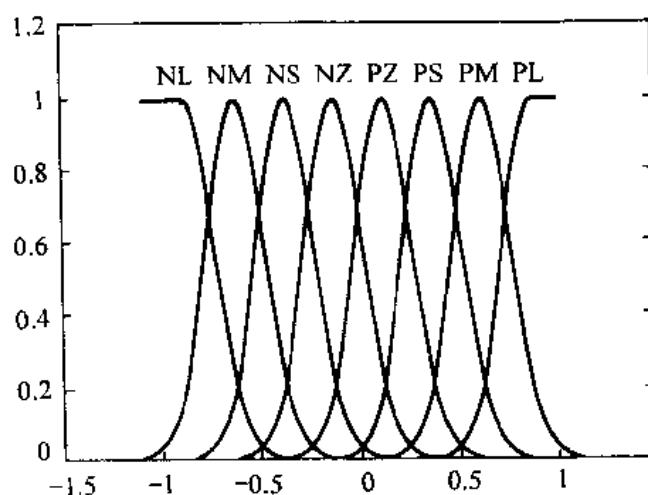


图 3.26 输入 x_1 和 x_2 隶属度函数

数的形状已经预先给定,所要调整的参数只是输出层的连接权值 $w_i (i=1,2,\dots,m)$ 。

为了对模糊神经网络进行训练,需要选取训练样本。这里按 $\Delta x_1 = \Delta x_2 = 0.1$ 的间隔均匀取点,用上面的解析式进行理论计算,得到 400 组输入输出的样本数据。利用这些样本数据对图 3.24 所示的模糊神经网络(其中 $n=2, m_1=m_2=8, m=64, r=1$)进行训练,通过调整输出层的连接权 $w_i (i=1,2,\dots,m)$,最后得到误差的学习曲线如图 3.27(a)所示,图中示出了学习率 $\beta=0.70$ 和 0.25 两种情况,显然取较大的 β 收敛也较快。图 3.27(b)所示为经 20 次学习后模糊神经网络所实现的输入输出映射的三维图形。为了显示网络的泛化能力,计算网络输出时采用了不同于训练时的数据(取 $\Delta x_1 = \Delta x_2 = 0.12$)。为了比较,图中同时给出了按解析式计算得到的理论结果的三维图形。可以看出,模糊神经网络的逼近效果是很好的。

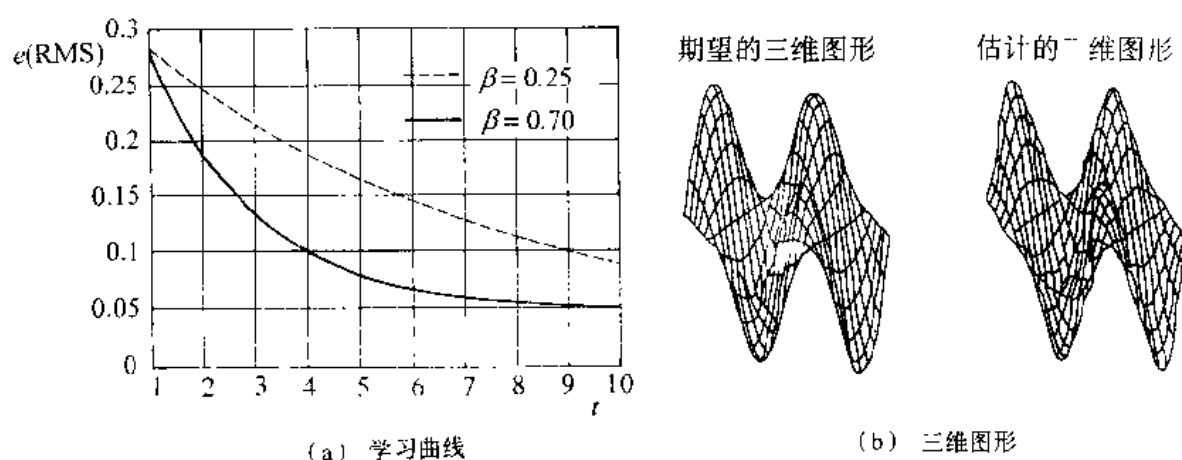


图 3.27 模糊神经网络的误差学习曲线及输入输出的三维图形

为了将模糊神经网络与 CMAC 进行比较,图 3.28 显示了这两种神经网络在相同样

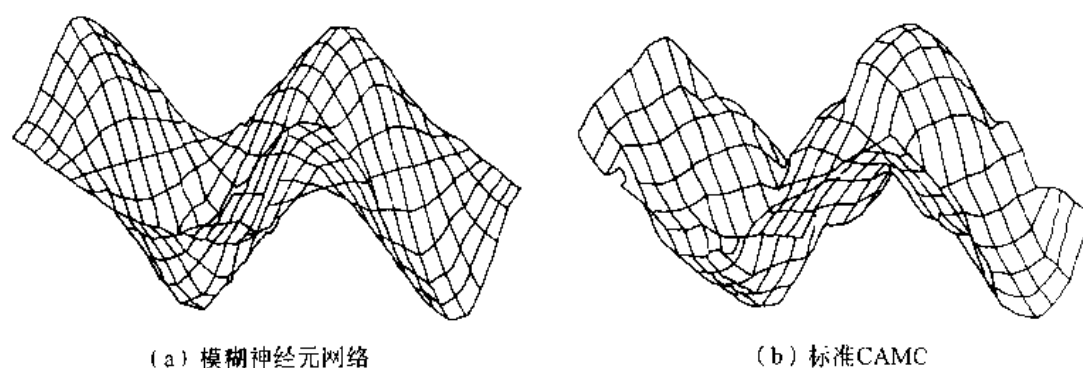


图 3.28 模糊神经网络与 CMAC 的比较

本集下对该函数进行逼近的比较。可以看出,模糊神经网络给出了比 CMAC 明显好的结果:模糊神经网络的最终逼近误差(均方根误差)为 0.0526,而 CMAC 的最终逼近误差为 0.3502。

3.5.2 基于 Takagi-Sugeno 模型的模糊神经网络

1. 模糊系统的 Takagi-Sugeno 模型

由于 MIMO 的模糊规则可分解为多个 MISO 模糊规则,因此下面也只讨论 MISO 模糊系统的模型。

设输入向量 $x = [x_1 \ x_2 \ \cdots \ x_n]^T$, 每个分量 x_i 均为模糊语言变量。并设

$$T(x_i) = \{A_i^1, A_i^2, \dots, A_i^{m_i}\} \quad i = 1, 2, \dots, n$$

其中 $A_i^j (j=1, 2, \dots, m_i)$ 是 x_i 的第 j 个语言变量值,它是定义在论域 U_i 上的一个模糊集合。相应的隶属度函数为 $\mu_{A_i^j}(x_i) (i=1, 2, \dots, n; j=1, 2, \dots, m_i)$ 。

Takagi 和 Sugeno 所提出的模糊规则后件是输入变量的线性组合,即

R_j : 如果 x_1 是 A_1^j and x_2 是 $A_2^j \cdots$ and x_n 是 A_n^j , 则

$$y_j = p_{j0} + p_{j1}x_1 + \cdots + p_{jn}x_n$$

其中 $j = 1, 2, \dots, m, m \leq \prod_{i=1}^n m_i$ 。

若输入量采用单点模糊集合的模糊化方法,则对于给定的输入 x ,可以求得对于每条规则的适用度为

$$\alpha_j = \mu_{A_1^j}(x_1) \wedge \mu_{A_2^j}(x_2) \wedge \cdots \wedge \mu_{A_n^j}(x_n)$$

或

$$\alpha_j = \mu_{A_1^j}(x_1) \mu_{A_2^j}(x_2) \cdots \mu_{A_n^j}(x_n)$$

模糊系统的输出量为每条规则的输出量的加权平均,即

$$y = \frac{\sum_{j=1}^m \alpha_j y_j}{\sum_{j=1}^m \alpha_j} = \sum_{j=1}^m \bar{\alpha}_j y_j$$

其中

$$\bar{\alpha}_j = \alpha_j / \sum_{j=1}^m \alpha_j$$

2. 模糊神经网络的结构

根据上面给出的模糊模型,可以设计出如图 3.29 所示的模糊神经网络结构。图中所示为 MIMO 系统,它是上面讨论的 MISO 系统的简单推广。

由图可见,该网络由前件网络和后件网络两部分组成,前件网络用来匹配模糊规则的前件,后件网络用来产生模糊规则的后件。

(1) 前件网络

前件网络由 4 层组成。第一层为输入层。它的每个结点直接与输入向量的各分量 x_i 连接,它起着将输入值 $x = [x_1 \ x_2 \ \cdots \ x_n]^T$ 传送到下一层的作用。该层的结点数 $N_1 = n$ 。

第二层每个结点代表一个语言变量值,如 NM, PS 等。它的作用是计算各输入分量属于各语言变量值模糊集合的隶属度函数 μ_i^j , 其中

$$\mu_i^j \triangleq \mu_{A_i^j}(x_i)$$

$i = 1, 2, \dots, n, j = 1, 2, \dots, m_i$ 。 n 是输入量的维数, m_i 是 x_i 的模糊分割数。例如,若隶属函数采用高斯函数表示的铃形函数,则

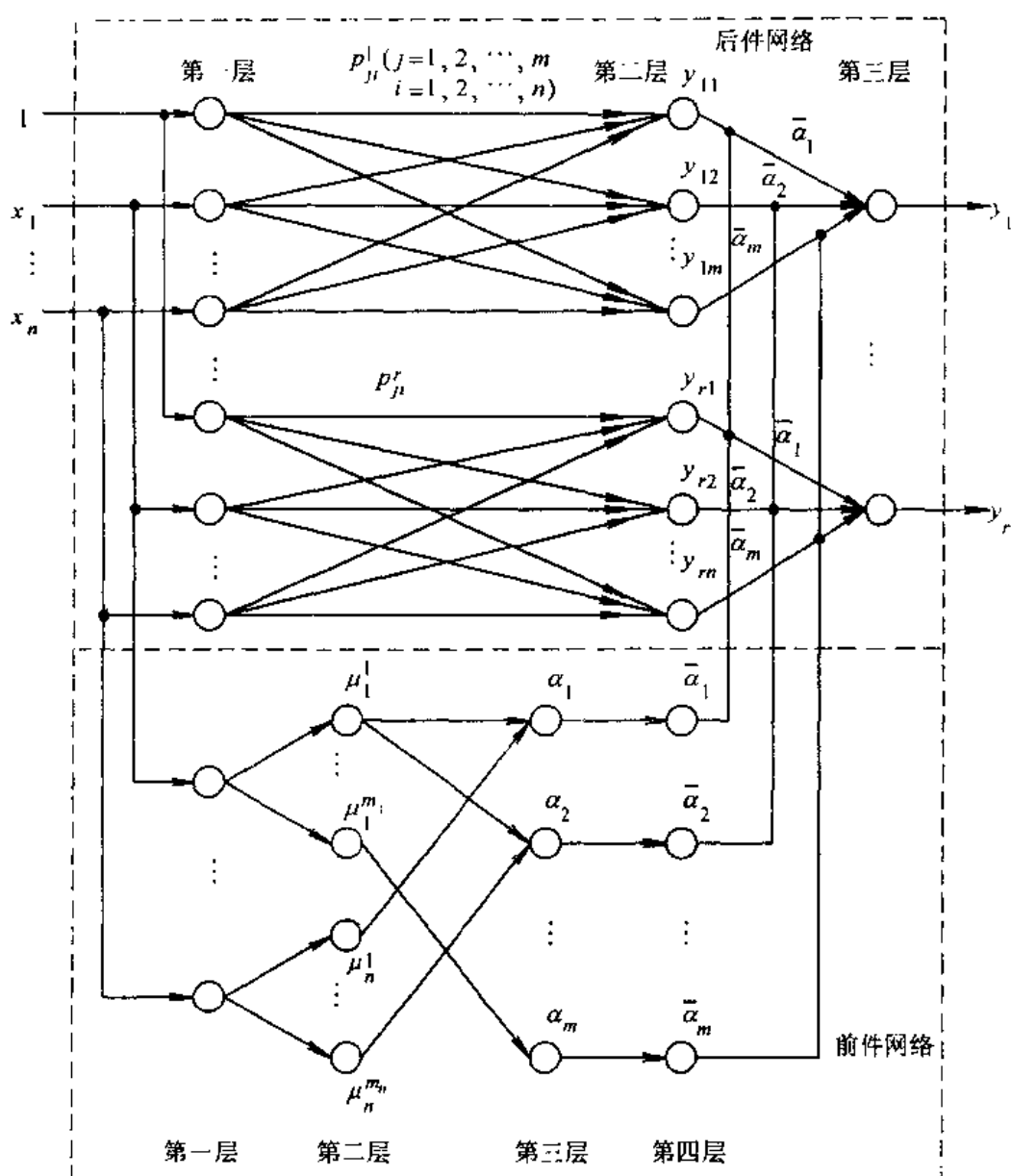


图 3.29 基于 Takagi-Sugeno 模型的模糊神经网络结构

$$\mu_i^j = e^{-\frac{(x_i - c_{ij})^2}{\sigma_{ij}^2}}$$

其中 c_{ij} 和 σ_{ij} 分别表示隶属函数的中心和宽度。该层的结点总数 $N_2 = \sum_{i=1}^n m_i$ 。

第三层的每个结点代表一条模糊规则,它的作用是用来匹配模糊规则的前件,计算出每条规则的适用度。即

$$\alpha_j = \min\{\mu_1^1, \mu_2^1, \dots, \mu_n^1\}$$

或

$$\alpha_j = \mu_1^1 \mu_2^1 \dots \mu_n^1$$

其中 $i_1 \in \{1, 2, \dots, m_1\}, i_2 \in \{1, 2, \dots, m_2\}, \dots, i_n \in \{1, 2, \dots, m_n\}, j = 1, 2, \dots, m, m = \prod_{i=1}^n m_i$ 。该层的结点总数 $N_3 = m$ 。对于给定的输入, 只有在输入点附近的语言变量值才有较大的隶属度值, 远离输入点的语言变量值的隶属度或者很小(高斯隶属度函数)或者为 0(三角形隶属度函数)。当隶属度函数很小(例如小于 0.05)时近似取为 0。因此在 α_j 中只有少量结点输出非 0, 而多数结点的输出为 0, 这一点类似于局部逼近网络。

第四层的结点数与第三层相同, 即 $N_4 = N_3 = m$, 它所实现的是归一化计算, 即

$$\bar{\alpha}_j = \alpha_j / \sum_{i=1}^m \alpha_i \quad j = 1, 2, \dots, m$$

(2) 后件网络

后件网络由 r 个结构相同的并列子网络所组成, 每个子网络产生一个输出量。

子网络的第一层是输入层, 它将输入变量传送到第二层。输入层中第 0 个结点的输入值 $x_0 = 1$, 它的作用是提供模糊规则后件中的常数项。

子网络的第二层共有 m 个结点, 每个结点代表一条规则, 该层的作用是计算每一条规则的后件, 即

$$y_{ij} = p'_{j0} + p'_{j1}x_1 + \dots + p'_{jn}x_n = \sum_{k=0}^n p'_{jk}x_k \quad i = 1, 2, \dots, r \quad j = 1, 2, \dots, m$$

子网络的第三层是计算系统的输出, 即

$$y_i = \sum_{j=1}^m \bar{\alpha}_j y_{ij} \quad i = 1, 2, \dots, r$$

可见, y_i 是各规则后件的加权和, 加权系数为各模糊规则的经归一化的适用度, 也即前件网络的输出用作后件网络第三层的连接权值。

至此, 图 3.29 所示的神经网络完全实现了 Takagi-Sugeno 的模糊系统模型。

3. 学习算法

假设各输入分量的模糊分割数是预先确定的, 那么需要学习的参数主要是后件网络的连接权 $p_{ji}^k (j=1, 2, \dots, m; i=0, 1, \dots, n; k=1, 2, \dots, r)$ 以及前件网络第二层各结点隶属函数的中心值 c_{ij} 及宽度 $\sigma_{ij} (i=1, 2, \dots, m; j=1, 2, \dots, m_i)$ 。

设取误差代价函数为

$$E = \frac{1}{2} \sum_{i=1}^r (y_{di} - y_i)^2$$

其中 y_{di} 和 y_i 分别表示期望输出和实际输出。下面首先给出参数 p_{ji}^k 的学习算法。

$$\frac{\partial E}{\partial p_{ji}^k} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial y_{kj}} \frac{\partial y_{kj}}{\partial p_{ji}^k} = -(y_{di} - y_i) \bar{\alpha}_j x_i$$

$$p_{ji}^k(l+1) = p_{ji}^k(l) - \beta \frac{\partial E}{\partial p_{ji}^k} = p_{ji}^k(l) + \beta (y_{di} - y_i) \bar{\alpha}_j x_i$$

其中 $j=1, 2, \dots, m; i=0, 1, \dots, n; k=1, 2, \dots, r$ 。

下面讨论 c_{ij} 及 σ_{ij} 的学习问题, 这时可将参数 p_{ji}^k 固定。从而图 3.29 可以简化为如图 3.30 所示。这时每条规则的后件在简化结构中变成了最后一层的连接权。

比较图 3.30 与图 3.31 可以发现, 该简化结构与基于标准模型的模糊神经网络具有

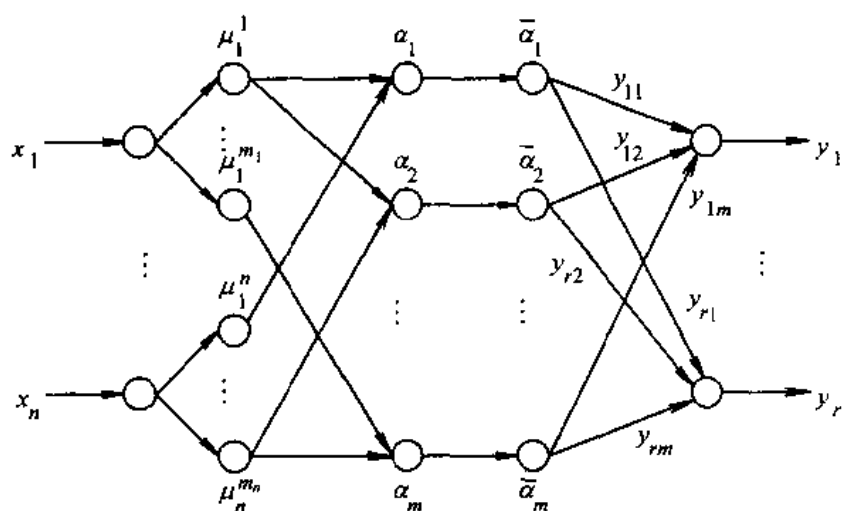


图 3.30 基于 Takagi-Sugeno 模型的模糊神经网络简化结构

完全相同的结构,这时只需令最后一层的连接权 $y_{ij}=w_{ij}$,则完全可以借用前面已得的结果,即

$$\begin{aligned}\delta_i^{(5)} &= y_{di} - y_i, \quad i = 1, 2, \dots, n \\ \delta_j^{(4)} &= \sum_{i=1}^r \delta_i^{(5)} y_{ij}, \quad j = 1, 2, \dots, m \\ \delta_j^{(3)} &= \delta_j^{(4)} \sum_{\substack{i=1 \\ j \neq 1}}^m \alpha_i / \left(\sum_{i=1}^m \alpha_i \right)^2, \quad j = 1, 2, \dots, m \\ \delta_{ij}^{(2)} &= \sum_{k=1}^m \delta_k^{(3)} S_{ij} e^{-\frac{(x_i - c_{ij})^2}{\sigma_{ij}^2}}, \quad i = 1, 2, \dots, n \quad j = 1, 2, \dots, m_i\end{aligned}$$

其中当 and 采用取小运算时,则当 μ_i^k 是第 k 个规则结点输入的最小值时

$$S_{ij} = 1$$

否则

$$S_{ij} = 0$$

当 and 采用相乘运算时,则当 μ_i^k 是第 k 个规则结点的一个输入时

$$S_{ij} = \prod_{\substack{j=1 \\ j \neq i}}^n \mu_i^j$$

否则

$$S_{ij} = 0$$

最后求得

$$\begin{aligned}\frac{\partial E}{\partial x_{ij}} &= -\delta_{ij}^{(2)} \frac{2(x_i - c_{ij})}{\sigma_{ij}^2} \\ \frac{\partial E}{\partial \sigma_{ij}} &= -\delta_{ij}^{(2)} \frac{2(x_i - c_{ij})^2}{\sigma_{ij}^3}\end{aligned}$$

$$c_{ij}(k+1) = c_{ij}(k) - \beta \frac{\partial E}{\partial c_{ij}}$$

$$\sigma_{ij}(k+1) = \sigma_{ij}(k) - \beta \frac{\partial E}{\partial \sigma_{ij}}$$

其中 $\beta > 0$ 为学习率, $i=1, 2, \dots, n; j=1, 2, \dots, m_i$ 。

对于上面介绍的两种模糊神经网络,当给定一个输入时网络(或前件网络)第三层的 $\alpha = [\alpha_1 \ \alpha_2 \ \dots \ \alpha_m]^T$ 中只有少量元素非 0,其余大部分元素均为 0。因而,从 x 到 α 的映射与 CMAC、B 样条及 RBF 神经网络的输入层的非线性映射非常类似。所以该模糊神经网络也是局部逼近网络。其中第二层的隶属度函数类似于感受域函数或基函数。

模糊神经网络虽然也是局部逼近网络,但是它是按照模糊系统模型建立的,网络中的各个结点及所有参数均有明显的物理意义,因此这些参数的初值可以根据系统的模糊或定性的知识来加以确定,然后利用上述的学习算法可以很快收敛到要求的输入输出关系。这是模糊神经网络比前面单纯的神经网络的优点所在。同时由于它具有神经网络的结构,因而参数的学习和调整比较容易,这是它比单纯的模糊逻辑系统的优点所在。

基于 Takagi-Sugeno 模型的模糊神经网络可以从另一角度来认识它的输入输出映射关系,若各输入分量的分割是精确的,即相当于隶属度函数为互相拼接的超矩形函数,则网络的输出相当于是原光滑函数的分段线性近似,即相当于用许多块超平面来拟合一个光滑曲面。网络中的 p_{ij}^k 参数便是这些超平面方程的参数,这样只有当分割越精细时,拟合才能越准确。而实际上这里的模糊分割互相之间是有重叠的,因此即使模糊分割数不多,也能获得光滑和准确的曲面拟合。基于上面的理解,可以帮助我们选取网络参数的初值。例如,若根据样本数据或根据其它先验知识已知输出曲面的大致形状时,可根据这些形状来进行模糊分割。若某些部分曲面较平缓,则相应部分的模糊分割可粗些;反之若某些部分曲面变化剧烈,则相应部分的模糊分割需要精细些。在各分量的模糊分割确定后,可根据各分割子区域所对应的曲面形状用一个超平面来近似,这些超平面方程的参数即作为 p_{ij}^k 的初值。由于网络还要根据给定样本数据进行学习和训练。因而初值参数的选择并不要求很精确。但是根据上述的先验知识所作的初步选择却是非常重要的,它可避免陷入不希望的局部极值并大大提高收敛的速度,这一点对于实时控制是尤为重要的。

3.6 基于神经网络的系统建模与辨识

前面各节我们介绍了各种神经网络。从本节开始,我们将着重讨论神经网络在系统建模、辨识与控制中的应用。

3.6.1 概述

人工或计算神经网络迄今已经历了半个世纪的研究,出现了数十种主要的网络结构和各种各样的网络学习算法。前面已指出,其中最具有代表性的有:MP 模型(McCulloch and Pitts, 1943)、Hebb 规则(Hebb, 1949)、Perceptron 感知机(Rosenblatt, 1957)、Adaline (Widrow, 1960)、Hopfield 网络(Hopfield 等, 1984; 1986)、BP 网络(Rumelhart 等,

1985)、Boltzmann 机及高阶 Boltzmann 机(Hinton 等,1984;1985)、ART(Grossberg 等,1986)、SOM(Kohonen,1984)、CMAC(Albus,1975)、BMAC(Lane,1992)、RBF(Sauner and Slotine,1984)、动态 BP 网络(Werbos,1990)、Elman 网络(Elman,1990)、Jordan 网络(Jordan,1986),以及模糊神经网络等。相应的学习算法包括:多层前馈神经网络的标准 BP 算法(SBP)及其各种变形、自适应变步长学习算法、最小二乘学习算法、二阶学习算法、高阶快速学习算法(如基于 Kalman 滤波的学习算法),以及能克服局部极小值的趋药性(Chemotaxis)算法和遗传算法(GA);Hopfield 网络的不动点学习算法、轨迹(Trajectory)学习算法和动态规划学习算法;Boltzmann 机的模拟退火和快速模拟退火学习算法;多层网络的竞争学习算法;ART 网络的 Grossberg 学习算法;SOM 网络的 Kohonen 学习算法;Necogitron 网络的福岛(Fukushima)学习算法等;以及模糊神经网络的再励学习算法与梯度学习算法等。

神经网络在系统建模、辨识与控制中的应用,大致以 1985 年 Rumelhart 的突破性研究为界。在极短的时间内,神经网络就以其独特的非传统表达方式和固有的学习能力,引起了控制界的普遍重视,并取得了一系列重要结果。迄今已经覆盖了控制理论中的绝大多数问题,如系统建模与辨识、PID 参数的整定、极点配置、内模控制、优化设计、预测控制、最优控制、自适应控制、滤波与预测、容错控制、模糊控制、专家控制和学习控制等。它甚至还应用于与控制有关的其他问题,如 A/D、D/A 转换、矩阵求逆、Jacobian 矩阵计算、QR 分解、Lyapunov 方程和 Riccati 方程求解等。

神经网络应用于控制领域的主要吸引力在于:

- 多层前馈神经网络能够以任意精度逼近任意非线性映射,给复杂系统的建模带来了一种新的、非传统的表达工具;
- 固有的学习能力降低了不确定性,增加了适应环境变化的泛化能力;
- 并行计算特点,使其有潜力快速实现大量复杂的控制算法(目前还有待于神经网络芯片技术的进步);
- 分布式信息存储与处理结构,从而具有独特的容错性;
- 能够同时融合定量与定性数据,使其能够利用连接主义的结构,与传统控制方法及符号主义的人工智能相结合;
- 多输入多输出的结构模型可方便地应用于多变量控制系统。

然而,经过 1989 年前后出现的研究高潮以后,单纯使用神经网络的辨识与控制方法的研究,目前已有停滞不前的趋势。究其原因,主要是因为:(1)近年来,神经网络本身的研究,如网型等未再有根本的突破,专门适合于控制问题的动态神经网络仍待进一步发展;(2)神经网络的泛化能力不足,制约了控制系统的鲁棒性;(3)网络本身的黑箱式内部知识表达方式,使其不能利用初始经验进行学习,易于陷入局部极小值;(4)分布并行计算的潜力还有赖于硬件实现技术的进步。

另一方面,模糊逻辑理论作为一种符号处理方法,为人类抽象的认知过程,如思维和推理等深度智能提供了较为系统的数学基础,能够模拟人类的某些定性的语言属性,可表达认知不确定性下的推理机制,但缺乏有效的学习算法;而连接主义的神经网络虽有很强的学习能力,但就其本质来说,一般只能模拟人类低层的感知能力,相比之下仅有较低的

智能,因此神经网络根本无法达到人们原来寄予的过高期望,即能够完全超越已陷入困境的经典人工智能方法,使基于神经网络的智能控制方法取代或普遍优于其它智能控制系统。

为了提供与生物神经系统智能控制行为具有更大相似性的智能控制方法,同时也为了克服神经网络控制方法的前述困难,目前将模糊逻辑(包括专家系统)与神经网络结合,发展模糊神经网络控制方法,已经成为模糊控制或神经网络控制研究的主要发展趋势。这种将两者以递阶方式融合起来的新网络模型及其控制方法,能够提供更加有效的智能行为、学习能力、自适应特点、并行机制和高度灵活性,使其能够更成功地处理各种不确定的、复杂的、不精确的和近似的控制问题。

在系统介绍神经网络控制方法之前,我们将首先讲述神经网络在系统建模与辨识中的应用,以此作为前者的基础和必要准备。在本节,我们将从函数逼近理论的角度,首先讨论神经网络对非线性映射的逼近能力。然后介绍利用多层静态网络,建立非线性系统的正向与逆动力学模型。最后给出一种利用部分递归网络辨识动态时间系统的方法。

3.6.2 逼近理论与网络建模

作为一种非传统的表达方式,神经网络可用来建立系统的输入输出模型,它们或者作为被控对象的正向或逆动力学模型,或者建立控制器的逼近模型,或者用以描述性能评价估计器。

状态空间表达式可以完全描述线性系统的全部动态行为,也可给出非线性系统的一般但却难于分析与设计的表达式。除此以外,对于线性系统,传递函数矩阵提供了定常系统的黑箱式输入输出模型。在时域中,利用自回归滑动平均模型(ARMA),通过各种参数估计方法,也可给出系统的输入输出描述。但对于非线性系统,基于非线性自回归滑动平均模型(NARMA),却难于找到一个恰当的参数估计方法,传统的非线性控制系统辨识方法,在理论研究和实际应用中都存在极大的困难。

相比之下,神经网络在这方面显示出明显的优越性。由于神经网络具有通过学习逼近任意非线性映射的能力,将神经网络应用于非线性系统的建模与辨识,可不受非线性模型类的限制,而且便于给出工程上易于实现的学习算法。

在控制问题的研究中,运用最为普遍的神经网络是多层前馈神经网络(MPLs),这主要是因为这种网络具有逼近任意非线性映射的能力。

逼近理论是数学中的经典问题。如 Volterra 级数等多项式逼近已被广泛应用于非线性控制系统(Schetzen,1980)。一般说来,最佳逼近问题需要首先解决逼近的存在性,即 f 需满足什么样的条件,才能用什么样的 \tilde{f} 以任意精度逼近。此外逼近的唯一性,以及最佳逼近元的构造等也是急需解决的重要问题。

在讨论神经网络逼近理论之前,我们首先简单地介绍泛函中的若干基本概念,并且不加证明地引述两个重要的逼近定理。

1. 预备知识

定义 3.4 (线性赋范空间) 假定 X 为一线性空间, $\|\cdot\|$ 为定义在 X 上的实函数。如果对 $\forall x, y \in X, \alpha \in R$, 有

- (1) $\|x\| \geq 0$,
- (2) $\|x\| = 0$ 当且仅当 $x=0$,
- (3) $\|\alpha x\| = |\alpha| \|x\|$,
- (4) $\|x+y\| \leq \|x\| + \|y\|$,

则称 $\|x\|$ 为 X 上的度量或范数, 并将定义了向量范数的线性空间 $(X, \|x\|)$ 称为线性赋范空间。

若线性赋范空间中任一收敛向量序列的极限均属于此线性赋范空间, 则称此线性赋范空间为完备的线性赋范空间, 或称 Banach 空间。可以证明, 欧氏空间 R^n 、 C^n 和 l^p 空间 ($p=1, 2, \dots, \infty$) 都是 Banach 空间。

在线性赋范空间里, $\|x-y\|$ 表示 x, y 的距离。对于 n 维欧氏空间 R^n , 若记 $x=(x_1, x_2, \dots, x_n)^T$ 为 R^n 中的任一元素, 则称

$$\|x\|_1 = \sum_{i=1}^n |x_i| \quad \|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2} \quad \|x\|_\infty = \max_i |x_i|$$

分别为 R^n 的 1 范数, 欧氏范数(2 范数)和 Chebychev 范数(∞ 范数)。

定义 3.5 (列紧性) 线性赋范空间 X 的子集 D 被称为列紧的, 如果 D 中的任意点列在 X 中都有一个收敛子列。若这个子列还收敛到 D 中的点, 则称 D 为自列紧的。显然, X 的自列紧子集必是有界闭集。

容易证明, 线性赋范空间 X 中的每个有限维有界闭子集 D 都是自列紧的。换句话说, 此时 D 中的每个无穷点列都有一个收敛于 D 中某一点的子序列。例如, 设 D 为 n 维欧氏空间 R^n 中的一个有界闭集, 即

$$D = \{(x_1, x_2, \dots, x_n) \mid -\infty < a_i \leq x_i < b_i < \infty, i=1, 2, \dots, n\}$$

则 D 为 X 的自列紧子空间。为叙述简洁起见, 若不作特别申明下面的记号 D 均指此定义。

定义 3.6 (稠密子集) 线性赋范空间 X 的子集 E 被称为在 X 中的稠密子集, 如果对 $\forall x \in X, \forall \epsilon > 0, \exists y \in E$, 使得 $\|x-y\| < \epsilon$ 。或者说, 对 $\forall x \in X, \exists \{x_n\} \subset E$, 使得 $x_n \rightarrow x$ 。

定义 3.7 (函数空间) 函数空间 $C(D)$ 被定义为由 R^n 的列紧子集 D 上所有实值连续函数组成的集合。容易看出, 函数空间 $C(D)$ 不仅是线性赋范空间, 而且在 Chebychev 范数意义下还是 Banach 空间。进一步地, 我们可以证明, 若 $\varphi_1, \varphi_2, \dots, \varphi_n$ 是线性赋范空间 $C(D)$ 的线性无关基函数, 则由此扩张成的线性子空间 $B = \text{span}\{\varphi_1, \varphi_2, \dots, \varphi_n\}$ 也必是自列紧空间, 这里的基函数 φ_i 可以是一些简单和易于计算的函数, 例如多项式、三角函数、有理函数和分段多项式等。

对于动态系统, 函数空间 $C[a, b]$ 被定义为有界闭区间 $[a, b]$ 上所有实值连续函数 $f(x(t))$ 组成的集合, 相应的 L_1 范数、 L_2 范数和 ∞ 范数分别被定义为

$$\|f\|_1 = \int_a^b |f(x)| dx \quad \|f\|_2 = \left(\int_a^b |f(x)|^2 dx \right)^{1/2} \quad \|f\|_\infty = \max_{a \leq t \leq b} |f(x(t))|$$

定义 3.7 (最佳逼近) 给定线性赋范空间 $X=(X, \|\cdot\|)$, B 为 X 的一个列紧子集, 对于 $\forall f \in X$, 存在 $\varphi \in B \subset X$, 使得

$$\|f - \varphi^*\| = \inf_{\varphi \in B} \|f - \varphi\|$$

则称 φ^* 为 B 中对 f 的最佳逼近元。

进一步地,取线性赋范空间 $X=C[a, b]$,线性子空间 $B=\text{span}\{\varphi_1, \varphi_2, \dots, \varphi_n\}$,范数 $\|f\|_\infty = \max_{a \leq t \leq b} |f(x(t))|$,则函数空间 $C[a, b]$ 关于 $\|\cdot\|_\infty$ 的最佳逼近问题,称为 Chebyshev 意义下的最佳逼近或称最佳一致逼近。类似地,若范数取为 L_2 范数,则称为最小二乘意义下的最佳逼近。

为了讨论最佳逼近的存在性,下面不加证明地给出如下定理。

定理 3.3 (Weierstrass) (Burkill, 1970; Yosida, 1971; Rudin, 1976) 假定 X 是线性赋范空间, $D \subset X$ 为 n 维列紧子空间, $C(D)$ 为定义在 D 上的连续函数空间,具有度量 $\|\cdot\|$, 设 $B=\text{span}\{\varphi_1, \varphi_2, \dots, \varphi_n\}$ 为多项式线性子空间,则对于 $\forall f \in C(D)$, 必存在多项式 $p^* \in B$, 使对 $\forall x \in D$, 一致地有

$$\|f - p^*\| < \epsilon$$

即 p^* 为 f 的最佳逼近多项式,这里 $\epsilon > 0$ 为任意给定的常数。这实际指出,由多项式全体组成的集合 B 在函数空间 $C(D)$ 中稠密。

现在我们考虑函数空间 $C[a, b]$ 在需要满足插值条件时的最佳逼近问题,这实际是一个 $f(x(t))$ 基于间隔 $[a, b]$ 中插值点 t_1, t_2, \dots, t_N 的内插问题。由上述定理,容易得到如下推论:

推论 3.1 若有界函数 $f \in C[a, b]$, $E = \{t_1, t_2, \dots, t_N\}$ 为 $[a, b]$ 中的点列,则存在最小二乘逼近多项式 $p^*(t)$, 使 $\sum_{i=1}^N |f(x(t_i)) - p^*(t_i)|^2$ 在所有多项式中极小。进一步地,存在最佳一致逼近多项式 $p^*(t)$, 使 $\max_{0 \leq i \leq N} |f(x(t_i)) - p^*(t_i)|$ 在所有多项式中极小。

值得指出的是,根据范数选择的不同,也可有其它不同类型的多项式逼近。它们在本质上都可解释为在适当定义范数的线性空间中最小范数问题,或非约束最优化问题。

Weierstrass 定理保证了多项式最佳逼近的存在性,因此也称为多项式最佳逼近存在定理。为了适合于神经网络分析之用,我们可将其一般化为如下定理。

定理 3.4 (Stone-Weierstrass 定理) (Burkill, 1970; Yosida, 1971; Rudin, 1976) 假定 X 是线性赋范空间, $D \subset X$ 为 n 维列紧子空间, $C(D)$ 为定义在 D 上的 Banach 空间,具有度量 $\|\cdot\|$ 。若 B 为 $C(D)$ 的子代数,即满足

- (1) B 含有恒一函数 $g(x)=1$;
- (2) B 为可分的,即对 D 中任意两个点 $x_1 \neq x_2$, 存在 $g(x) \in B$, 使得 $g(x_1) \neq g(x_2)$;
- (3) B 为代数闭包,即对 B 中任意两个函数 g, h , 均有 gh 及 $\alpha g + \beta h$ 也在 B 中, 这里 α, β 为实常数,从而 B 在 Banach 空间 $C(D)$ 中是稠密的。

对于 $\forall f \in C(D)$, 在子代数 B 中必存在一个函数 $g(x)$, 使对 $\forall x \in D$, 一致地有

$$\|f(x) - g(x)\| < \epsilon$$

这里 $\epsilon > 0$ 为任意给定的常数。

上述定理可以进一步推广到非线性函数 $f(x)$ 为间断可测实值函数的情形,即对列紧集 D 上几乎处处有界的函数 f , 我们总可以找到一个连续函数序列 g_r , 使其几乎处处收

收敛于 f

$$\lim_{r \rightarrow \infty} \|f - g_r\| = 0$$

由此可知,由基本模块或计算模块组成的无限大的神经网络,总可以一致地逼近 $C(D)$ 中的任意函数,而一个有限大的网络,则只能精确地逼近 D 中某个子集上的函数。

2. 多层前馈神经网络的逼近能力

从逼近理论的角度来看,多层前馈神经网络(MLPs)实际是一个带学习参数的非线性映射 $g(w, x)$ 之集合 B ,以及相应对权参数向量 w 之学习算法所组成的连接主义表达。由前述 Stone-Weierstrass 定理可知,一个连续或间断可测实值函数 $f(x)$,总可以由一个神经网络任意充分地逼近,其条件是当且仅当该逼近网络 B 为 $C(D)$ 的子代数。为此我们必须使 MLPs 满足如下三个条件:

(1) MLPs 应具有产生 $g(w, x) = 1$ 的能力。这在许多情况下都是存在的。例如,令输出阈值单元相应的连接权为 1,而令其它所有连接权为零,或通过一个单位输出激发函数,并令输入为零等,均可实现这一条件;

(2) 可分性条件,这在 MLPs 中相当于要求逼近网络具有泛化能力,即所谓不同输入产生不同输出(若 $x_1 \neq x_2 \in D$, 则 $g(w, x_1) \neq g(w, x_2) \in B$),这显然也可保证;

(3) 代数闭包条件要求 MLPs 能产生函数的和与积。相加是显然的,因为这可以通过增加一层输出神经元,将两个网络 $g \in B, h \in B$ 之输出简单相加即可,显然增加了一层的 MLPs 也属于 B ,即 $\alpha g + \beta h \in B$ 。为得到两个函数的积,可以对输出神经元激发函数引入一个变换,例如可通过指数函数、对数函数等将 g, h 表示成一个函数和的形式,这在 MLPs 中也是经常使用的。这一条件实际上相当于要求逼近网络 B 在 Banach 空间 $C(D)$ 中是稠密的。换句话说,满足上述三个条件的神经网络,必可以任意精度逼近任意非线性连续或分断连续函数。

为了具体说明起见,现在对多层前馈神经网络,引入所谓全局逼近神经网络与局部逼近神经网络的概念。

考虑一非线性输入输出过程,其输入为 $x \in R^n$,输出为 $y \in R$,组成相应的输入输出训练集 $\{x, y\}$ 。神经网络 $g(x, w)$ 通过学习自适应地完成对映射 $f: x \rightarrow y$ 的逼近。由于该映射可被认为是一个超曲面 $\Gamma \subset R^n \times R$, $\{x, y\}$ 为 Γ 上的点,因此网络的逼近实际上是对离散空间点 $\{x, y\}$ 的基于基函数的最佳拟合过程。所谓全局逼近神经网络就是在整个输入空间上的逼近,例如 BP 网络、GMDH 等。而局部逼近神经网络则是在输入空间中某条状态轨迹附近的逼近,例如 CMAC、B 样条、RBF 和 FLN 网络等。在结构上,两者的区别在于:后者从输入空间到隐层为固定非线性,从隐层到输出为线性自适应,而前者两层均为非线性自适应。这两类前馈神经网络也可基于定义在权空间 $w \in R^p$ 上的误差超曲面 $E \subset R^p \times R$ 进行相应的解释。由于局部逼近网络关于连接权 w_i 为线性的,其误差超曲面将只有唯一的全局极小点,因此学习速度较快。

有关前馈神经网络逼近理论的研究,已给出了若干结果(Cybenko, 1988; Funahashi, 1989; Hornik, 1989; Carrol, 1989)。已经证明,全局逼近网络集合 B_1 与局部逼近网络集合 B_2 , 是否在 Banach 空间 $C(D)$ 中稠密且满足 Stone-Weierstrass 定理的其他条件,将决定于相应的激发函数类 $\phi(\cdot)$ 和 $\Psi(\cdot)$ 的选择。一般说来,在两种情况下,选择均值非零且

$L^p(D)$ 范数 ($\|f\|_p = \left\{ \int_D |f|^p \right\}^{1/p}, 1 \leq p < \infty$) 有限的函数,如指数衰减函数、sigmoid 函数和 Gaussian 基函数等,均可使集合 B_1, B_2 为 $C(D)$ 的子代数 (Stinchcombe and White, 1989)。然而这并不意味着集合 B_1, B_2 也是最佳逼近。为了提供最佳逼近,我们需要证明该逼近集也是存在集,而这通常又只需证明它们是列紧集。进一步研究表明,由于局部逼近网络类 B_2 是通过基函数 $\Psi(\cdot)$ 的有限线性组合构成的逼近函数,它们显然是闭集和有界集,从而也就是列紧集和存在集。另外,如果赋范线性空间 $C(D)$ 是严格凸的,则由逼近理论可知,此最佳逼近同时也是唯一的。不幸的是,全局逼近网络类 B_1 关于连接权 w_i 为非线性的,因此 B_1 既非存在集,也非唯一逼近,尽管它完全满足 Stone-Weierstrass 定理。

在具体实现时,前馈神经网络的这种逼近能力,实际受到隐层数及隐层单元数的限制。换句话说,对任意非线性函数,到底需要多少隐层数及隐层单元数,才能以任意精度逼近?例如,我们可以直观地说,具有两个隐层的网络,肯定比单个隐层的网络具有更高的逼近精度和泛化能力,而且在总体上可以有更少的计算单元。又如,径向基函数神经网络 (RBF) 将比具有 sigmoid 函数的 BP 网络,具有更佳的逼近能力。但这些都是以增加计算复杂性作为代价的。事实上,逼近精度往往还与拟逼近的函数类型,以及训练集有关。对此复杂问题,目前尚无系统的结果。

利用静态多层前馈神经网络建立系统的输入输出模型,本质上是基于网络的逼近能力,通过学习获知系统差分方程中的未知非线性函数。多层前馈网络的这种逼近能力,除了连接主义的优点外,从理论上说,不会超过其他传统的逼近方法。

3.6.3 利用多层静态网络的系统辨识

神经网络作为系统的一种非传统黑箱式表达工具,其内部结构完全可不为人所知。换句话说,利用神经网络对系统进行建模时,我们最好能做到不先验假定系统的模型。但遗憾的是,利用目前的静态多层前馈神经网络,我们尚不能做到这一点。对于拟辨识的动力学系统,必须预先给出定阶的差分方程(如 NARMA 模型)。

系统辨识中的一个重要问题是系统的可辨识性,即对于一个给定的模型类,是否能在该模型类内足够表达所研究的系统 (Ljung, 1983; 1987)。神经网络缺乏这样的具体理论结果。以后我们将假定所选择的神经网络,能够足够地表达相应的系统。

1. 正向模型

所谓正向模型是指利用多层前馈神经网络,通过训练或学习,使其能够表达系统正向动力学特性的模型。图 3.31 给出了获得系统正向模型的网络结构示意图。其中神经网络与待辨识系统并联,两者的输出误差,即预测误差 $e(t)$ 被用作网络的训练信号。显然,这是一个典型的有人监督学习问题,实际系统作为教师,向神经网络提供学习算法所需的期望输出。对于全局逼近的前馈网络结构,可根

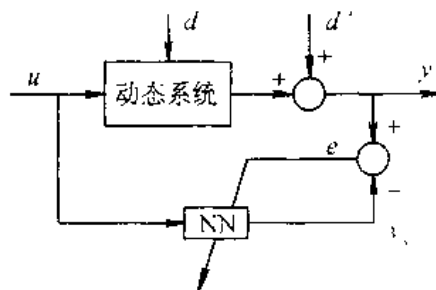


图 3.31 正向模型

据拟辨识系统的不同而选择不同的学习算法。如当系统是被控对象或传统控制器时,一般可选择 BP 学习算法及其各种变形,这时代替被控对象的神经网络,可用来提供控制误差的反向传播通道,或直接替代传统控制器,如 PID 控制器等。而当系统为性能评价器时,则可选择再励学习算法。不过这里的网络结构并不局限于上述选择,我们也可选择局部逼近的神经网络,如小脑模型关节控制器(CMAC)等。

由于在控制系统中,拟辨识的对象通常是动态系统,因此这里就存在一个如何进行动态建模的问题。一个办法是对网络本身引入动态环节,如下面将要介绍的动态递归网络,或者在神经元中引入动态特性。另一个办法,也就是目前通常采用的方法,即首先假定拟辨识对象为线性或非线性离散时间系统,或者人为地离散化为这样的系统,利用 NARMA 模型

$$y(t+1) = f[y(t), \dots, y(t-n+1); u(t), \dots, u(t-m+1)]$$

以便在将 $u(t), \dots, u(t-m+1), y(t), \dots, y(t-n+1)$ 作为网络的增广输入, $y(t+1)$ 作为输出时,利用静态前馈网络学习上述差分方程中的未知非线性函数 $f(\cdot)$ 。显然,这时无法表达对象的干扰部分,除非对干扰也建立相应的差分方程模型类。

2. 逆模型

建立动态系统的逆模型,在神经网络控制中起着关键的作用,并且得到了最广泛的应用。这将在本节中进行详细的介绍。

下面首先讨论神经网络逆建模的输入输出结构,然后介绍两类具体的逆建模方法。

假定上式非线性函数 f 可逆,容易推出

$$u(t) = f^{-1}[y(t), \dots, y(t-n+1), y(t+1); u(t-1), \dots, u(t-m+1)]$$

注意上式中出现了 $t+1$ 时刻的输出值 $y(t+1)$ 。由于在 t 时刻不可能知道 $y(t+1)$,因此我们可用 $t+1$ 时刻的期望输出 $y_d(t+1)$ 来代替 $y(t+1)$ 。对于期望输出而言,其任意时刻的值总可以预先求出。此时,上式成为

$$u(t) = f^{-1}[y(t), \dots, y(t-n+1), y_d(t+1); u(t-1), \dots, u(t-m+1)]$$

同样地, $u(t-1), \dots, u(t-m+1), y(t), \dots, y(t-n+1), y_d(t+1)$ 可作为网络的增广输入, $u(t)$ 可作其输出。这样,利用静态前馈神经网络进行逆建模,也就成了学习逼近上述差分方程中的未知非线性函数 $f^{-1}(\cdot)$ 。

(1) 直接逆建模

直接逆建模也称广义逆学习(Generalized Inverse Learning),如图 3.32 所示。从原理

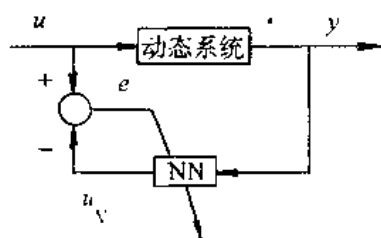


图 3.32 直接逆建模

上说,这是一种最简单的方法。由图中可以看出,拟辨识系统的输出作为网络的输入,网络输出与系统输入比较,相应的输入误差用来进行训练,因而网络将通过学习建立系统的逆模型。不过所辨识的非线性系统有可能是不可逆的,这时利用上述方法,就将得到一个不正确的逆模型。因此,在建立系统的逆模型时,可逆性必须首先假定。

为了获得良好的逆动力学特性,网络学习时所需的样本集,一般应妥为选择,使其比未知系统的实际运行范围更大。但实际工作时的输入信号很难先验给定,因为控制目标是

使系统的输出具有期望的运动,对于未知被控系统,期望输入不可能给出。另一方面,在系统辨识中为保证参数估计算法一致收敛,一个持续激励的输入信号必须提供。尽管对传统自适应控制,已经提出了许多确保持续激励的条件,但对神经网络,这一问题仍待进一步研究。由于实际工作范围内的系统输入 $u(t)$ 不可能预先定义,而相应的持续激励信号又难于设计,这就使该法在应用时,有可能给出一个不可靠的逆模型,为此我们可以采用以下建模方法。

(2) 正-逆建模

正-逆建模也称狭义逆学习(Specialized Inverse Learning)。如图 3.33 所示,这时待辨识的网络 NN 位于系统前面,并与之串联。网络的输入为系统的期望输出 $y_d(t)$,训练误差或者为期望输出与系统实际输出 $y(t)$ 之差,或者为与已建模神经网络正向模型之输出 $y_N(t)$ 之差,即

$$e(t) = y_d(t) - y(t)$$

或

$$e(t) = y_d(t) - y_N(t)$$

其中神经网络正向模型可用前面讨论的方法给出。

该法的特点是:通过使用系统已知的正向动力学模型,或增加使用已建模的神经网络正向模型,以避免再次采用系统输入作为训练误差,使待辨识神经网络仍然沿期望轨迹(输出)附近进行学习。这就从根本上克服了使用系统输入作为训练误差所带来的问题。此外,对于系统不可逆的情况,利用此法也可通过学习得到一个具有期望性能的特殊逆模型(Jordan and Rumelhart, 1991)。

这类建模方法有三种不同的实现途径。

方法一:直接利用系统的实际输出与期望输出之差作为网络逆模型的训练误差。但存在的主要问题是,这时必须知道拟辨识系统的正向动力学模型,以便借之反传误差,这显然与系统解析模型未知矛盾。既然系统的解析模型已知,我们似可由此直接推得系统的逆模型,再去辨识系统的逆模型已无必要。不过当系统的精确模型无法确知,推导其逆模型又显得过于繁琐时,利用神经网络进行辨识,仍不失为一种较好的选择。事实上, Jordan 与 Rumelhart(1991)已经证明,即使系统的解析模型不太精确,利用此法也可望得到一个精确的逆模型。

方法二:但已知系统的正向模型毕竟有悖于我们这里讨论的辨识问题,因此可考虑将此系统代之以相应的已建模神经网络正向模型,即用神经网络正向模型之输出 $y_N(t)$ 代替系统的实际输出 $y(t)$,从而由期望输出 $y_d(t)$ 与 $y_N(t)$ 形成训练误差。这里的神经网络正向模型可由前面介绍的方法预先建立,显然可由它提供误差的反向传播通道。相比之下,此法适宜于有噪声的系统,在不可能利用实际系统已知模型的情形下,该法显示出其优越性。缺点是 $y_N(t)$ 不可能完全等于实际输出 $y(t)$,神经网络正向模型的建模误差,必然影响待辨识逆模型的精度。

方法三:如图 3.33 所示,我们可设想仍然利用系统的实际输出构成训练误差,但反向

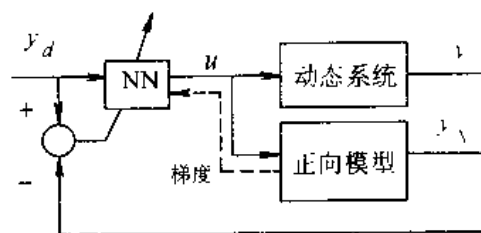


图 3.33 正-逆建模

传播通道则由神经网络正向模型提供。由于正向模型仅起误差梯度信息的反向传播作用,即使有一点误差,也不是至关重要的,它一般只影响逆模型神经网络的收敛速度。显然,这种方法综合了前两种方法的优点,同时还克服了它们的缺点。

3.6.4 利用动态网络的系统辨识

如前所述,利用静态多层前馈网络对动态系统进行辨识,实际是将动态时间建模问题变为一个静态空间建模问题,这就必然出现诸多问题。如需要先验假定系统的 NARMA 模型类,需要对结构模型进行定阶,特别是随着系统阶次的增加或阶次未知时,迅速膨胀的网络结构,将使学习收敛速度更加缓慢。此外较多的输入节点也将使相应的辨识系统对外部噪声特别敏感。

相比之下,动态递归网络提供了一种极具潜力的选择,代表了神经网络建模、辨识与控制的发展方向。

在本小节,我们将介绍一种修改的 Elman 动态递归网络,然后给出 Elman 网络在线性动态系统辨识中的应用。

1. 基本 Elman 动态递归网络

与前馈神经网络分为全局与局部逼近网络类似,动态递归神经网络也可分为完全递归与部分递归网络。完全递归网络具有任意的前馈与反馈连接,且所有连接权都可进行修正。而在部分递归网络中,主要的网络结构是前馈,其连接权可以修正;反馈连接由一组所谓“结构”(Context)单元构成,其连接权不可以修正。这里的结构单元记忆隐层过去的状态,并且在下一时刻连同网络输入,一起作为隐层单元的输入。这一性质使部分递归网络具有动态记忆的能力。

(1) 网络结构

在动态递归网络中,Elman 网络(Elman,1990)具有最简单的结构,它可采用标准 BP 算法或动态反向传播算法进行学习。一个基本 Elman 网络的结构示意图如图 3.34 所示。

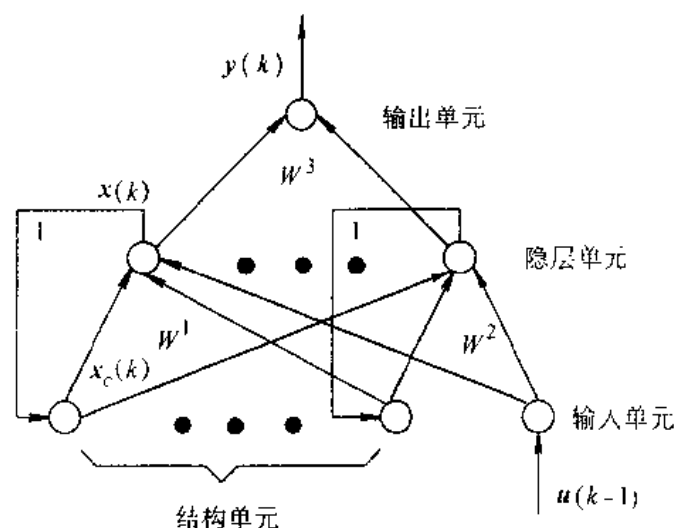


图 3.34 基本 Elman 网络的结构示意图

从图中可以看出,Elman 网络除输入层、隐层及输出层单元外,还有一个独特的结构单元。

与通常的多层前馈网络相同,输入层单元仅起信号传输作用,输出层单元起线性加权和作用,隐层单元可有线性或非线性激发函数。而结构单元则用来记忆隐层单元前一时刻的输出值,可认为是一个一步时延算子。因此这里的前馈连接部分可进行连接权修正,而递归部分则是固定的即不能进行学习修正,从而此 Elman 网络仅是部分递归的。

具体地说,网络在 k 时刻的输入不仅包括目前的输入值 $u(k-1)$,而且还包括隐层单元前一时刻的输出值 $x_c(k)$,即 $x(k-1)$ 。这时,网络仅是一个前馈网络,可由上述输入通过前向传播产生输出,标准的 BP 算法可用来进行连接权修正。在训练结束之后, k 时刻隐层的输出值将通过递归连接部分,反传回结构单元,并保留到下一个训练时刻($k+1$ 时刻)。在训练开始时,隐层的输出值可取为其最大范围的一半,例如当隐层单元取为 Sigmoid 函数时,此初始值可取为 0.5,当隐层单元为双曲正切函数时,则可取为 0。

下面对 Elman 网络所表达的数学模型进行分析。

如图 3.34 所示,设网络的外部输入为 $u(k-1) \in R^r$,输出为 $y(k) \in R^n$,若记隐层的输出为 $x(k) \in R^n$,则有如下非线性状态空间表达式成立,

$$\begin{aligned}x(k) &= f(W^1 x_c(k) + W^2 u(k-1)) \\x_c(k) &= x(k-1) \\y(k) &= g(W^3 x(k))\end{aligned}$$

其中 W^1, W^2, W^3 分别为结构单元到隐层、输入层到隐层,以及隐层到输出层的连接权矩阵, $f(\cdot)$ 和 $g(\cdot)$ 分别为输出单元和隐层单元之激发函数所组成的非线性向量函数。特别地,当隐层单元和输出单元采用线性函数且令隐层及输出层的阈值为 0 时,则可得到如下线性状态空间表达式

$$\begin{aligned}x(k) &= W^1 x(k-1) + W^2 u(k-1) \\y(k) &= W^3 x(k)\end{aligned}$$

这里隐层单元的个数就是状态变量的个数,也即是系统的阶次。

显然,当网络用于单输入单输出系统时,我们只需一个输入单元和一个输出单元。即使考虑到这时的 n 个结构单元,隐层的输入也仅有 $n+1$ 个。这与将上述状态方程化为差分方程,并利于静态网络进行辨识时,需要 $2n$ 个输入相比,无疑有较大的减少,特别是当 n 较大时。另外,由于 Elman 网络的动态特性仅由内部的连接提供,因此它无需直接使用状态作为输入或训练信号,这也是 Elman 网络相对于静态前馈网络的优越之处。

Pham 等(1990)发现,上述网络在采用标准 BP 学习算法时,仅能辨识一阶线性动态系统。原因是标准 BP 算法只有一阶梯度,致使基本 Elman 网络对结构单元连接权的学习稳定性较差,从而当系统阶次增加或隐层单元增加时,将直接导致相应的学习率极小(为保证学习收敛),以致不能提供可接受的逼近精度。对此我们可利用下面将要介绍的动态反向传播学习算法,或对基本 Elman 网络进行扩展。

(2) 学习算法

由上面的式子可知,

$$x_c(k) = x(k-1) = f(W_{k-1}^1 x_c(k-1) + W_{k-1}^2 u(k-2))$$

又由于 $x_c(k-1) = x(k-2)$,上式可继续展开。这说明 $x_c(k)$ 依赖于过去不同时刻的连接权 $W_{k-1}^1, W_{k-2}^2, \dots$,或者说 $x_c(k)$ 是一个动态递推过程。因此可将相应推得的反向传播算

法称为动态反向传播学习算法(Kuan, 1989)。

考虑如下总体误差目标函数

$$E = \sum_{p=1}^N E_p$$

其中

$$E_p = \frac{1}{2} (y_d(k) - y(k))^T (y_d(k) - y(k))$$

对隐层到输出层的连接权 W^3

$$\frac{\partial E_p}{\partial w_{ij}^3} = - (y_{d,i}(k) - y_i(k)) \frac{\partial y_i(k)}{\partial w_{ij}^3} = - (y_{d,i}(k) - y_i(k)) g'_i(\cdot) x_j(k)$$

令 $\delta_i^0 = (y_{d,i}(k) - y_i(k)) g'_i(\cdot)$, 则

$$\frac{\partial E_p}{\partial w_{ij}^3} = - \delta_i^0 x_j(k) \quad i = 1, 2, \dots, m; \quad j = 1, 2, \dots, n$$

对输入层到隐层的连接权 W^2

$$\frac{\partial E_p}{\partial w_{jq}^2} = \frac{\partial E_p}{\partial x_j(k)} \frac{\partial x_j(k)}{\partial w_{jq}^2} = \sum_{i=1}^m (-\delta_i^0 w_{ij}^3) f'_i(\cdot) u_q(k-1)$$

同样令 $\delta_j^1 = \sum_{i=1}^m (\delta_i^0 w_{ij}^3) f'_i(\cdot)$, 则有

$$\frac{\partial E_p}{\partial w_{jq}^2} = - \delta_j^1 u_q(k-1) \quad j = 1, 2, \dots, n; \quad q = 1, 2, \dots, r$$

类似地, 对结构单元到隐层的连接权 W^1 , 我们有

$$\frac{\partial E_p}{\partial w_{jl}^1} = - \sum_{i=1}^m (\delta_i^0 w_{ij}^3) \frac{\partial x_j(k)}{\partial w_{jl}^1} \quad j = 1, 2, \dots, n; \quad l = 1, 2, \dots, n$$

注意到上面的式子, $x_c(k)$ 依赖于连接权 w_{il}^1 , 故

$$\begin{aligned} \frac{\partial x_j(k)}{\partial w_{jl}^1} &= \frac{\partial}{\partial w_{jl}^1} (f_j(\sum_{i=1}^n w_{ji}^1 x_{c,i}(k) + \sum_{i=1}^r w_{ji}^2 u_i(k-1))) \\ &= f'_j(\cdot) \{x_{c,l}(k) + \sum_{i=1}^n w_{ji}^1 \frac{\partial x_{c,i}(k)}{\partial w_{jl}^1}\} \\ &= f'_j(\cdot) \{x_l(k-1) + \sum_{i=1}^n w_{ji}^1 \frac{\partial x_i(k-1)}{\partial w_{jl}^1}\} \end{aligned}$$

上式实际构成了梯度 $\partial x_j(k) / \partial w_{jl}^1$ 的动态递推关系, 这与沿时间反向传播的学习算法类似 (Werbos, 1988)。由于

$$\Delta W_{ij} = - \eta \frac{\partial E_p}{\partial w_{ij}}$$

故基本 Elman 网络的动态反向传播学习算法可归纳如下:

$$\Delta w_{ij}^3 = \eta \delta_i^0 x_j(k) \quad i = 1, 2, \dots, m; \quad j = 1, 2, \dots, n$$

$$\Delta w_{jq}^2 = \eta \delta_j^1 u_q(k-1) \quad j = 1, 2, \dots, n; \quad q = 1, 2, \dots, r$$

$$\Delta w_{jl}^1 = \eta \sum_{i=1}^m (\delta_i^0 w_{ij}^3) \frac{\partial x_j(k)}{\partial w_{jl}^1} \quad j = 1, 2, \dots, n; \quad l = 1, 2, \dots, n$$

$$\frac{\partial x_j(k)}{\partial w_{ji}^1} = f'_j(\cdot) \{x_i(k-1) + \sum_{i=1}^n w_{ji}^1 \frac{\partial x_i(k-1)}{\partial w_{ji}^1}\}$$

这里

$$\delta_i^0 = (y_{d,i}(k) - y_i(k))g'_{i'}(\cdot)$$

$$\delta_j^s = \sum_{i=1}^m (\delta_i^0 w_{ij}^s) f'_j(\cdot)$$

当 $x_i(k-1)$ 与连接权 w_{ji}^1 之间的依赖关系可以忽略时, 由于 $\partial x_j(k)/\partial w_{ji}^1 = f'_j(\cdot) x_{c,i}(k) = f'_j(\cdot) x_i(k-1)$, 上述算法就退化为如下标准 BP 学习算法。

$$\Delta w_{ij}^3 = \eta \delta_i^0 x_j(k) \quad i = 1, 2, \dots, m; j = 1, 2, \dots, n$$

$$\Delta w_{jq}^2 = \eta \delta_j^s u_q(k-1) \quad j = 1, 2, \dots, n; q = 1, 2, \dots, r$$

$$\Delta w_{jl}^1 = \eta \delta_j^s x_{c,l}(k) \quad j = 1, 2, \dots, n; l = 1, 2, \dots, n$$

2. 修改的 Elman 网络

(1) 网络结构

图 3.35 给出了一种修改 Elman 网络的结构示意图。这是解决高阶系统辨识的更好方案。

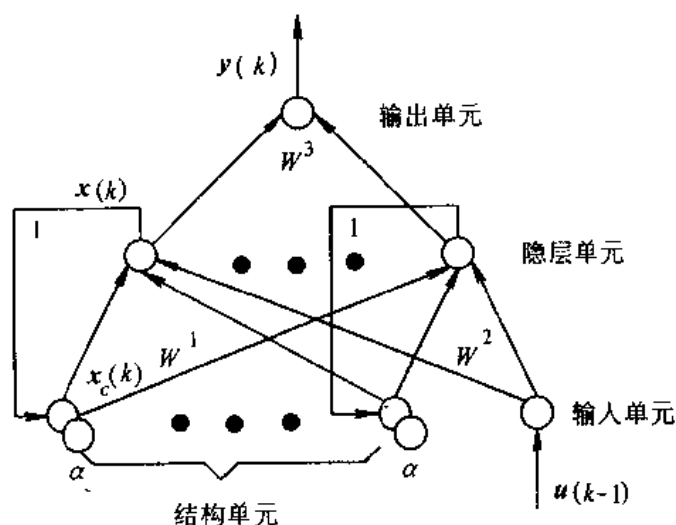


图 3.35 一种修改 Elman 网络的结构示意图

比较图 3.34 及图 3.35 可以看出, 两者的不同之处在于: 修改的 Elman 网络在结构单元中, 有一个固定增益 α 的自反馈连接。因此, 结构单元在 k 时刻的输出, 将等于隐层在 $k-1$ 时刻的输出加上结构单元在 $k-1$ 时刻输出值的 α 倍, 即

$$x_{c,l}(k) = \alpha x_{c,l}(k-1) + x_l(k-1) \quad l = 1, 2, \dots, n$$

其中 $x_{c,l}(k)$ 和 $x_l(k)$ 分别表示第 l 个结构单元和第 l 个隐层单元的输出, α 为自连接反馈增益。显然当相同的固定增益 α 为零时, 修改的 Elman 网络就退化为基本的 Elman 网络。

与前面的式子类似, 由修改 Elman 网络描述的非线性状态空间表达式为

$$x(k) = f(W^1 x_c(k) + W^2 u(k-1))$$

$$\begin{aligned}x_c(k) &= x(k-1) + \alpha x_c(k-1) \\y(k) &= g(W^3 x(k))\end{aligned}$$

(2) 学习算法

由于对结构单元增加了自反馈连接,修改的 Elman 网络可利用标准 BP 学习算法辨识高阶动态系统。与基本 Elman 网络标准 BP 学习算法的推导完全相同,容易得到修改 Elman 网络的标准 BP 学习算法为

$$\begin{aligned}\Delta w_{ij}^3 &= \eta \delta_i^3 x_j(k) \quad i=1,2,\cdots,m; j=1,2,\cdots,n \\ \Delta w_{jq}^2 &= \eta \delta_j^2 u_q(k-1) \quad j=1,2,\cdots,n; q=1,2,\cdots,r \\ \Delta w_{jl}^1 &= \eta \sum_{i=1}^m (\delta_i^3 W_{ij}^3) \frac{\partial x_j(k)}{\partial w_{jl}^1} \quad j=1,2,\cdots,n; l=1,2,\cdots,n\end{aligned}$$

如前所述,由于在推导 Elman 网络的标准 BP 算法时,不考虑 $x_{c,i}(k)$ 与 w_{jl}^1 之间的依赖关系,故

$$\partial x_j(k) / \partial w_{jl}^1 = f'_j(\cdot) x_{c,i}(k)$$

代入前面的式子,得

$$f'_j(\cdot) x_{c,i}(k) = f'_j(\cdot) x_i(k-1) + \alpha f'_j(\cdot) x_{c,i}(k-1)$$

因而有

$$\frac{\partial x_j(k)}{\partial w_{jl}^1} = f'_j(\cdot) x_i(k-1) + \alpha \frac{\partial x_j(k-1)}{\partial w_{jl}^1}$$

将上式与前面的式子比较,两者非常相近。这就回答了为什么修改的 Elman 网络只利用标准 BP 学习算法,就能达到基本 Elman 网络利用动态反传算法所达到的效果,即能有效地辨识高于一阶的动态系统。

3. 基于修改 Elman 网络的动态系统辨识

考虑如下三阶线性动态系统

$$G(s) = \frac{K}{(s+1)(s+2)(s+3)}$$

假定该系统未知,为了辨识其正向动力学模型,我们不妨采用图 3.31 的系统辨识结构。仿真中,400 个数据的样本集可由均匀分布产生,即网络的输入由具有均匀分布的随机数产生,训练准则为输出的均方根(r. m. s)误差。若取自反馈增益 $\alpha=0.65$,SBP 算法的学习率 $\eta=0.01$,动量项 $\alpha'=0.1$,采样周期 $T=0.1\text{sec}$,网络结构采用 $1 \times 4 \times 1$ 。则在经过 200 000 次学习迭代后,可使 r. m. s 误差 $E=0.025685$ 。

3.7 神经网络控制

3.7.1 概述

上面介绍了动态系统正向与逆模型的建模方法,这些方法在相当程度上构成了神经网络控制结构的设计基础。从本节开始,我们将转而研究神经网络控制方法。

对神经网络控制的研究,我们最终可从人脑控制行为的生理学研究中得到启发,因为这是人工神经网络及其控制方法所追求的终结目标。经过过去 20 多年的研究,已揭示出

人脑的结构和功能特征,实际表现为一个控制器(Albus, 1975; Ito, 1984; Kawato 等, 1987)。事实上,神经中枢系统对手臂、双足及体姿的控制,其表现是如此完美。不管需完成的任务多么复杂,如高难度的体操动作,人脑并不需要操作对象与环境的定量数学模型,也无需求解任何微分方程。生物神经网络控制系统这种对不确定、复杂、不精确和近似问题的控制能力,是大多数传统控制方法所难以达到的。

尽管目前的人工神经网络控制方法,距上述目标仍相距遥远。然而,相对于一般的控制方法,神经网络控制系统所特有的学习能力、潜在的分布并行计算特点,以及对多传感信息的处理性能等,仍使其具有许多潜在的优势。由于目前使用的人工神经网络,不论是网型、学习算法和网络规模等,比真实的生物神经系统,仍极其原始、简单,这就使相应的控制方法出现了许多暂时难以克服的困难。如前所述,单纯使用神经网络的控制方法的研究,目前甚至有停滞不前的趋势。原因很多,除人们一开始对它寄予的期望过高外,主要是因为:(1)近年来,神经网络本身的研究,如网型等未再有根本的突破,专门适合于控制问题的动态神经网络仍待进一步发展;(2)神经网络的泛化能力不足,制约了控制系统的鲁棒性;(3)网络本身的黑箱式内部知识表达方式,使其不能利用初始经验进行学习,易于陷入局部极小值;(4)分布并行计算的潜力还有赖于硬件实现技术的进步等。

尽管如此,目前对神经网络及其控制方法的研究,仍然方兴未艾。前已指出,将模糊逻辑、专家系统和机器学习等符号处理方法融合进神经网络方法,发展具有不同智能粒度的连接主义与符号主义综合集成系统,仍极具潜力,可以说代表了神经网络及其控制方法的主要发展趋势。

本书将主要致力于目前已有的典型方法。在本节,我们将首先讨论基于神经网络的若干典型控制结构方案,或对其归纳分类,然后针对实际控制问题,具体介绍三种有一定代表性的全局逼近、局部逼近和模糊神经网络控制系统。书中的许多内容均取自于作者的研究结果。

3.7.2 神经网络控制结构

迄今为止,有关神经网络控制方法与结构的文献被大量提出和使用。由于分类方法的不同,结果自然有所不同。限于本书的目的,我们不想也不可能罗列出所有方法。但我们认为,典型的控制结构至少包括如下方案:神经网络监督控制(或称神经网络学习控制);神经网络自适应控制(自校正、模型参考控制,含直接与间接自适应控制);神经网络内模控制;神经网络预测控制;神经网络自适应评判控制(或称神经网络再励控制)等。神经网络控制结构方案的研究,构成了神经网络控制方法的设计基础。

1. 神经网络监督控制

一般地说,当被控对象的解析模型未知或部分未知时,利用传统的控制理论设计控制器,已被证明是极其困难的。但这并不等于该系统是不可控的。在许多实际控制问题中,人工控制或PID控制可能是唯一的选择。但在工况条件极其恶劣,或控制任务只是些单调、重复和繁重的简单操作时,我们有必要应用自动控制器代替上述手工操作。

取代人工控制的途径大致有两种。一是将手工操作中的经验总结成普通的规则或模糊规则,然后构造相应的专家控制器或模糊控制器,这已在前面各章中讲述。二是在知识

难于表达的情况下,应用神经网络学习人的控制行为,即对人工控制器建模,然后用此神经网络控制器代替之。

这种通过对人工或传统控制器进行学习,然后用神经网络控制器取代或逐渐取代原控制器的方法,称为神经网络监督控制或 COPY 控制。

图 3.36 给出了这类神经网络控制方法的结构方案示意图。

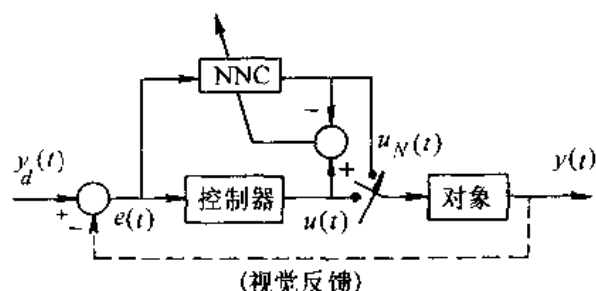


图 3.36 神经网络监督控制(I)

从图中可以看出,神经网络监督控制实际就是建立人工控制器的正向模型。经过训练,神经网络将记忆该控制器的动态特性,并且接受传感信息输入,最后输出与人工控制器相似的控制作用。但此法的缺点是,人工控制器是靠视觉反馈进行控制的,在用神经网络控制器进行控制后,由于缺乏视觉反馈,由此构成的控制系统实际是一个开环系统,这就使其稳定性和鲁棒性均得不到保证。

为此,我们可考虑在传统控制器,如 PID 控制器基础上,再增加一个神经网络控制器,如图 3.37 所示。此时神经网络控制器实际是一个前馈控制器,因此它建立的是被控对象的逆模型。由图中容易看出,神经网络控制器通过向传统控制器的输出进行学习,在线调整自己,目标是使反馈误差 $e(t)$ 或 $u_1(t)$ 趋近于零,从而使自己逐渐在控制作用中占据主导地位,以便最终取消反馈控制器的作用。但与上述结构不同,这里的反馈控制器仍然存在,一旦系统出现干扰等,反馈控制器仍然可以重新起作用。因此,采用这种前馈加反馈的监督控制方法,不仅可确保控制系统的稳定性和鲁棒性,而且可有效地提高系统的精度和自适应能力。

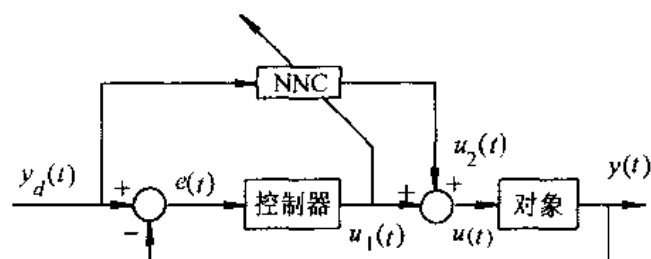


图 3.37 神经网络监督控制(I)

2. 神经网络直接逆控制

顾名思义,神经网络直接逆控制就是将被控对象的神经网络逆模型,直接与被控对象

串联起来,以便使期望输出(即网络输入)与对象实际输出之间的传递函数等于1,从而在将此网络作为前馈控制器后,使被控对象的输出为期望输出。直接逆控制已被应用于机器人控制。例如,Miller 基于 CMAC 网络,利用直接逆控制,使 PUMA 机械手的跟踪精度达到百分之…的数量级。

神经网络直接逆控制在结构上与前述的逆模型辨识有许多相似之处。显然,该法的可用性在相当程度上取决于逆模型的准确程度。由于缺乏反馈,简单连接的直接逆控制将缺乏鲁棒性。为此,我们一般应使其具有在线学习能力,即逆模型的连接权必须能够在线修正。

图 3.38 给出了两种结构方案。在图 3.38(a)中,NN1 和 NN2 具有完全相同的网络结构(逆模型),并且采用相同的学习算法,即 NN1 和 NN2 的连接权都沿 $E = \frac{1}{2} \sum_k e(k)^T e(k)$ 的负梯度方向进行修正。上述评价函数也可采用其他更一般的加权形式,这时的结构方案如图 3.38(b)所示。

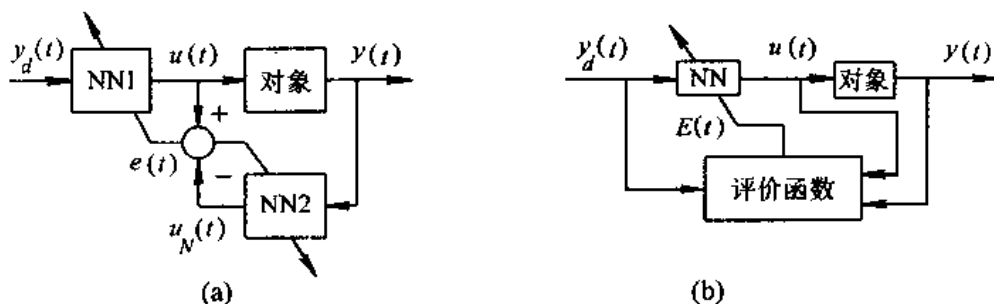


图 3.38 神经网络直接逆控制

3. 神经网络自适应控制

与传统自适应控制相同,神经网络自适应控制也可分为自校正控制(STC)与模型参考控制(MRAC)两种。两者的区别是:自校正控制将根据对系统正向和(或)逆模型辨识的结果,直接调节控制器内部参数,使系统满足给定的性能指标。而在模型参考控制中,闭环控制系统的期望性能由一个稳定的参考模型描述,它被定义为 $\{r(t), y^m(t)\}$ 输入-输出对,控制系统的目的就是要使被控对象的输出 $y(t)$ 一致渐近地趋近于参考模型的输出,即

$$\lim_{t \rightarrow \infty} \|y(t) - y^m(t)\| \leq \epsilon$$

其中, ϵ 是一个给定的小正数。

(1) 神经网络自校正控制

神经网络自校正控制也分为间接与直接控制。它们的根本区别在于,前者使用常规控制器,离线辨识的神经网络估计器需要具有足够高的建模精度;而后者则同时使用神经网络控制器和神经网络估计器,其中估计器可进行在线修正。

• 直接自校正控制

神经网络直接自校正控制,有时也称神经网络直接逆控制,它们本质上是完全一致的。其结构框图如图 3.38 所示。

- 间接自校正控制

神经网络间接自校正自适应控制的结构框图,如图 3.39 所示。

不失一般性,假定被控对象为如下单变量仿射非线性系统

$$y_{k+1} = f(y_k) + g(y_k)u_k$$

若利用神经网络对非线性函数 $f(y_k)$ 和 $g(y_k)$ 进行离线辨识,得到具有足够逼近精度的估计值 $\hat{f}(y_k)$ 和 $\hat{g}(y_k)$,则常规控制律可直接给出为

$$u_k = [y_{d,k+1} - \hat{f}(y_k)] / \hat{g}(y_k)$$

其中 $y_{d,k+1}$ 为 $k+1$ 时刻的期望输出值。

类似地,我们也可以利用神经网络估计输出响应的特性参数,如上升时间 t_r 、超调量 σ 、或二阶系统的自然振荡频率 ω_n 及阻尼系数 ζ 等,然后用常规的极点配置方法调整控制器的参数。

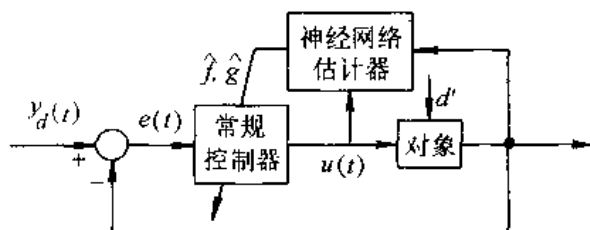


图 3.39 神经网络间接自校正自适应控制

(2) 神经网络模型参考控制

神经网络模型参考控制也有直接与间接控制之分,如图 3.40 与图 3.41 所示。

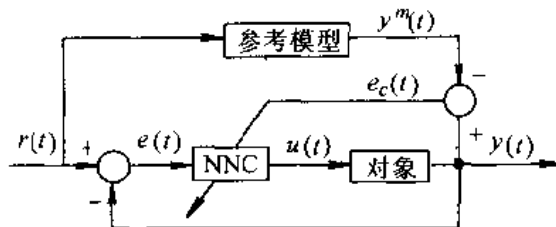


图 3.40 神经网络直接模型参考自适应控制

- 直接模型参考控制

由图 3.40 可知,在神经网络直接模型参考控制中,神经网络控制器的作用是使被控对象与参考模型输出之差 $e_c(t) = y(t) - y^m(t) \rightarrow 0$ 或 $e_c(t)$ 的二次型最小。与正-逆建模中的方法一类似,误差 $e_c(t)$ 的反向传播必须确知被控对象的数学模型,这给 NNC 的学习修正带来了许多问题。为此,我们可采用其中的方法二、方法三,这就构成了下面的间接模型参考控制。

- 间接模型参考控制

如图 3.41 所示,神经网络辨识器 NNI 首先离线辨识被控对象的正向模型,并可由

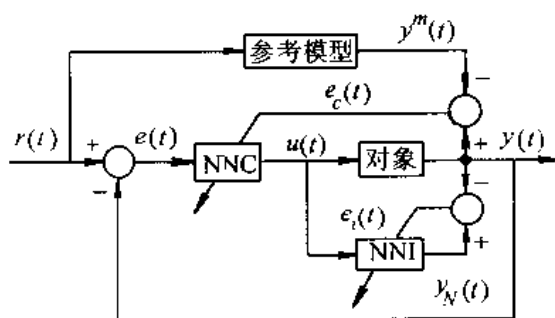


图 3.41 神经网络间接模型参考自适应控制

$e_c(t)$ 进行在线学习修正。显然 NNI 可为 NNC 提供误差 $e_c(t)$ 或其梯度的反向传播通道。由于参考模型输出可视为期望输出,因此在对象部分已知的情况下,若将 NNC 改为常规控制器,此法将与前面介绍的间接自校正控制方法类同。

4. 神经网络内模控制

经典的内模控制将被控系统的正向模型和逆模型直接加入反馈回路,已被证明具有许多好的性质。迄今,内模控制的鲁棒性和稳定性分析已被透彻研究(Morari,1989),并且已被发展为非线性控制的一种重要方法(Economou,1986)。

在内模控制中,系统的正向模型与实际系统并联,两者输出之差被用作反馈信号,此反馈信号又由前向通道的滤波器及控制器进行处理。由内模控制的性质可知,该控制器直接与系统的逆有关,而引入滤波器的目的则是为了获得期望的鲁棒性和跟踪响应。图3.42给出了内模控制的神经网络实现。其中,被控对象的正向模型及控制器(逆模型)均由神经网络实现,滤波器仍然是常规的线性滤波器,显然这种实现是极其直观的(Hunt and Sbarbaro,1991)。

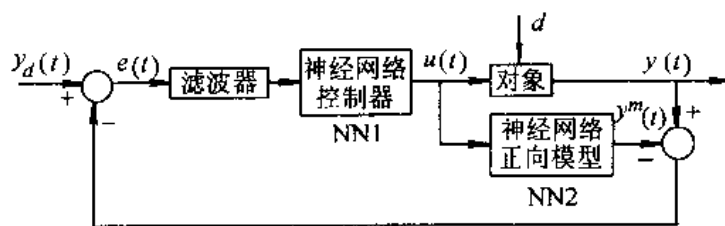


图 3.42 神经网络内模控制

5. 神经网络预测控制

预测控制,又称基于模型的控制,是70年代后期发展起来的一类新型计算机控制算法。该法的特征是预测模型、滚动优化和反馈校正。已经证明,该法对非线性系统具有期望的稳定性能(Keerthi,1986;Mayne,1990)。

神经网络预测控制的结构方案,如图3.43所示。其中神经网络预测器建立了非线性被控对象的预测模型,并可在线学习修正。利用此预测模型,我们就可以由目前的控制输入 $u(t)$,预报出被控系统在将来一段时间范围内的输出值

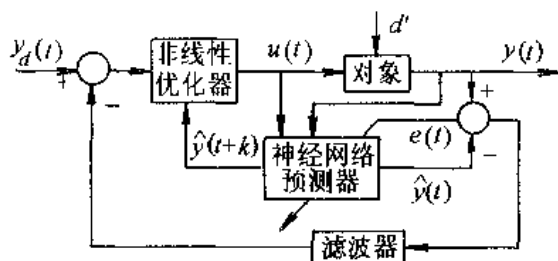


图 3.43 神经网络预测控制

$$y(t+j|t) \quad j = N_1, N_1 + 1, \dots, N_2$$

其中, N_1 、 N_2 分别称为最小与最大输出预报水平,反映了所考虑的跟踪误差和控制增量的时间范围。若 $t+j$ 时刻的预报误差定义为

$$e(t+j) = y_d(t+j) - y(t+j|t)$$

则非线性优化器将使如下二次型性能指标极小,以便得到适宜的控制作用 $u(t)$,即

$$J = \sum_{j=N_1}^{N_2} e^2(t+j) + \sum_{j=1}^{N_2} \lambda_j \Delta u^2(t+j-1)$$

这里, $\Delta u(t+j-1) = u(t+j-1) - u(t+j-2)$,且 λ 为控制加权因子。

神经网络预测控制的算法为:

- (1) 计算未来的期望输出序列 $y_d(t+j)$, $j = N_1, N_1 + 1, \dots, N_2$;
- (2) 利用神经网络预测模型,产生预报输出 $y(t+j|t)$, $j = N_1, N_1 + 1, \dots, N_2$;
- (3) 计算预报误差 $e(t+j) = y_d(t+j) - y(t+j|t)$, $j = N_1, N_1 + 1, \dots, N_2$;
- (4) 极小化性能指标 J ,获得最优控制序列 $u(t+j)$, $j = 0, 1, 2, \dots, N_2$;
- (5) 采用第一控制量 $u(t)$,然后返回到(1)。

顺便指出,由于这里的非线性优化器实际上是一个优化算法,因此我们可设想利用动态反馈网络,来实现这一算法,并进一步构成动态网络预测器。

6. 神经网络自适应评判控制

上述各种控制方法,不管采取何种神经网络控制结构,它们有一点在本质上是共同的:即都要求提供被控对象的期望输入。

我们知道,神经网络之学习方法一般可分成三种类型:监督学习、再励学习与无人监督学习。监督学习虽有最高的学习效率,但它需要教师提供网络的期望输出,对于神经网络控制器,也就是需要提供期望的控制信号。一般说来,控制的目标就是找到被控系统的这一期望输入。在系统模型未知或部分未知的情况下,显然这是难于预先提供的。无人监督学习利用网络的输入数据构造内部教师模型,不再接受其它信息,相当于自组织聚类方法,但学习效率十分低下。再励学习(reinforcement learning)介于两者之间,只需要系统的一个标量评价值,这在缺乏被控系统的精确观测值,只能获得定性的信息反馈时,显然是十分有用的。

作为人与动物行为的一种普遍机制,“再励”这一术语最早源于心理学。它的一般性理论与随机自动机理论有关。早期工作可追溯到 Bush 等(1958)及 Tsetlin(1973)所得结果。

神经网络自适应评判控制,首先由 Barto 等(1983)提出,然后由 Anderson(1989)及 Berenji(1989;1990;1992)等加以发展。特别是 Berenji 的工作,已将神经网络自适应评判控制发展为模糊神经网络自适应评判控制,得到了所谓基于近似推理和再励学习的 AR-IC 及 GARIC 系统。

神经网络自适应评判控制通常由两个网络组成,如图 3.44 所示。其中自适应评判网络在整个控制系统中,相当于一个需要进行再励学习的“教师”。其作用有二:一是通过不断的奖励、惩罚等再励学习,使自己逐渐成为一个“合格”的教师,其再励学习算法的收敛性已得到证明;二是在学习完成后,根据被控系统目前的状态及外部再励反馈信号 $r(t)$,如倒立摆成功与失败的信号(0/1),产生一再励预测信号 $\hat{p}(t)$,并进而给出内部再励信号 $\hat{r}(t)$,以期对目前控制作用的效果作出评价。控制选择网络的作用相当于一个在内部再励信号指导下进行学习的多层前馈神经网络控制器。该网络在进行上述学习后,将根据编码后的系统状态,在允许控制集中选择下一步的控制作用。控制选择网络也可以是一个模糊神经网络控制器。

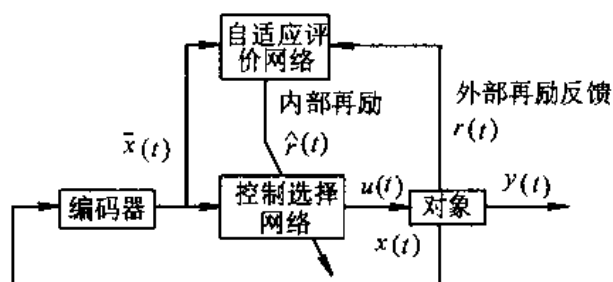


图 3.44 神经网络自适应评判控制

从这里可以看出,神经网络自适应评判控制与人脑的控制与决策过程比较接近,除应随时了解一些定性信息外,它完全不需要被控系统的先验定量模型,特别适合于许多具有高度非线性和严重不确定性的复杂系统的控制。

3.7.3 基于全局逼近神经网络的控制

1. 引言

前面我们已经研究了各种典型的神经网络模型,并且系统地介绍了神经网络控制结构方案。下面我们将结合具体的控制问题,从全局逼近、局部逼近的角度,分别给出三种实际的神经网络控制系统。在有关的讨论中,我们将侧重于控制学习算法的推导和实际结果的分析。

在介绍多层前馈神经网络时,我们已讨论了全局和局部逼近神经网络。已经指出,全局逼近网络是在整个权空间上对误差超曲面的逼近,即对输入空间中的任意一点,任意一个或多个连接权的变化都会影响到整个网络的输出,其泛化能力遍及全空间,如 BP 网络等。由于在全局逼近网络中,每一个训练样本都会使所有连接权发生变化,这就使相应的学习收敛速度极其缓慢。当网络规模较大时,这一特点使其实际上难以在线应用。而局部逼近网络只是对输入空间一个局部邻域中的点,才有少数相关连接权发生变化,如

CMAC、B 样条、RBF 和 FLN 网络等。鉴于在每次训练中只是修正少量连接权,而且可修正的连接权是线性的,因此其学习速度极快,并且可保证权空间上误差超平面的全局收敛特性。

正是由于全局与局部逼近神经网络的上述区别,使相应构成的神经网络控制系统,在选择结构方案和控制学习方法时,存在不同的侧重和考虑。将神经网络控制方法划分为基于全局与局部逼近网络的控制系统,既反映了各自的特点,同时又体现了迄今神经网络控制方法的发展过程。

值得指出的是,只考虑这两类网络的不同特点,在所述神经网络控制结构方案中,选择全局或局部逼近的神经网络,便可构成各种基于全局逼近或局部逼近网络的控制系统。因此我们已无需再针对这种新的类型划分,对其结构方案作重复性的论述了。本书之所以专门列写出这几小节,一是试图突出这种划分的特点;二是尝试通过某些具体系统的研究,使在一般性地介绍各种典型控制结构方案后,能够对整个神经网络控制系统有一个更加系统的了解。

2. 基于全局逼近神经网络的异步自学习控制系统

早期的工作大多假定对象为线性或非线性离散时间系统,或人为地离散化为这样的系统,以便利用全局逼近的静态 BP 网络学习差分方程中的未知非线性函数,从而获得控制作用的正向传播和输出误差的反向传播等。由此将网络或者作为被控对象的直接或逆动力学模型,或者作为神经网络控制器,或者作为性能评价估计器,以便构成各种控制结构方案。典型的结果如应用于倒立摆的具有再励学习的自适应评判控制(Barto, 1988; Anderson, 1989; Lee and Berenji, 1989)、自校正非线性控制(Chen, 1989)、模型参考非线性控制(Narendra, 1990)、非线性内模控制(Hunt, 1991)与非线性系统辨识(Chen, 1990)等。此外,将静态 BP 网络与动态环节相结合构成的所谓沿时间传播的动态 BP 网络(Werbos, 1990),以及前已指出的能描述状态空间表达式的部分递归网络,如 Elman 网络(Elman, 1990)、Jordan 网络(Jordan, 1986),实际上也都是全局逼近网络,由此构成的控制系统也都具有基于全局逼近网络控制方法的特点。

下面我们将结合两关节机械手的控制,具体讨论一种利用全局逼近网络的异步自学习控制系统。

不失一般性,考虑如下非线性连续时间闭环系统

$$\dot{x}_k(t) = f(x_k(t), u_k(t), t) \quad y_k(t) = g(x_k(t), t)$$

其中 $x_k(t) \in R^n$ 为 t 时刻第 k 步学习时的状态, $y_k(t) \in R^m$ 为输出, $u_k(t) \in R^r$ 。

取异步自学习控制律为

$$u_{k+1}(t) = u_k(t) + \phi(e_k(t), t)$$

这里 $\phi(\cdot, \cdot)$ 为异步自学习控制算子,且输出误差定义为

$$e_k(t) = y_d(t) - y_k(t)$$

其中 $y_d(t)$ 为给定的期望输出, $k=0, 1, \dots$ 为学习迭代次数。

由第 5 章的“学习控制”可知,异步自学习控制的基本思想是:第 $k+1$ 次学习时的输入 $u_{k+1}(t)$ 将基于第 k 次学习时的经验 $\phi(e_k(t), t)$ 和输入 $u_k(t)$ 获得,并且随着其中“有效”经验的不断积累而使 $e_k(t) \rightarrow 0$ 或 $y_k(t) \rightarrow y_d(t)$, $k \rightarrow \infty$ 。从而可望使实际输出经过“学习”而

逐渐逼近其期望输出。

典型的异步自学习控制方法包括早期的 PID 型学习控制,以及近期发展的最优学习控制、随机学习控制和自适应学习控制等,它们的本质区别在于学习算子 $\phi(\cdot, \cdot)$ 的具体形式,后者的选择需保证相应的学习收敛性。

基于上述讨论,下面我们将尝试利用全局逼近神经网络,通过定义二次型 Lyapunov 函数及相应的代价函数,以实现对学习算子 $\phi(\cdot, \cdot)$ 的逼近,从而构成所谓神经网络异步自学习控制系统。

若 $P \in R^{m \times m}$ 为正定对称加权阵,设 Lyapunov 函数为

$$v(e_k(t)) = e_k^T(t) P e_k(t)$$

相应的代价函数定义为

$$J = \frac{(v(e_{k+1}(t)) - v(\hat{e}_{k+1}(t)))^2}{v(\hat{e}_{k+1}(t))}$$

式中 $\hat{e}_{k+1}(t)$ 为由学习动态特性的网络模型给出的输出误差,此网络模型由 BP 网络实现,可首先利用典型的异步自学习控制方法,如 PID 型学习控制进行离线学习。而 $e_{k+1}(t)$ 则由如下收敛模型给出:

$$e_{k+1}(t) = A_c e_k(t)$$

其中 $A_c \in R^{m \times m}$ 为 Hurwitz 矩阵,它可有预先配置的极点或期望的收敛性能,相当于一个参考模型。

易知,这时

$$\frac{\partial J}{\partial u_k(t)} = \frac{\partial J}{\partial \hat{y}_{k+1}(t)} \frac{\partial \hat{y}_{k+1}(t)}{\partial u_k(t)}$$

上式右端的第一项可由上述式子解析求出,即

$$\frac{\partial J}{\partial \hat{y}_{k+1}(t)} = \frac{\partial J}{\partial \hat{e}_{k+1}(t)} \frac{\partial \hat{e}_{k+1}(t)}{\partial \hat{y}_{k+1}(t)} = 2 \left[\left(\frac{v(e_{k+1}(t))}{v(\hat{e}_{k+1}(t))} \right)^2 - 1 \right] \hat{e}_k^T(t)$$

而第二项则可由学习动态特性的网络模型反向传播给出,这与 BP 算法思路类似,只是对连接权 $w_{ij}(t)$ 的修正改成了对输入 $u_k(t)$ 的学习修正。此时

$$\frac{\partial \hat{y}_{k+1,i}(t)}{\partial u_{k,l}(t)} = \sum_{j=1}^{m_H} \frac{\partial \hat{y}_{k+1,i}(t)}{\partial o_j^H} \frac{\partial o_j^H}{\partial net_j^H} \frac{\partial net_j^H}{\partial u_{k,l}(t)} = \sum_{j=1}^{m_H} \delta_j o_j^H (1 - o_j^H) w_{jl}^H$$

其中 $\delta_j = \hat{y}_{k+1,i} (1 - \hat{y}_{k+1,i}) w_{ij}^H$ 为反向传播到隐层的广义误差, $\hat{y}_{k+1,i}(t)$ 表示 $y_{k+1}(t)$ 的第 i 个分量, $u_{k,l}(t)$ 表示 $u_k(t)$ 的第 l 个分量, o_j^H 表示隐层第 j 个神经元的输出, net_j^H 表示隐层第 j 个神经元的净输入, m_H 表示隐层神经元的个数。相应的神经网络异步自学习控制律为

$$u_{k+1}(t) = u_k(t) - \eta \frac{\partial J}{\partial u_k(t)} + \alpha (u_k(t) - u_{k-1}(t))$$

这里 $\eta > 0, 1 \leq \alpha < 1$ 与 BP 算法类似,分别为学习率和动量项参数,它们可进一步采用各种高阶快速学习算法。

图 3.45 给出了神经网络异步自学习控制系统的方框图。

事实上,由上面式子容易得到

因要求 $\Delta v(e_s(t))$ 负定, 故 Q 必须正定, 即 Lyapunov 方程 $A_c^T P A_c - P = -Q$, 在给定正定对称阵 Q 时, 需存在唯一正定解 P , 而这可由 A_c 为 Hurwitz 矩阵予以保证, 从而即可证得此神经网络异步自学习控制系统为学习渐近收敛 (稳定)。

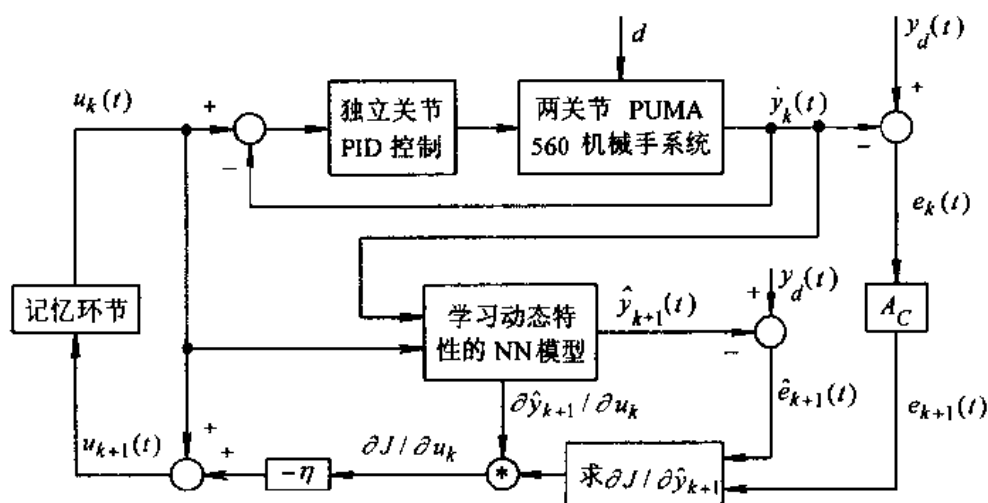


图 3.45 神经网络异步自学习控制系统方框图

3. 两关节机械手的仿真结果

以 PUMA560 机械手关节 2(肩关节)和关节 3(肘关节)的轨迹跟踪控制为例。图 3.46 给出了当随机干扰 $\sigma_d=0.1$ 时的仿真结果(即非重复情形)。这里取仿真时间 $T_{\text{TOT}}=2$ 秒, 采样周期 $T=0.001$ 秒;独立关节 PID 控制的各参数整定为 $K_{p1}=76.6, K_{d1}=40.8, K_{i1}=40.0$ (关节 2), $K_{p2}=57.7, K_{d2}=15.7, K_{i2}=40.0$ (关节 3);PID 型异步自学习控制各学习因子选择为 $KL_{p1}=0.2, KL_{d1}=KL_{i1}=0$ (关节 2), $KL_{p2}=0.2, KL_{d2}=KL_{i2}=0$ (关节 3); $A_c=0.707$;BP 网络的学习率 $\eta=0.87$, 动量项参数 $\alpha=0.7$;初始条件 $\theta_{10}=\theta_{20}=0$;跟踪精度要求为 $E_{\text{RMS},1}=E_{\text{RMS},2}=0.05$;最大控制力矩限制为 $\tau_{\text{max},1}=520\text{N}\cdot\text{m}$ (关节 2), $\tau_{\text{max},2}=260\text{N}\cdot\text{m}$ (关节 3)。

单纯利用 PID 型异步自学习控制和独立关节 PID 控制时的学习动态特性(假定此时无干扰),首先由 BP 网络进行离线学习获得。

从图 3.46 可以明显看出,由于各关节之间存在较强的耦合,单纯利用独立关节 PID 控制不能取得较好的跟踪效果($E_{\text{RMS}} \approx 0.5$),但在采用上述方法后,经过 $l=25$ 步的学习,就可使相应的精度达到 $E_{\text{RMS}} \approx 0.05$,即大约提高了 1 个数量级之多。

4. 讨论

由于神经网络控制器实际上是一个非线性控制器,因此一般难于对其进行稳定性分

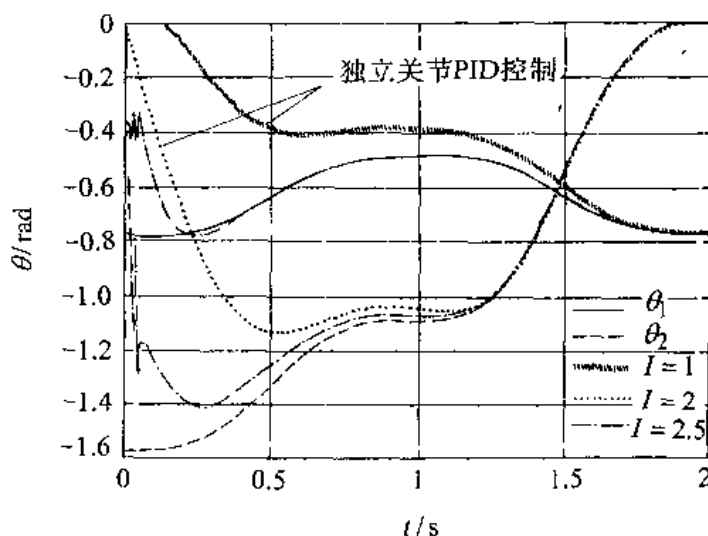


图 3.46 仿真结果

析。前已指出,全局逼近网络在控制系统中的作用,主要体现在两个方面:一是提供一个类似于传统控制器的神经网络控制器;二是为神经网络控制器进行在线学习,提供性能指标关于控制误差梯度的反向传播通道,如需要建立被控对象的正向网络模型等。除此之外,结合稳定性分析,对神经网络控制结构方案进行特别设计,常常可以为困难的分析问题,提供一个有效的解决途径。本节的例子较好地说明了这一点。

从理论上说,相对于在权空间上局部逼近误差超曲面的网络而言,全局逼近网络在总体上具有更大泛化能力。但对我们最感兴趣的轨迹附近,就某一局部而言,当误差超曲面比较复杂时,由于全局逼近网络存在严重的局部极值问题,再加之其缓慢的学习收敛速度,其泛化能力往往还不如局部逼近网络,因为后者对局部的逼近是线性“全局”最优的。一般说来,在基于全局逼近网络的控制系统中,缺乏鲁棒性和在线学习能力较差,通常成为其实际应用的两个限制性“瓶颈”问题。

3.7.4 基于局部逼近神经网络的控制

近几年的进展主要集中在采用局部逼近网络构成的控制系统。主要结果包括:小脑模型关节控制器(CMAC)控制(Albus, 1975; Miller, 1987, 1989, 1990; Kraft, 1990),此法已广泛应用于机器人控制。网络的特点是局部逼近,学习速度快,可以实时应用。不足之处是采用间断超平面对非线性超曲面的逼近,可能精度不够,同时也得不到相应的导数估计;采用高阶 B 样条的 BMAC(Lane, 1992)控制,则部分弥补了 CMAC 的不足,但计算量略有增加;基于高斯径向基函数(RBF)网络的直接自适应控制(Poggio, 1984; Sanner and Slotine, 1991),这是有关非线性动态系统的神经网络控制方法中,较为系统、逼近精度最高的一种方法。但它需要的固定或可调连接权太多,且高斯径向非线性函数的计算也太多,利用目前的串行计算机进行仿真实现时,计算量与内存过大,很难实时实现。其他基于局部逼近网络或相联网络的方法,如模糊神经网络控制系统,将在下一节专门予以叙述。

1. 指标驱动 CMAC 控制

在第 3.4.1 节我们讨论了 CMAC 神经网络。下面我们将针对一类带参数变化的未知单输入单输出系统,介绍一种基于如上升时间、超调量、稳态误差等时域指标的 CMAC 间接自校正控制系统。

与图 3.39 类似,图 3.47 给出了指标驱动 CMAC 控制系统的结构方案示意图。

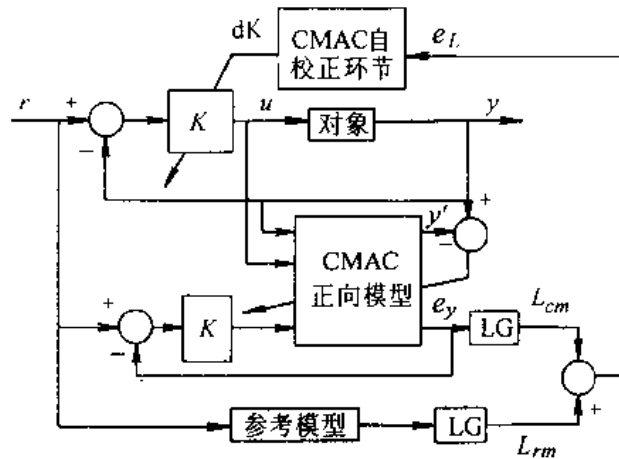


图 3.47 指标驱动 CMAC 控制系统的结构方案

从图中可以看出,该系统可分为 4 个部分:(1) 常规控制器与单位反馈;(2) 对象的 CMAC 正向模型;(3) 指标误差计算环节;(4) CMAC 自校正模块。系统中采用两个 CMAC 模块,一个用于建立被控对象的正向网络模型,另一个用作神经网络非线性映射器。CMAC 模型与参考模型之间的输出被用来产生指标误差向量。CMAC 自校正模块将此指标误差映射为控制器增益变化,从而修正相应的控制规律。

为了简单起见,这里的常规控制器暂采用比例控制器。若采用 PID 控制器,则只需再增加两个 CMAC 自校正模块,或直接采用具有三个输出的 CMAC 自校正模块,以便网络输出对应于 PID 三个分量的变化。对于比较复杂的控制对象,可进一步采用非线性 PID 控制或其它自校正控制律。

与此同时,CMAC 正向模型将在线地学习被控系统的动态特性,如果可能,它还将学习传感器和执行机构的动态模型。前面已指出,这需要提供某些先验知识,如系统的阶次等,以便给出差分方程的结构模型。

在指标误差计算环节中,首先由两个指标生成模块 LG(Label Generation)分别根据闭环 CMAC 正向模型与参考模型的每个响应曲线,计算各自的时域指标向量,然后完成如下的指标误差计算,即

$$e_L = L_{cm} - L_{rm}$$

其中 L_{cm} , L_{rm} 分别为闭环 CMAC 正向模型与参考模型的时域指标向量。该环节在功能上相当于模式识别中的特征抽取。

核心的 CMAC 自校正模块,将此指标误差向量映射为比例控制的增益变化,以便对控制器进行学习修正。众所周知,人们习惯使用并且具有明确物理意义的时域与频域指标

等与控制规律之间的关系,不可能存在任何解析关系。在这种情况下,应用神经网络通过学习建立两者的非线性映射关系,可能是一种较好的选择。

一般说来,指标向量可以分成两类,即品质指标与控制约束。对于定常或时变系统的主要动态特性,这里采用上升时间、超调、稳态误差等时域指标描述。针对具体的应用,当然也可以采用其他类型的指标。而控制约束则反映了实际所能采取的最大控制量。将这些数字量作为系统性能的特征,既符合习惯使用的性能表达,同时又能大大地压缩所需的数据训练量。

如果说上述指标反映了系统暂态响应的主要特征,那么上面式子的指标误差向量则体现了控制系统相对于参考模型的性能。这里的参考模型给出了期望的时域指标。进一步地,我们可以对此误差向量采用各种范数测度,如对其归一化或加权平均等,以便获得更加简洁有效的性能指标表示。

在上述结构方案中,CMAC 自校正模块在监督学习下将完成如下非线性映射

$$e_L \rightarrow \Delta K$$

难点是如何提供训练样本集。这可以说是本系统成败的关键。为此我们采用如下的方法,离线地提供训练样本集。

- (1) 在图 3.47 由网络模型组成的闭环控制系统中,整定一个初始比例增益 K_0 ;
- (2) 给定期望的时域性能指标 L_{rm} ;
- (3) 令 $r(t) = 1(t)$,对闭环网络模型的单位阶跃响应,计算相应的时域性能指标 $L_{m,0}$,从而得到指标误差向量 $e_{L,0}$;
- (4) 在可能的工作范围内,以一定的量化等级,将控制器增益进行摄动 ΔK ,按(3)得到相应的指标误差向量 $e_{L,i}$,并记录下此对样本,这里 $i=1,2,\dots,N$ 。

上述样本集反映了指标误差向量 $e_{L,i}$ 与控制器增益 ΔK 之间的点集映射关系。但用如此 N 对样本集训练 CMAC 自校正模块,可能存在一个严重的问题,即在线工作时,某个指标误差向量可能根本不出现在训练样本集中,从而有可能使网络的输出为 0。这个问题不单是本系统存在,在前面介绍的所有全局或局部逼近网络中都会出现此类问题,这实际上体现了网络是否具有足够的泛化能力。

应该注意,这里得到的 CMAC 自校正模块是否真正有效,还将取决于参考模型或期望的性能指标是否选择的合理,即对于给定的控制器和增益范围,系统是否真能达到这一指标。例如,对二阶系统,我们就不能要求在比例控制下能同时达到超调量为 0 和稳态误差为 0。

上述方法简单、实用,比较接近于我们在设计 PID 控制时所采用的经验。它通过学习来建立响应与 PID 增益之间的关系。显然,这种映射关系在进行大量离线训练之后,尚可进行在线学习,它不需要指定 CMAC 自校正网络的参数化结构模型。

2. 结果与讨论

该法已实际应用于燃气涡轮发动机系统。在 Pham 等(1995)进行的仿真研究中,使用了数字控制系统与采样模型。被控对象假定为二阶线性系统,其参数任意选择为 $\omega_n = 1.0, \zeta = 0.3$ 。参考模型选为比例增益为 6.5 时的闭环等价系统。CMAC 自校正模块训练样本集的容量 $N=17$ 。初始增益 $P_0=5.0$ 。增益变化范围 $[1.0, 9.0]$,步长为 0.5。指标

误差向量 e_L 由超调量 σ 、上升时间 t_r ，以及稳态误差 SSE 组成。采用单位阶跃响应。8 个仿真结果如表 3.5 所示，其中 P_i 为初始控制增益。表 3.6 给出了分别对应于上述 8 个仿真之每次迭代时的增益值。

表 3.5 仿真结果

	P_i	ω_n^*, ζ^*	P_{ref}	$\omega_n^{ref}, \zeta^{ref}$	e_L 初值	e_L 终值
1	3.0	1.0, 0.3	6.5	1.0, 0.3	23.09, -8.90, -0.10	0.34, -0.42, 0.00
2	3.0	0.8, 0.5	6.5	1.0, 0.3	36.50, -12.61, -0.34	15.41, 0.85, -0.22
3	3.0	1.2, 0.2	6.5	1.0, 0.3	14.88, -11.66, 0.08	1.03, -0.85, 0.10
4	3.0	0.8, 0.2	6.5	1.0, 0.3	11.91, -13.85, -0.24	-0.30, -2.50, -0.18
5	3.0	1.2, 0.5	6.5	1.0, 0.3	36.16, -10.40, 0.00	14.96, 0.70, 0.08
6	3.0	1.0, 0.3	5.0	1.2, 0.4	9.64, -7.24, -0.18	0.45, -2.28, -0.11
7	3.0	1.0, 0.3	OL	1.0, 0.3	16.26, -25.71, 0.88	16.26, -25.74, 0.88
8	3.0	1.2, 0.1	6.5	1.2, 0.4	22.95, -12.15, -0.08	1.02, -0.33, -0.02

表 3.6 8 个仿真结果中每次迭代时的增益值

在线迭代次数	仿真 1	仿真 2	仿真 3	仿真 4	仿真 5	仿真 6	仿真 7	仿真 8
0	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00
1	5.30	5.59	4.56	5.57	4.64	5.30	—	5.45
2	6.58	5.62	5.10	5.21	5.87	5.13	—	6.19
3	6.44	—	4.99	5.04	5.98	4.97	—	—
4	6.40	—	—	4.90	—	4.81	—	—
5	6.38	—	—	4.77	—	4.65	—	—
6	—	—	—	4.68	—	4.49	—	—
7	—	—	—	—	—	4.33	—	—
8	—	—	—	—	—	4.18	—	—
9	—	—	—	—	—	4.03	—	—
10	—	—	—	—	—	3.98	—	—

从表 3.6 中可以看出，当增益变化 ΔP 落入某个预先设定的阈值后，如这里的 0.02，学习过程必定收敛。我们现在来分析表 3.5 的仿真结果。

在第 1 个仿真结果中，对象和参考模型具有同样的参数，因此有可能得到零指标误差。由表 3.6 可知，此时自校正过程只需 5 次迭代即可完成，其中全部变化值的 95% 发生在头两步。在第 2 到第 5 个仿真例子中，我们让对象参数与参考模型略有不同，此时指标误差虽迅速减小，但不会到 0。在第 6 个仿真结果中，选择参考模型参数使其与对象模型略有不同，这时收敛速度明显减慢，尽管在第一步已发生了 65% 的变化。在第 7 个仿真结

果中,对给定的二阶被控系统,我们选择了一个“坏”的期望指标,即要求其响应具有大的上升时间和零稳态误差,此时由于 CMAC 网络表现出对该指标误差“无知”,从而导致网络输出为 0。最后一个仿真例子说明了映射特性或网络拓扑的重要性。此时虽然所选参数与前几次仿真不同,但系统不仅可获得零指标误差,而且只需大约两步就可达到。

通过考察样本数据,我们可进一步归纳出期望映射 $e_L \rightarrow \Delta P$ 之间的定性关系,即

e_L	ΔP
+, -, -	+
0, 0, 0	0
-, +, +	--

本小节介绍了一种利用 CMAC 自校正模块的比例控制间接自适应系统。仿真结果表明,该法简单、实用,具有明显的几何意义,可进一步推广为 PID 控制或其它自校正控制,以便应用于更加一般的时变线性系统或非线性系统。

综上所述,作为另外一种处理非线性、不确定性的有力工具,神经网络控制方法在不长的时间内已取得长足的进展,尽管人们已认识到它的许多局限性。例如,网络本身的黑箱式内部知识表达,使其不能利用初始经验进行学习,易于陷入局部极小值;分布并行计算的优点还有赖于硬件实现技术的进步等等。但作为一种控制方法,我们认为存在的主要问题是:(1) 缺乏一种专门适合于控制问题的动态神经网络。上述方法,不论是全局逼近还是局部逼近的方法,就其本质都是用静态网络处理连续时间动态系统的控制问题,这就不可避免地带来了差分模型的定阶及网络规模随阶次迅速增加的复杂性问题;(2) 鲁棒性较差使其较少实际应用。神经网络的泛化能力在相当程度上决定了控制系统的鲁棒性。全局逼近方法的泛化能力受大量局部极值与缓慢学习收敛速度的制约,而上述局部逼近方法则受存储容量与实时性的严重限制,这种矛盾无法用上述网络模型解决。

3.7.5 模糊神经网络控制

1. 引言

日前,将符号主义的模糊逻辑与连接主义的神经网络结合,发展模糊神经网络控制方法(Lee, C. C. and Berenji, H. R., 1989; Lin, C.-T., 1991; Nomura, H., 1992; Hayashi, Y., 1992; Jang, J. S., 1992),为上述困难的解决带来了新的希望。一般说来,利用连接主义表达的模糊逻辑控制器,必然引入了学习机制,同时也给这种局部逼近网络带来了诸多结合的优点,如存储容量的减小,泛化能力的增加,以及连接主义结构的容错性等。特别地,模糊逻辑处理连续时间动态系统的能力,可能为动态神经网络的研究带来根本的出路。因此,无论从模糊控制,还是从神经网络控制研究的角度来看,两者的结合(大致还包括专家系统)可以说代表了该领域未来的主要发展方向。

模糊神经网络主要有三种结构:(1) 输入信号为普通变量,连接权为模糊变量;(2) 输入信号为模糊变量,连接权为普通变量;(3) 输入信号与连接权均为模糊变量。它们尚可根据网型及学习算法中的点积运算是使用模糊逻辑运算(fuzzy logic operations),还是使用模糊算术运算(fuzzy arithmetic operations),而分成常规(regular)和混合(hybrid)型模糊神经网络。

有关模糊神经网络的想法是由 S. C. Lee 及 E. T. Lee 首先提出的(1974,1975),他们基于 0~1 之间的中间值,推广了 MP 模型。1990 年 Takagi 在一篇综述文章中讨论了神经网络与模糊逻辑的融合问题,但当时除 Keller(1985)提出在感知机中加入隶属函数以及 Yamakawa 的模糊神经元(1989)外,有关的研究工作极少。此后,Yamakawa(1990,1992),特别是 Gupta(1990~1992)提出了大量的模糊神经元模型。但大量的工作主要集中在网型和学习算法的研究上。如 C. C. Lee 及 H. R. Berenji 基于近似推理和再励学习的 ARIC(1989~1992)及 GARIC(1992~1994)、J. S. Jang 的基于自适应网络的模糊推理系统 ANFIS(1992),以及基于 ART 的模糊 ARTMAP(Carpenter 等,1992)等。不过这些结果采用的主要是网型 I,或者说仅是由神经网络实现的模糊逻辑控制器,这就使其不能充分发挥模糊神经网络将高层次的符号主义与具感知功能之连接主义结合的优点。

在本小节我们将在第 3.5 节讨论的基础上,进一步介绍模糊神经网络控制系统。出于与前两小节同样的理由,这里将侧重于具体系统的研究,而不会涉及全面系统的讨论。

2. 具有再励学习的神经网络模糊 BOXES 控制系统

BOXES 算法首先是针对倒立摆控制提出的(Michie and Chambers,1968)。它的基本思想是经过量化,把问题空间(如倒立摆的状态空间)分解为确定数量的非重叠区域(box),然后假定每个 box 中含有一个局部精灵(local demon)。若将倒立摆直至失败的时间作为性能指标,则此局部精灵将相对于该性能指标进行学习,以确定状态进入某个 box 时,如何选择控制作用(向左或向右),并且对失败时间进行估计。而全局精灵(global demon)则始终监视系统状态,并且在出现失败时警戒局部精灵向失败学习。但 Michie 与 Chambers 的上述 BOXES 算法简单地将状态空间量化为互不重叠的 box,使 box 之间没有做任何泛化,而且如何进行划分也需要更多的先验知识。

为了克服上述困难,本小节通过对各个状态变量定义输入隶属函数,以便将各个 boxes 的刚性边界扩展为重叠的模糊柔性边界,即采用所谓模糊 box 来分割问题空间,并利用神经网络予以实现。如此不但解决了简单量化所带来的信息丢失,而且增加了重要的泛化能力,并且可以通过神经网络的学习机制,对模糊 box 的空间位置分布进行学习修正。图 3.48 给出了神经网络模糊 BOXES 控制系统的一般框图。

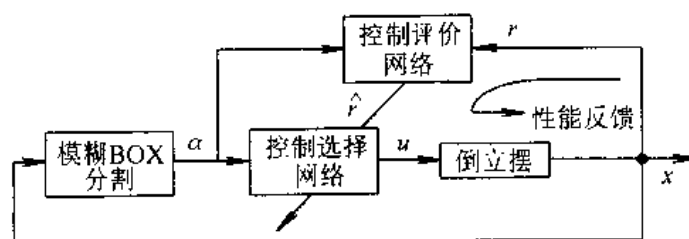


图 3.48 神经网络模糊 BOXES 控制系统的一般框图

从图中可以看出,该系统主要由控制评价网络(CEN)与控制选择网络(CSN)组成。模糊 box 分割环节将状态空间划分为 $N_B = \prod_{i=1}^n N_i$ 个模糊 box,并输出各模糊 box 的点火强度或规则适用度 α_i ,其中 N_i 为第 i 个状态变量 x_i 所对应的语言变量值的个数, $i=1,2,$

$\dots, n, j=1, 2, \dots, N_B$ 。因此,采用模糊 BOXES 算法实质上就是用模糊 box 代替模糊控制中的输入隶属函数或神经网络的激活函数分割状态空间,显然这更具一般性。

(1) 控制评价网络(Control Evaluation Network 或 CEN)

该网络在整个控制系统中相当于一个需要进行再励学习的“教师”。其作用有二:一是对系统的状态进行评价,给出相应的再励预测,并进一步给出用以指导 CSN 进行学习的内部再励信号;二是利用再励学习,对 N_a 个具有非零点火强度的模糊 box(即状态进入的各模糊 box)进行评分(score),以获知哪个模糊 box 是“安全的”(评分高),哪个是“危险的”(评分低),其中 $N_a \leq N_B$,这部分内容将在稍后一些叙述。

进一步地,假定 CSN 在施加控制作用 $u(t)$ 之后,系统的新状态为 $x(t+1)$,相应的点火强度为 $\alpha(t+1)$ 。此时,CEN 将对此新状态进行评价,即有

$$p(t+1) = \sum_{i=1}^{N_B} v_i(t) \alpha_i(t+1)$$

这里, $p(t+1)$ 一般称为再励预测(prediction of reinforcement),再励期望或加权评分,且 $v_i(t)$ 为第 i 个模糊 box 在 t 时刻的评分,它实际上也就是 CEN 的连接权。

故利用前一时刻的 $p(t)$ 及外部再励信号 $r(t+1)$,内部或直觉再励信号可计算如下。

$$\hat{r}(t+1) = \begin{cases} 0 & \text{initial state} \\ r(t+1) - p(t) & \text{failure} \\ r(t+1) + \gamma p(t+1) - p(t) & \text{success} \end{cases}$$

其中 $0 \leq \gamma \leq 1$ 为折扣率(discount rate),它实际上反映了对当前评价的置信度(Witten, 1977)。一般可取 $\gamma=0.9 \sim 0.95$ 。

对倒立摆控制问题,这里假定外部再励信号为

$$r(t+1) = \begin{cases} -1 & \text{failure} \\ 0 & \text{success} \end{cases}$$

由上面两式子得知:第一,当倒立摆未失败时,由于 $r(t+1)=0$,故 $\hat{r}(t+1)=\gamma p(t+1) - p(t)$,即 $\hat{r}(t+1)$ 为当前状态的再励预测(折扣 γ)与前一时刻再励预测之差。因此若假设 $\gamma=1$ 时, $\hat{r}(t+1)$ 增加意味着这是奖励事件(rewarding events), $\hat{r}(t+1)$ 减少则意味着这是惩罚事件(penalizing events);第二,当失败发生时,由于失败状态不可能出现在任意模糊 box 中(由定义),因此失败时所有 $\alpha_i(t+1)=0$,从而由前面式子, $p(t+1)=0$ 。又此时 $r(t+1)=-1$,故 $\hat{r}(t+1)=-1-p(t)$ 。即一个不可预测的失败将导致 $\hat{r}(t+1)$ 为负。

(2) 控制选择网络(Control Selection Network 或 CSN)

该网络的作用相当于一个在 CEN 内部再励信号指导下进行学习的模糊控制器,但有如下两个特点:一是每个模糊 box 中均包含一个可进行学习的输出模糊语言变量,因此表征输入输出语言变量之间关系的模糊规则库(定性关系)已隐含定义在各模糊 box 中,不再有通常的连接结构,虽然这种关系在量级上仍然可以通过学习调整;二是为了利用链式法则推出相应的学习算法,CSN 采用了这里提出的局部拟 COA 清晰化法。

(3) 局部拟 COA 清晰化方法(Defuzzification)

通常使用的清晰化方法有三种,即最大判据法(The Max Criterion Method)、MOM 法(The Mean of Maximum Method)和 COA 法(The Center of Area Method)。一般说来,COA 法(重心法)能产生一个更加优越的结果(Larkin, L. I., 1985),但此法难于给出一个完整的解析公式,使应用链式法则推导学习算法时遇到较大的困难。

为了绕开上述困难,我们自然可联想到其他模糊推理方法,如利用 Takagi 型的模糊推理方法。但利用模糊规则库的方法,由于物理意义明确,在输入输出语言变量之间的定性关系十分清楚的前提下,则仍然不失为一种好方法。

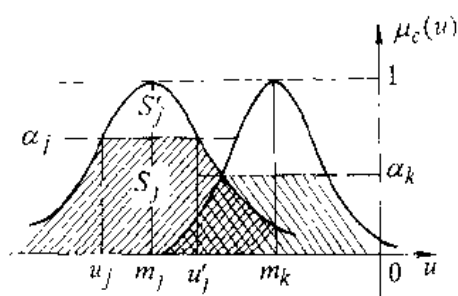


图 3.49 局部拟 COA 清晰化方法

为了在这种情况下也能得到相应的学习算法,我们采取了两项措施:(1) 如图 3.49 所示,对 S'_j 的面积简单地用一个三角形的面积近似,从而可得到重心的解析公式,这就是所谓对 COA 法的拟解析化;(2) 不考虑输出隶属函数的交叉重叠,而是通过学习获得相同的效果,这就是“局部”的含义。

此时,容易推得

$$S'_j = (1 - \alpha_j) \sigma_j \sqrt{\ln \alpha_j^{-1}}$$

从而

$$S_j = \begin{cases} \sqrt{\pi} \sigma_j - S'_j & \alpha_j \neq 0 \\ 0 & \alpha_j = 0 \end{cases}$$

令

$$\bar{S}_j = \frac{S_j}{\sum_{k=1}^{N_a} S_k} = \frac{\sqrt{\pi} \sigma_j - (1 - \alpha_j) \sigma_j \sqrt{\ln \alpha_j^{-1}}}{\sum_{k=1}^{N_a} [\sqrt{\pi} \sigma_k - (1 - \alpha_k) \sigma_k \sqrt{\ln \alpha_k^{-1}}]}$$

故 CSN 的清晰化控制作用为

$$u = \frac{\sum_{j=1}^{N_a} m_j S_j}{\sum_{k=1}^{N_a} S_k} = \sum_{j=1}^{N_a} m_j \bar{S}_j$$

这里 N_a 为由状态 $x(t)$ 激活的具有非零点火强度的模糊 box 的个数。

3. 再励学习算法

神经网络的学习方法一般分为三种类型:监督学习、再励学习与无人监督学习。监督学习虽然具有最高的学习效率,但它需要教师提供网络的期望输出。对于神经网络控制器,也就是需要提供期望的控制信号,这显然难于实现。无人监督学习利用网络的输入数据构造内部教师模型,不再接受其它信息,相当于自组织聚类方法,但学习效率十分低下。再励学习(reinforcement learning)介于两者之间,只需要输出一个标量评价值,如使用倒立摆的成功或失败(0/—1)二进制信息。

作为人与动物行为的一种普遍学习机制,“再励”这一术语最早源于心理学。它的一般

性理论与随机自动机理论有关。早期的工作可追溯到 Bush 等(1958)及 Tsetlin(1973)所得的结果。

(1) 控制评价网络(CEN)的再励学习算法

前已指出, $\hat{r}(t+1)$ 的变化反映了对成功或失败事件的奖励或惩罚。事实上, 这将通过如下再励学习算法反映到 CEN 对 N_a 个模糊 box 的评分上, 即根据 $\hat{r}(t+1)$ 的变化, 也将对评分 v_i 进行奖惩, 故有

$$v_i(t+1) = v_i(t) + \beta \hat{r}(t+1) \bar{e}_i(t)$$

式中 $0 < \beta \leq 1$ 为 CEN 的学习率, $\bar{e}_i(t)$ 称为传导性迹 (Eligibility Trace), 且 $i = 1, 2, \dots, N_a$ 。

突触递质的传导性 (eligibility) 是由 Klopff 的理论首先提出的 (1972), 更早期的工作可在 Farley and Clark (1954) 及 Minsky (1954) 的文献中找到。它实质上反映了人脑遗忘 (或记忆) 与刺激频度之间的关系。简单地说, 可将各模糊 box 的 $\bar{e}_i(t)$ 视为该 box 被激活或被状态进入的频度。

为了简单起见, 这里只产生一个按指数衰减的传导性迹。它可由如下线性差分方程描述为

$$\bar{e}_j(t+1) = \lambda \bar{e}_j(t) + (1 - \lambda) \alpha_j(t)$$

其中, $0 \leq \lambda < 1$ 为指数衰减率, $j = 1, 2, \dots, N_a$ 。

从前面式子易知, 当状态从“危险的”box (评分低) 到“安全的”box (评分高) 时, $\hat{r}(t+1) > 0$ 增加为奖励事件, 此时 CEN 也将奖励评分 $v_i(t+1)$, 使其增加 (指出其“安全性”); 否则, 则意义相反。

(2) 控制选择网络(CSN)的梯度型学习算法

若将 CEN 给出的再励预测 $p(t+1)$ 作为指导 CSN 进行学习的性能指标, 则 CSN 的控制选择策略或输出隶属函数的变化 (中心、宽度) 需使 $p(t+1)$ 为最大, 为此可采用通常的链式法则推导出相应的梯度型学习公式。

由前面的式子

$$\frac{\partial u}{\partial m_j} = \bar{S}_j$$

$$\frac{\partial u}{\partial \sigma_j} = m_j \frac{\partial \bar{S}_j}{\partial \sigma_j} = m_j \frac{\partial \bar{S}_j}{\partial S_j} \frac{\partial S_j}{\partial \sigma_j} = \frac{m_j}{N_a} (1 - \bar{S}_j) [\sqrt{\pi} - (1 - \alpha_j) \sqrt{\ln \alpha_j^{-1}}] \sum_{i=1} S_i$$

又由于

$$\frac{\partial p(t+1)}{\partial m_j} = \frac{\partial p(t+1)}{\partial u} \frac{\partial u}{\partial m_j}, \quad \frac{\partial p(t+1)}{\partial \sigma_j} = \frac{\partial p(t+1)}{\partial u} \frac{\partial u}{\partial \sigma_j}$$

尽管 $u(t)$ 对 m_j 和 σ_j 的依赖是直接的, 但 $p(t+1)$ 与控制作用 $u(t)$ 之间只有间接的关系, 难于推出显式的解析表达式。事实上, 当 $u(t)$ 作用时, 首先需经被控对象的动态过程产生新状态 $x(t+1)$, 然后再经 CEN 的传递才能得到 $p(t+1)$ 。故可近似取

$$\frac{\partial p(t+1)}{\partial u} \approx \text{sign} \left\{ \frac{\hat{r}(t+1)}{u(t) - u(t-1)} \right\}$$

故相应的梯度型学习算法为

$$m_j(t+1) = m_j(t) + \eta \hat{r}(t+1) \frac{\partial p(t+1)}{\partial m_j},$$

$$\sigma_j(t+1) = \sigma_j(t) + \eta \hat{r}(t+1) \frac{\partial p(t+1)}{\partial \sigma_j},$$

其中, $0 < \eta \leq 1$ 为 CSN 的学习率, $j=1, 2, \dots, N_B$ 。

由上述两式可知, 作为体现奖惩的性能指标, $\hat{r}(t+1)$ 的变化不仅影响到 CEN 的评分, 而且也将影响到 CSN 对控制作用的选择, 或对输出隶属函数的奖惩。总的看来, 输出隶属函数的变化或 CSN 对控制作用的选择, 需沿模糊 box 再励预测的梯度方向, 使其从任意位置移动到更“安全”或评分更高的模糊 box。

4. 在倒立摆控制中的应用

应该指出, 迄今为止相当多的模糊神经网络都是结合控制问题, 特别是倒立摆控制问题提出的 (Lee, 1989~1991; Hayashi, 1989; Berenji, 1991~1993; Lin, 1991; Jang, 1992)。作为智能控制研究中的一个经典对象, 在倒立摆问题中应用神经网络方法, 首推 Widrow 等人 (1964, 1987) 的工作。但较具代表性的结果则主要是由美国加州大学柏克利分校, 以 L. A. Zadeh 为首的“fuzzy group”作出的。如 C. C. Lee 与 H. R. Berenji 基于近似推理和再励学习的 ARIC (1989~1992) 及 GARIC (1992~1994), 以及 J. S. Jang 的基于自适应网络的模糊推理系统 ANFIS (1992) 等。他们从 fuzzy logic 的角度分别发展了 Barto (1983) 与 Anderson (1989) 等人以及 Werbos (1990) 等人的结果。而 Barto, 特别是 Anderson 后来的结果, 则是研究有关倒立摆再励学习控制的早期经典之作。

图 3.50 给出了一级倒立摆的示意图。

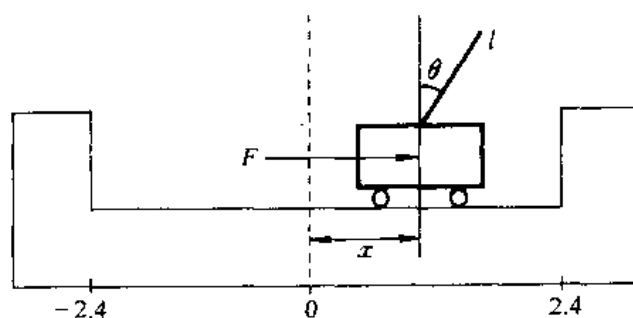


图 3.50 倒立摆系统示意图

考虑摩擦时倒立摆的运动方程可由如下非线性微分方程描述。

$$\ddot{\theta} = \frac{g \sin \theta + \cos \theta \left[\frac{-F - m_p l \dot{\theta}^2 \sin \theta + \mu_c \text{sgn}(\dot{x})}{m_c + m_p} \right] - \frac{\mu_p \dot{\theta}}{m_p l}}{l \left[\frac{4}{3} - \frac{m_p \cos^2 \theta}{m_c + m_p} \right]}$$

$$\ddot{x} = \frac{F + m_p l [\dot{\theta}^2 \sin \theta - \ddot{\theta} \cos \theta] - \mu_c \operatorname{sgn}(\dot{x})}{m_c + m_p}$$

其中典型数据为: $g=9.8\text{m/s}^2$ (重力加速度), $m_c=1.0\text{kg}$ (小车质量), $m_p=0.1\text{kg}$ (杆的质量), $l=0.5\text{m}$ (杆的半长), $\mu_c=0.0005$ (小车相对于导轨的摩擦系数), $\mu_p=0.000\ 002$ (杆相对于小车的摩擦系数)。 F 为作用于小车上的力, 它相当于图 3.48 中的 u , 其他符号与通常约定相同。

若令 $a_1=(m_p/m)l\dot{\theta}^2\sin\theta$, $a_2=(m_p/m)l\ddot{\theta}\cos\theta$, $a=\ddot{x}$, $a_F=(1/m)F$, 这里 $m=m_c+m_p$ 。则容易得到如下倒立摆方程的简化形式。

$$\begin{aligned}\ddot{\theta} &= \frac{3}{4l}(g\sin\theta - a\cos\theta - \frac{1}{m_p l}\mu_p\dot{\theta}), \\ \ddot{x} &= a_F + a_1 - a_2 - \frac{1}{m}\mu_c\operatorname{sgn}(\dot{x})\end{aligned}$$

进一步地, 设 $x=[\theta\ \dot{\theta}\ x\ \dot{x}]^T$, 则有如下状态方程为

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= f_1 = \frac{3}{4l}(g\sin x_1 - a\cos x_1 - \frac{1}{m_p l}\mu_p x_2) \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= f_2 = a_F + a_1 - a_2 - \frac{1}{m}\mu_c\operatorname{sgn}(x_4)\end{aligned}$$

式中, $f_1=\ddot{\theta}$, $f_2=\ddot{x}$ 。

在上述动态方程的仿真中, 我们采用了自动变步长 Runge-Kutta-Fehlberg 数值积分法。采样周期 $T=20\text{ms}$ 。失败状态(failure)定义为 $|\theta|>12^\circ$ 或 $|x|>2.4\text{m}$ 。初始条件取为 $\theta(t_0)=2^\circ$, 其他均为 0。

倒立摆各状态变量模糊 box 分割的定义, 以及输出隶属函数与 9+4 条简单模糊规则, 在此从略。

仿真时 CEN 与 CSN 的各参数选择为: $\eta=0.7$, $\gamma=0.95$, $\beta=0.5$, $\lambda=0.8$ 。图 3.51 给出了采用本方法的学习曲线, 并与 Michie 与 Chambers、Barto 及 Anderson 的仿真结果进行了比较。

从图中可以看出, 采用本小节的方法, 可在学习失败 47 次后, 使杆直至失败的时间达到大约 80 000 多个采样周期, 即可使杆“稳定”26.7 分钟以上。

5. 讨论

上面我们具体研究了神经网络模糊 BOXES 再励学习控制系统。基本思路是: 利用模糊 box 分割问题空间, 使每个模糊 box 不仅具有 CEN 给出的评分, 含有作为控制作用的输出语言变量, 而且整个模糊 box 还隐含定义了模糊规则库。为了得到 CSN 的学习算法, 文中还提出了局部拟 COA 清晰化算法, 并成功地将此再励学习控制系统应用于倒立摆系统的控制中。进一步的研究包括: 模糊 box 空间分割的自组织学习算法, 模糊 box 的微分流形描述, 以及由此对 FLC、CMAC、BMAC、RBF 及各种模糊神经网络进行的统一的稳定性与鲁棒性分析; 将该法推广应用于多关节机械手的非线性自适应控制等。

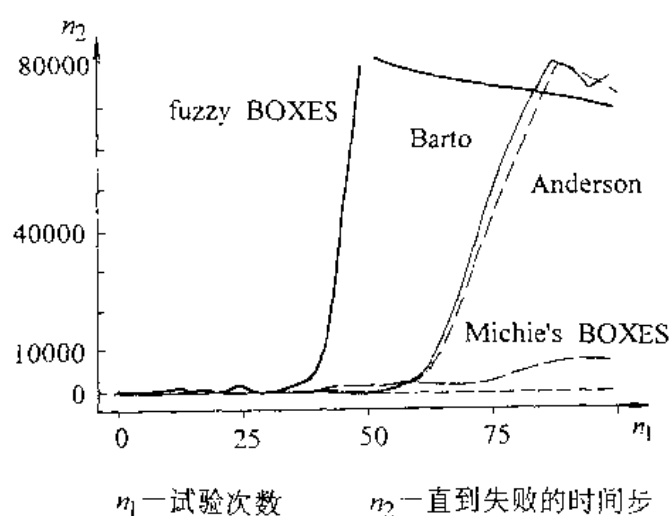


图 3.51 学习性能的比较

总的看来,神经网络(ANN)与模糊逻辑(FL),作为处理信息的不确定性、非线性和不适定性(III-Defined),以及模拟人脑感知与思维功能的两种最重要的工具,存在着许多相似之处。例如,它们本质上都表达了一个高度非线性的输入输出关系,输出都不再是AI符号逻辑系统中的二值布尔代数值,都具有泛化能力(generalization ability),并且 FLC 中的输入隶属函数与神经元的感受野(receptive field),清晰化时的加权平均法与神经网络的乘积和之间,在原理上也都具有相同的作用。因此,设想将两者的优点结合起来,发展模糊神经网络(FNN),必将为复杂系统的控制问题提供一种更加有力的工具。

3.7.6 有待解决的问题

在短短几年时间里,神经网络控制的研究,无论从理论到应用都取得了许多可喜的进展,应该说是相当惊人的。但我们必须看到,人们对生物神经网络的研究与了解还相距甚远,所使用的形式神经网络模型无论从结构还是网络规模,都是真实神经网络的极简单模拟,因此神经网络控制的研究还非常原始,而迄今的结果也大都停留在仿真或实验室研究阶段,完整、系统的理论体系,大量艰难而富有挑战性的理论问题尚未解决,真正在线应用成功的实例也待进一步发展。

我们认为从总体上来看,今后的研究应致力于如下几个方面:

- 基础理论性研究,包括神经网络的统一网络模型与通用学习算法,网络的层数、单元数、激发函数的类型、逼近精度与拟逼近非线性映射之间的关系,持续激励与收敛,神经网络控制系统的稳定性、能控性、能观性及鲁棒性等;
- 研究专门适合于控制问题的动态神经网络模型,解决相应产生的对动态网络的逼近能力与学习算法问题;
- 神经网络控制算法的研究,特别是研究适合于神经网络分布式并行计算特点的快速学习算法;
- 对成熟的网络模型与学习算法,研制相应的神经网络控制专用芯片。

3.8 神经网络在机器人控制中的应用

上一节介绍了基于神经网络的系统建模与控制的一般方法。原理上,神经网络可以应用于控制的各个方面。但神经网络应用于机器人控制是研究得最多也是成果取得最多的一个方面。所以本节着重讨论这方面的问题。

广义上讲,机器人控制主要包括以下三方面内容:任务规划、路径规划和运动控制。任务规划处理有关作业的信息,它根据作业的要求规划出较具体的子任务或动作序列。路径规划则是在给定起点、终点、中间点以及某些必要的限制条件下,规划出机械手末端所要经过的路径点的序列。这些限制条件可能包括速度和加速度的限制或者避碰的要求等。运动控制则是根据给定的路径点及机器人的运动学和动力学特性,求出适当的关节力矩来产生所需要的运动。对于以上三个方面的问题,原则上均可以用神经网络来加以实现,且确实已有不少人在这方面进行研究工作。而目前研究得较多的则主要在后两个方面。即路径规划和运动控制。所以本节也只讨论这两方面的问题。

3.8.1 神经网络运动学控制

机器人控制之所以比较困难,其主要原因在于要求的运动轨迹是在直角坐标空间中给定的,而实际的运动却是通过安装在关节上的驱动部件来实现,因而需要将机械手末端在直角坐标空间的运动变换到关节的运动,也就是需要进行逆运动学的计算。这个计算取决于机器人的手臂参数及所使用的算法。但是我们知道,对于具有四肢的动物(包括人),他们运动时很自然地便完成了从目标空间到驱动器(肌肉)坐标的转换。这个变换一方面是在基因中先天编好的。另一方面它又是通过后天地学习来不断地加以完美的。该生物系统的运动控制为机器人的神经网络控制提供了很好的参考模型。这种控制不需要各个变量之间的准确的解析关系模型,而只要通过对大量例子的训练即可实现。

1. 基于特征网络的运动学控制

在基于运动学的机器人控制方法中,分解运动速度的方法是比较典型的一种。它是一种在直角坐标空间(相对于关节坐标空间)进行闭环的控制方法。对于那些需要准确轨迹跟踪的任务(如弧焊等),必须采用这样的控制方法。分解运动速度控制的系统结构如图 3.52 所示。

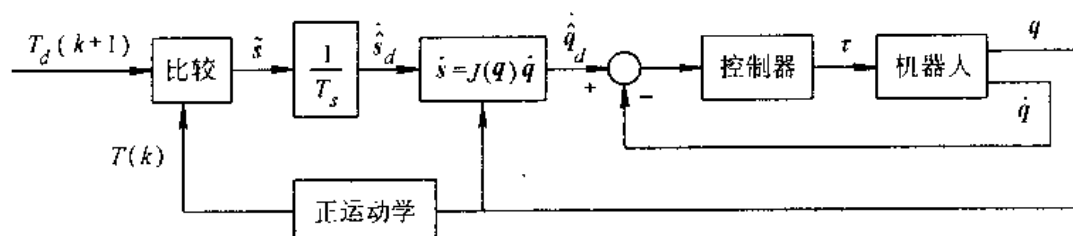


图 3.52 分解运动速度控制的系统结构图

可以看出,其中最关键的部分是速度逆运动学计算,即计算

$$\dot{q} = J^{-1}(q)\dot{s}$$

因此,该控制方法有时也称为逆雅可比控制。此式的计算不仅需要有效的雅可比矩阵求逆算法,而且需要知道机器人的运动学参数。如果采用神经网络,则可不必知道这些参数。因此它可作为求解速度逆运动学的另外一种颇具吸引力的方法。

对于上式的非线性关系若采用通常的神经网络,则该网络的输入为 \dot{s} 和 q , 输出为 \dot{q} 。显然这是一个 $2n$ 到 n 的映射。对于 6 个自由度的机器人,则具有 12 个输入和 6 个输出。对于这样一个多输入多输出且高度非线性的映射,必须要有足够多的训练样本才能学习到所需要的非线性映射关系。例如,若每个输入变量选取 10 个值,则总共就需取 10^{12} 个样本。对这么多的样本进行训练是不可想象的。而取过份少的样本则可能学不到所需要的非线性映射关系。为此可采用如图 3.53 所示的两个网络。其中上面的网络称为功能网络,它实现如上式所示的计算。这是一个线性网络,其输出结点无偏置项,且激发函数即为 $f(x)=x$ 。该网络共有 n^2 个连接权,它恰好对应于 $J^{-1}(q)$ 的 n^2 个元素。下面的网络称为特征网络。它由 n^2 个解耦的子网络所组成,每个子网络有 n 个输入 1 个输出。也就是说,每个子网络负责学习矩阵 $J^{-1}(q)$ 中的一个元素。 n 个输入量对于每个子网络都是相同的。每个子网络均有两个隐层。隐层和输出层结点均选择 S 形激发函数。这样就将一个 n 到 n^2 的映射问题简化为 n^2 个 n 到 1 的函数映射,从而大大简化了学习问题。而且这 n^2 个子网络可以并列地学习。因此增加自由度是增加要学习的函数的数目。

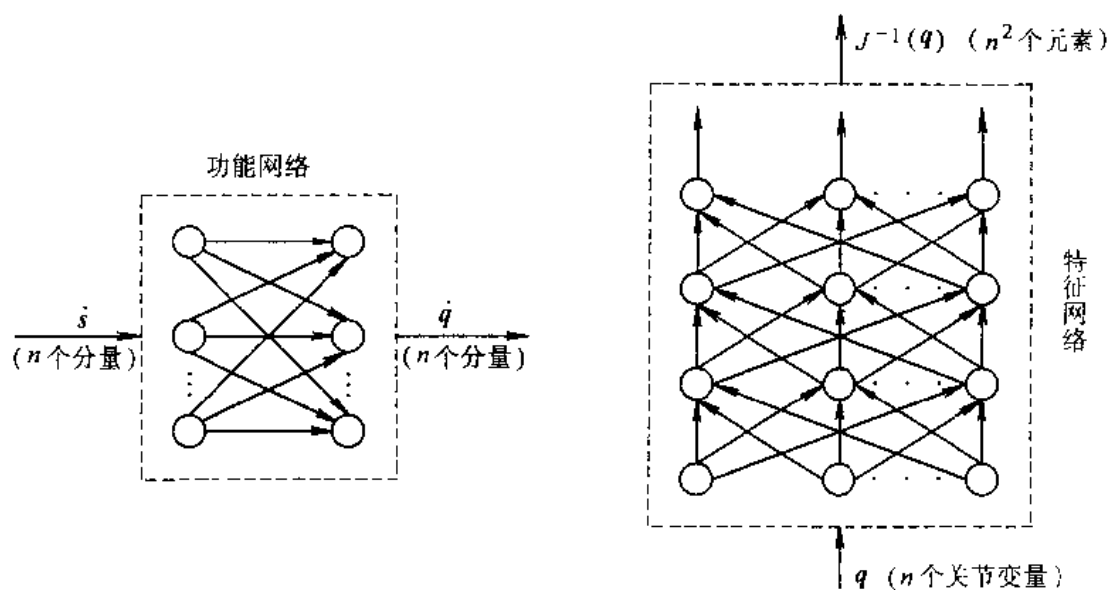


图 3.53 用于机器人逆运动学的特征网络

图 3.54 显示了采用如图 3.53 所示的特征网络和采用 \dot{s} 和 q 作为输入、 \dot{q} 作为输出的标准前馈网络这两种情况下进行学习的仿真结果。机械手采用 PUMA560 的 3 个自由度模型,共采用 400 个随机产生的样本。特征网络,各层的结点数为 3—25—15—9,学习率和动量项因子分别取 0.0003 和 0.9。为了互相比,标准网络的各层的结点数为 6—25—15—3,即两个隐层的结点数是相同的,学习率和动量项因子与上面相同。

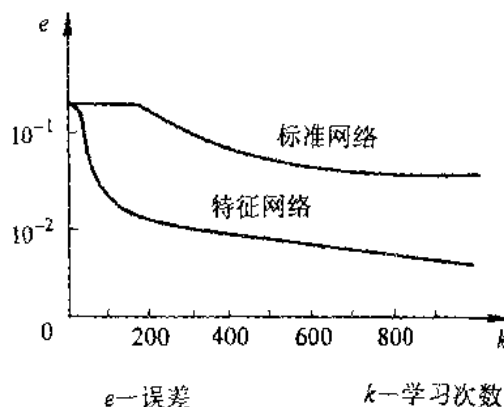


图 3.54 三自由度机械手的学习曲线

由图看出,基于特征网络的学习性能明显优于标准网络。在经过 1000 次学习后,标准网络和基于特征网络的均方根误差分别为 0.0303 和 0.0036,在头 200 次的学习过程中,基于特征网络的学习性能的改善尤为明显。

图 3.55 表示了利用神经网络进行机器人运动学控制的系统结构,它与图 3.52 所示的分解运动速度控制具有完全相同的结构,只不过这时的速度逆运动学计算由神经网络来实现。

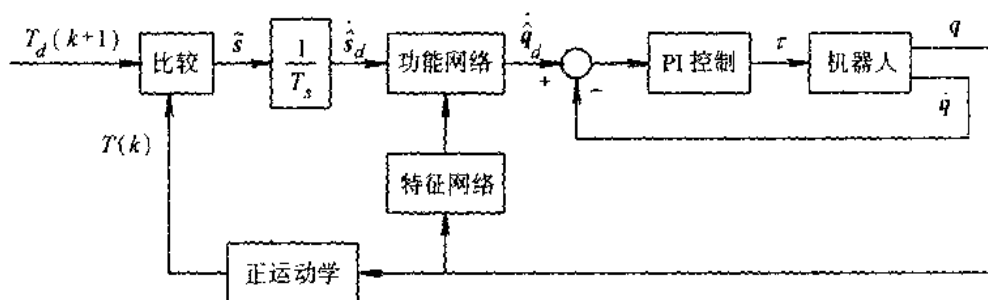


图 3.55 基于特征网络的运动学控制

2. 基于双向映射神经网络的运动学控制

通常的机器人运动学控制主要是基于正、逆运动学的计算。它不仅计算麻烦而且需要经常进行校准才能保持精度。尤其是对于冗余机器人,由于其逆运动学求解的困难,通常的机器人运动学控制方法更显困难。因此可考虑采用神经网络来自动地实现正逆运动学的计算,即使对于冗余机器人,这种方法也是适用的。下面介绍一种双向映射神经网络,该网络主要由一个多层前馈网组成,其隐层为正弦激发函数。从网络的输出到输入有一个反馈连接,形成循环回路,正向网络严格地实现正运动学方程。反馈连接的作用是修改网络的输入(关节变量)以使用网络输出(末端位姿)朝着期望的位姿点运动。这种双向映射网具有如下一系列优点:a. 提供精确的正、逆运动学计算;b. 只需要简单的训练;c. 能够处理冗余机器人逆运动学的多解问题。d. 由于采用基于李雅普诺夫函数方法来控制末

端轨迹朝着期望的位姿点运动,因而它也可直接用于轨迹的生成。

(1) 正运动学模型

设机械手由 n 个旋转关节组成,则其正运动学必为这些关节角的三角函数的组合。一般情况下可写成,

$$\bar{s}_k(\theta) = \sum_{i=1}^m l_i^k \sin[(w_i^k)^T \theta] \quad k = 1, 2, \dots, l$$

其中

$$\theta = [\frac{\pi}{2} \theta_1 \dots \theta_n]^T \quad w_i^k = [w_{i0}^k w_{i1}^k \dots w_{im}^k]^T \quad w_{ij} \in [-1, 0, 1]$$

\bar{s}_k 表示末端位姿的第 k 个分量。

上面式子的函数关系可用图 3.56 所示的多层前馈网来实现,其中隐层取正弦函数为激发函数,它可实现上面式子的准确关系。通常的 S 形激发函数只能实现函数的近似。

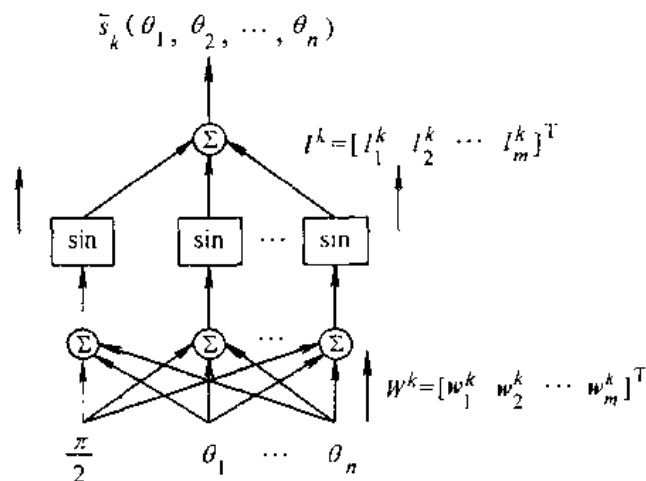


图 3.56 机械手的正运动学模型

为了实现上面的关系,正弦隐层结点的最大个数为 3^n 。当机械手的某些关节为滑动关节时,相应的 θ_i 取为常数,而将某些 l_i^k 处理为变量。

根据机械手的具体结构, w_i^k 可以预先设置。剩下的问题便是估计参数 l_i^k 。它可用如下的最小方差算法来计算。

$$l_i^k(j+1) = l_i^k(j) + \eta (s_k^d - \bar{s}_k(\theta^d)) \sin[(w_i^k)^T \theta^d]$$

其中 \bar{s}_k^d 表示末端位姿期望值的第 k 个分量, θ^d 表示期望的关节变量, η 是学习率。

采用上述方法(用足够多的结点数且预先给定 w_i^k)学习正运动学的好处是可以获得精确的正运动学解,且收敛速度很快。其缺点是可能需要太多的结点,且可能有很多为零,因此可以考虑用较少的结点,而使 l_i^k 和 w_i^k 均通过 BP 学习算法或它的变形来加以确定。采用这种简化网络的缺点是训练时间长且不能获得精确解,而且有时还可能陷于局部的极值。为此可采用一种递阶的自组织学习算法,它通过监控网络的性能,自动地增补隐层的正结点。

(2) 基于李雅普诺夫函数的逆运动学求解

前面已介绍了通过计算正运动学来迭代求解逆运动学的方法,其基本思路如图 3.57 所示。若迭代算法是收敛的(用控制系统的术语,即系统是稳定的),则最终有 $\tilde{s} \rightarrow 0$, 即 $\tilde{s} = \tilde{s}^d$, 从而 θ 便为所求逆运动学解。

取李雅普诺夫函数

$$V = \frac{1}{2} \tilde{s}^T \tilde{s} + \frac{1}{2} \tilde{\theta}^T \tilde{\theta}$$

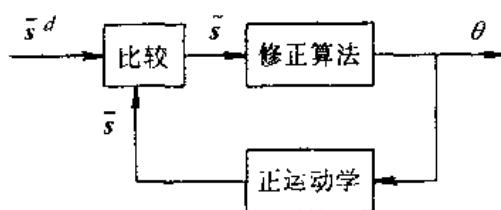


图 3.57 迭代求解逆运动学的基本思路

其中 \tilde{s} 表示 \tilde{s}^d 与 \tilde{s} 之间的误差向量, \tilde{s}^d 是期望的位姿, \tilde{s} 是实际的位姿。 $\tilde{\theta} = \theta^d - \theta$, θ 是实际关节向量, θ^d 是关节空间的限制向量。

对上式求导得

$$\dot{V} = \left(\frac{\partial V}{\partial \theta} \right)^T \dot{\theta} = - \left(\tilde{s}^T \frac{\partial \tilde{s}}{\partial \theta} + \tilde{\theta}^T \right) \dot{\theta} = - (\tilde{s}^T \bar{J} + \tilde{\theta}^T) \dot{\theta}$$

其中 \bar{J} 为雅可比矩阵,若取

$$\dot{\theta} = \frac{1}{2} \frac{\|\tilde{s}\|^2}{\|\bar{J}^T \tilde{s} + \tilde{\theta}\|^2} (\bar{J}^T \tilde{s} + \tilde{\theta})$$

代入前面式子得

$$\dot{V} = - \frac{1}{2} \|\tilde{s}\|^2$$

由于 $\dot{V} < 0$, 所以系统是渐近稳定的, 并最终达到 $\tilde{s} \rightarrow 0$ 。从而实现了所要求的逆运动学计算。上面计算 $\dot{\theta}$ 式子便是所要求的修正算法。该式中需要用到 \bar{J} , 它也可以用图 3.58 所示的神经网络来实现, 其方法类似于正运动学的计算。且它可以利用正运动学计算网络的部分结果。如图中 W^k 与前面是相同的。

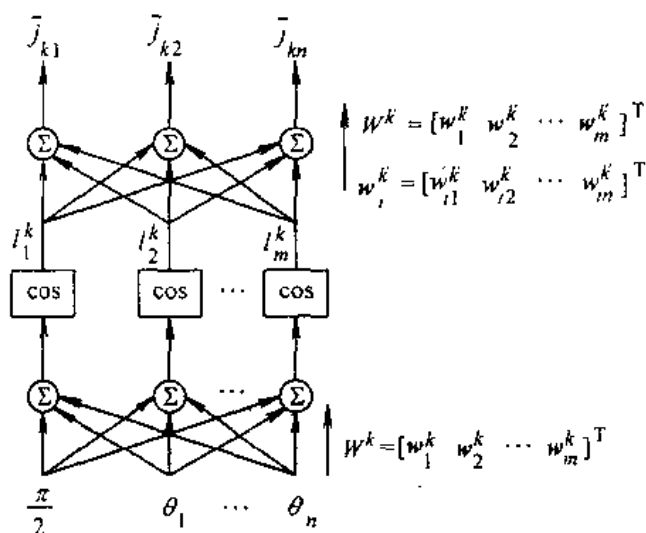


图 3.58 计算雅可比矩阵的神经网络

图 3.59 表示利用双向映射神经网络计算逆运动学的总的结构图。在具体计算时还必须注意以下几个具体问题。

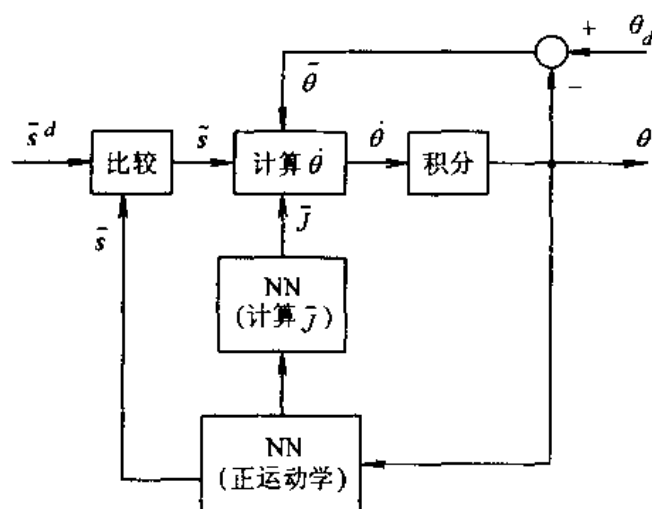


图 3.59 利用双向映射神经网络计算逆运动学

1) 当 $\tilde{s} \neq 0$ 而 $\tilde{s}^T J + \tilde{\theta}^T = 0$ 时, V 到达它的一个局部极值点, 此时 $\|\dot{\theta}\| \rightarrow \infty$ 。在该点 θ 将出现一个跳变, 它对跳出局部极值点是有益处的。但是要采取适当的限制措施, 避免使 θ 越出工作空间。

2) 当 \tilde{s} 接近 0 时, 由于在 \dot{V} 的表达式中第二项起主要作用, 从而使 θ 不是直接朝使 $\tilde{s} = 0$ 的方向移动。为克服此问题, 可在李雅普诺夫函数中加入拉格朗日乘子:

$$V(\lambda, \theta) = \frac{\lambda}{2} \tilde{s}^T \tilde{s} + \frac{1}{2} \tilde{\theta}^T \tilde{\theta}$$

当 \tilde{s} 越接近 0, λ 便越变大, 为了实现这一点, 可取

$$\dot{\theta} = \frac{\lambda \left(\frac{1}{2} \|\tilde{s}\|^2 \right)^\alpha + \frac{1}{2} \|\tilde{s}\| \|\tilde{\theta}\|}{\|\lambda \tilde{J}^T \tilde{s} + \tilde{\theta}\|^2} (\lambda \tilde{J}^T \tilde{s} + \tilde{\theta})$$

$$\dot{\lambda} = \frac{\|\tilde{\theta}\|}{\|\tilde{s}\|}$$

其中 α 用来控制收敛性, 通常取 $\frac{1}{2} < \alpha < 1$ 。

3) 关节空间中的限制向量 θ^l 需要根据实际情况来具体加以选择。若无特殊限制要求, 则可选 $\theta^l = \theta$, 即 $\tilde{\theta} = 0$ 。也就是在 V 中不考虑这一项; 若要求各关节转过的角度尽量小, 则可选 $\theta^l = \theta^0$, θ^0 为起始关节角位置; 若要求运动轨迹尽量不越出工作空间, 则可选 $\theta^l = \theta^m$, θ^m 表示关节变量容许的限制范围的中心值。

3.8.2 神经网络动力学控制

在机器人动力学控制方法中, 较典型的是计算力矩控制和分解运动加速度控制。前者

在关节空间闭环,后者在直角坐标空间闭环。画出分解运动加速度控制的系统结构如图 3.60 所示。

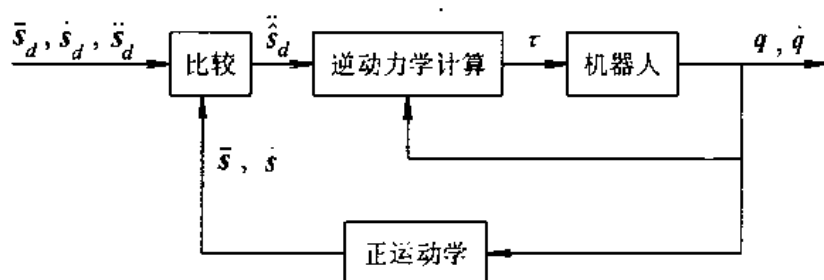


图 3.60 分解运动的加速度控制

在图 3.60 中

$$\ddot{s} = \ddot{s}_d + K_1(\dot{s}_d - \dot{s}) + K_2(\bar{s}_d - \bar{s})$$

其中 \$\bar{s}\$ 表示 \$\bar{s}_d\$ 与 \$\bar{s}\$ 之间的误差向量。

若机器人的动力学模型表示为

$$H(q)\ddot{q} + h(q, \dot{q}) + G(q) = \tau$$

其中 \$H(q)\$ 表示机器人的惯性矩阵, \$h(q, \dot{q})\$ 表示离心力和哥氏力项, \$G(q)\$ 表示重力项, \$\tau\$ 为关节力矩向量, 则

$$\tau = H(q)J^{-1}(q)[\ddot{s} - \dot{J}(q, \dot{q})\dot{q}] + h(q, \dot{q}) + G(q)$$

可得整个系统的误差方程为

$$\ddot{s} + K_1\dot{s} + K_2\bar{s} = 0$$

可见, 只要 \$K_1\$ 和 \$K_2\$ 为正定对角阵, 整个系统是完全解耦且渐近稳定的。

在上面的控制结构中, 关键是逆动力学计算。这里主要有两方面的问题: 一是计算工作量很大, 难以满足实时控制的要求; 二是需要知道机器人的运动学和动力学参数。要获得这些参数, 尤其是动力学参数往往是很困难的。采用神经网络来实现逆动力学的计算, 原则上可以克服上述两个问题。由于神经网络的并行计算的特点, 它完全满足实时性的要求, 同时它是通过输入输出的数据样本经过学习而实现动力学的非线性关系, 因而它并不依赖机器人的参数。

具体用神经网络来实现也有其自身的问题。我们知道, 当神经网络所实现的函数非常复杂且非线性很强时, 学习所需的时间也将非常长。机器人逆动力学即属于这样的情况。它的输入是 \$\ddot{s}_d, \dot{s}_d, \bar{s}_d, q, \dot{q}\$, 输出是 \$\tau\$。对于 6 自由度机器人, 共有 18 个输入和 6 个输出。若要覆盖整个工作空间, 其训练数据样本的个数将是大得惊人的数字, 实际上是不可能实现的。在前面讨论逆运动学时, 我们采用特征网络和功能网络相分离的方法来减小训练样本的个数。这里也可采用类似的方法, 即将整个系统分解为多个子系统, 分别对每个子系统进行学习, 可以使问题得以简化。针对上面的式子具体分解的方法如下:

$$s_1(q, \dot{q}) = \dot{J}(q, \dot{q})\dot{q}$$

$$\begin{aligned}\ddot{s}_2 &= \ddot{\hat{s}}_d - \dot{s}_1 \\ \tau_1 &= H(q)J^{-1}(q)\ddot{s}_2 \\ \tau_2 &= h(q, \dot{q}) \\ \tau_3 &= G(q) \\ \tau &= \tau_1 + \tau_2 + \tau_3\end{aligned}$$

根据上述分解,可以画出用神经网络实现的控制结构如图 3.61 所示。

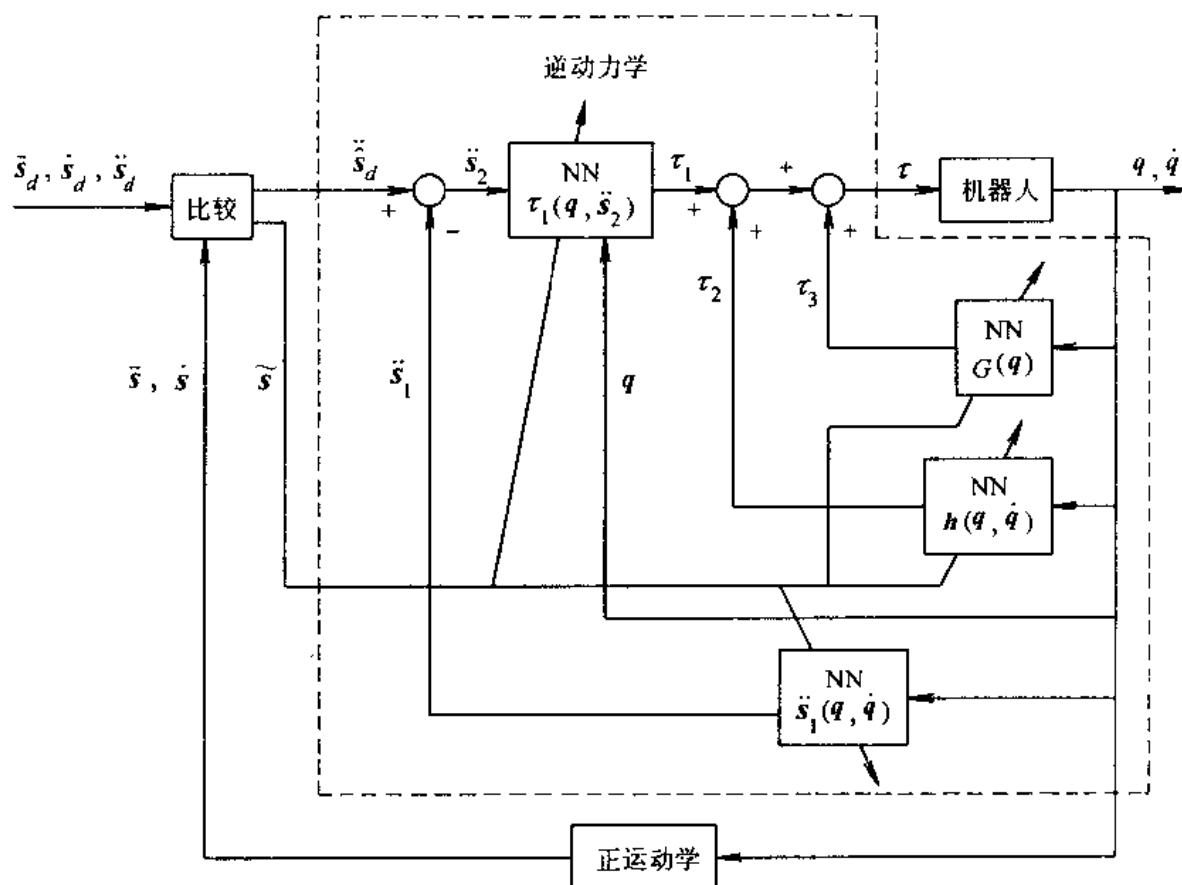


图 3.61 基于功能分解的神经网络控制

图 3.61 中共有 4 个神经子网络(若正运动学也用神经网络实现,则共 5 个),这时每个神经网络的输入维数减小了,而且结构也简单了。因此学习比较容易。图 3.61 是一种在线学习的结构,适合于实时控制。但是初始连接权系数的选择很关键,如果距离太远,可能很难收敛到要求的值。因此实际上通常先采用图 3.62 所示的结构进行离线学习,然后再用到图 3.61 的结构。

以上讨论的均为直角坐标空间闭环情况。由于综合考虑了整个系统,从而产生了输入输出维数太高,学习困难的问题。若考虑关节空间闭环,并忽略关节间的耦合作用,那末输入输出的维数将大为减小,学习也将容易得多。图 3.63 表示了独立关节神经网络自适应控制的结构。

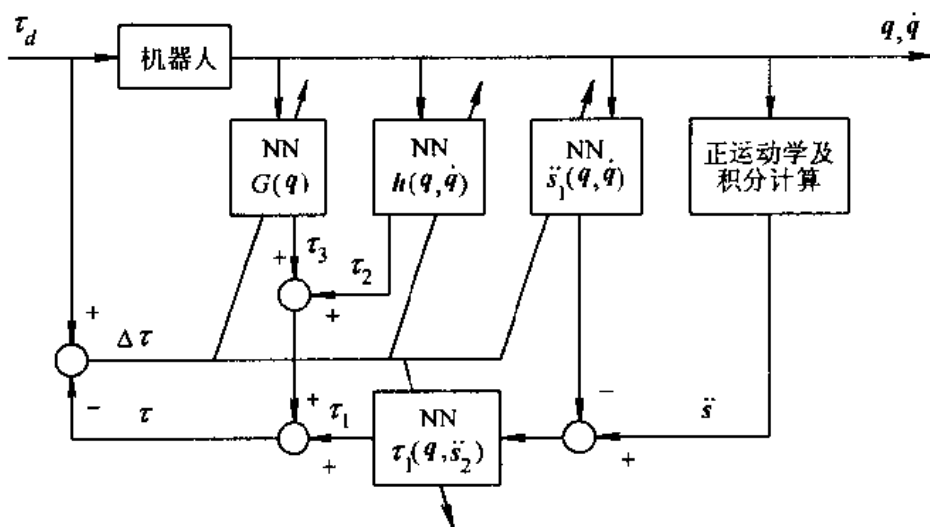


图 3.62 基于功能分解的神经网络离线学习结构

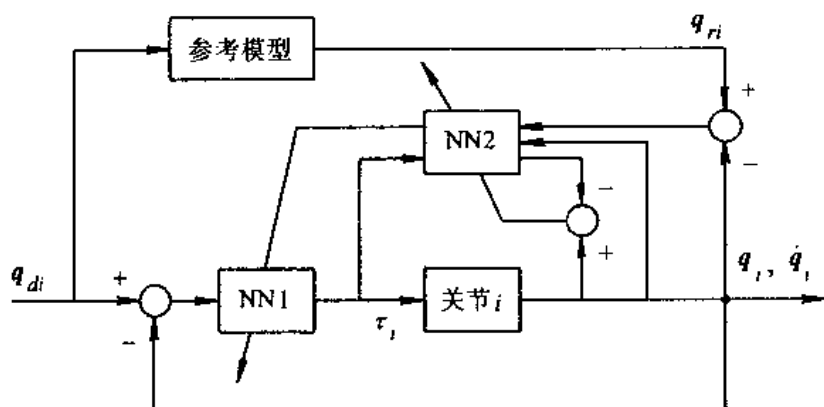


图 3.63 独立关节神经网络自适应控制

图 3.63 所表示的只是其中一个关节的控制结构,对于 6 个关节的机械手,则共有 6 个这样的控制结构。每个关节的控制器由两个神经网络组成。

这里 NN1 起控制器的作用,NN2 主要是为了得到一个正向的动力学模型,以使得 $\Delta q_n = q_n - q_i$ 和 $\Delta \dot{q}_n = \dot{q}_n - \dot{q}_i$ 能反向传播得到 $\Delta \tau_i$ 。这里 NN2 共有三个输入: τ_i , q_i 和 \dot{q}_i , 输出为时间差分 Δq_i 和 $\Delta \dot{q}_i$ 。

以上给出了神经网络动力学控制的两种结构。图 3.61 是基于功能分解的方法用多个子神经网络实现了机器人的逆动力学,并按照分解运动加速度的控制结构实现了机器人的神经网络动力学控制。该控制结构的优点是考虑了整个机器人的动力学特性,同时通过功能分解使计算复杂性得到一定程度的简化。主要的问题是难于根据所能测量的量来调整这些子神经网络连接权。图 3.63 是一种独立关节的神经网络自适应控制结构。它的主要优点是结构简单,实现容易,主要问题是未考虑关节间的耦合动力学特性。以上两种结构还有一个共同的特点,即神经网络担负了全部的控制器任务,这就对神经网络初始权值

的设置提出了较高的要求,所设置的初始权值必须确保系统能够稳定地运行,然后在运行过程中再不断地加以调整和完善。也就是说这种控制结构对于神经网络的离线训练提出了更高的要求。

针对上面所提到的问题,下面给出一种将常规控制与神经网络控制相结合的结构。设机器人的动力学模型仍为

$$H(q)\ddot{q} + h(q, \dot{q}) + G(q) = \tau$$

所采用的控制规律为

$$\tau = K_p \tilde{q} + K_d \dot{\tilde{q}} + W\alpha(q_d, \dot{q}_d, \ddot{q}_d) + \epsilon_m \operatorname{sgn}(\dot{\tilde{q}} + c\tilde{q})$$

$$\dot{W} = \Gamma[(\dot{\tilde{q}} + c\tilde{q})^T \alpha(q_d, \dot{q}_d, \ddot{q}_d)]$$

其中 $\tilde{q} = q_d - q$, $\dot{\tilde{q}} = \dot{q}_d - \dot{q}$, K_p, K_d 及 Γ 均为对称正定阵(通常取正定对角阵)。

可以看出,上面的控制规律由三部分所组成。即

$$\tau = \tau_{fb} + \tau_{ff} + \tau_{sm}$$

其中

$$\tau_{fb} = K_p \tilde{q} + K_d \dot{\tilde{q}}$$

是常规的 PD 反馈控制

$$\tau_{ff} = W\alpha(q_d, \dot{q}_d, \ddot{q}_d)$$

是神经网络控制,它实现如下的逆动力学特性

$$W\alpha(q_d, \dot{q}_d, \ddot{q}_d) = H(q_d)\ddot{q}_d + h(q_d, \dot{q}_d) + G(q_d)$$

可见这一项是由神经网络实现的前馈控制。

$$\tau_{sm} = \epsilon_m \operatorname{sgn}(\dot{\tilde{q}} + c\tilde{q})$$

是滑动模态控制,它主要用来增强系统的鲁棒性。其中 ϵ_m 取神经网络拟合误差的上界值。

可以证明,当 K_p 和 K_d 是充分大的正定矩阵、 c 是充分小的正常数时,该控制规律可确保闭环系统是渐近稳定的。即

$$\lim_{t \rightarrow \infty} \tilde{q} = 0, \quad \lim_{t \rightarrow \infty} \dot{\tilde{q}} = 0$$

画出该神经网络控制的系统结构如图 3.64 所示。

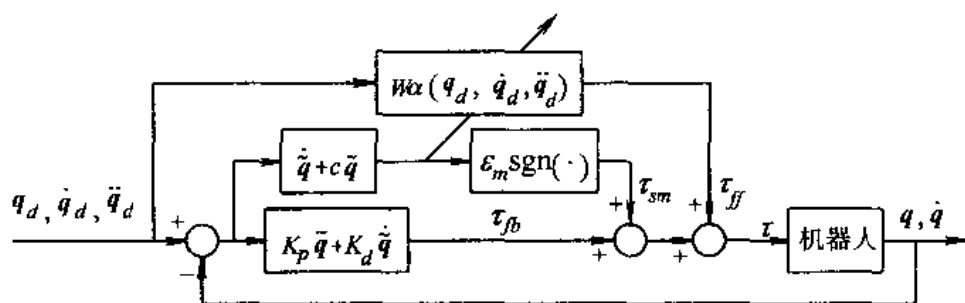


图 3.64 神经网络前馈控制与常规的反馈控制相结合的机器人控制系统结构

在该控制结构中的神经网络可用前面介绍过的任何一种局部逼近网络,即 CMAC、B 样条、RBF 或模糊神经网络等。该神经网络的学习也可分为两种情况。第一种情况是在接

入控制系统之前需进行离线的学习和训练。这时首先需取得在实际工作轨迹附近的一组训练数据。这些数据可通过实验或仿真得到,然后根据这些数据样本对其进行离线学习,其学习算法为

$$w_{ij}(k+1) = w_{ij}(k) + \beta[\tau_i^d - (\tau_{ff})_i]\alpha_j/\alpha^T\alpha$$

其中 $i=1,2,\dots,r$, r 为机器人的关节数; $j=1,2,\dots,m$, m 为基函数(或称感受域函数)的个数; τ_i^d 是第 i 个关节的期望力矩, $(\tau_{ff})_i$ 是神经网络的第 i 个输出; α_j 是第 j 个基函数,它是网络输入量 q_d, \dot{q}_d 和 \ddot{q}_d 的函数; β 为学习率,为了确保学习算法的收敛性,可取 $0 < \beta < 2$, 实际上通常取 $0 < \beta < 1$ 。

第二种情况是该神经网络接入控制系统后的在线学习,其学习算法如前面的式子所示。实际计算时可采用如下的离散学习算法

$$w_{ij}(k+1) = w_{ij}(k) + \gamma(\dot{\tilde{q}}_i + c\tilde{q}_i)\alpha_j, \quad \gamma > 0$$

3.8.3 神经网络路径规划

路径规划问题是要找到一条从起始位置到目标位置的无碰撞路径,通常解决这个问题的方法是首先构造一个数据结构来表示工作空间;然后通过对这个数据结构的搜索寻找到一条无碰撞的路径。已提出的方法主要有位形空间法、图搜索法、拓扑法等。但是这些方法通常存在组合爆炸的问题,而且从二维寻优问题扩展到三维寻优问题也存在很大的困难。

这里我们介绍一种神经网络,它是并列连接网络结构。它可以实时地进行无碰撞路径规划。该网络对一系列的路径点进行规划,其目标是使得整个路径的长度尽量短,同时又要尽可能远离障碍物。从数学的观点,它等效于优化一个代价函数,该代价函数由路径长度和碰撞罚函数两部份组成。之所以适合于并列计算,主要因为:第一障碍物是用连接模型来表示的;第二单个路径点运动的计算只需用到局部的信息。该网络可以处理将物体视为一个质点的情况,同时也可将其处理为能平衡和旋转的三维实体。最后,该网络结合模拟退火算法可以解决局部极值的问题。

1. 无碰撞路径的表示

如图 3.65 所示,无碰撞路径可用一系列中间点来表示,相邻点之间用线段相连。这样的表示有如下好处:(1) 可通过指定足够多的点来达到任意的精度;(2) 可将原始问题分解为一组统一的规模较小的任务。在这些小任务中,问题仅仅变为要关心一个点与障碍物的关系;(3) 由于将路径规划问题局限为一系列路径点,从而便于实现大量的并列和分布计算。

为了对路径与障碍物之间的碰撞性质加以量化,一条路径的碰撞罚函数定义为各路径点的碰撞罚函数之和,而一个点的碰撞罚函数是通过它对各个障碍物的神经网络表示得到的。其基本想法是,障碍物均假设为多面体,它可用一组线性不等式来表示,于是在障碍物中的点必定满足所有不等式的限制。

图 3.66 表示了一个这样的神经网络。底层的三个结点分别表示给定路径点的坐标 x 、 y 和 z ,中间层的每个结点相应于障碍物的一个不等式限制条件;底层和中间层的连接权系数就等于不等式中 x 、 y 、 z 前面的系数,中间层每个结点的阈值等于相应不等式中的

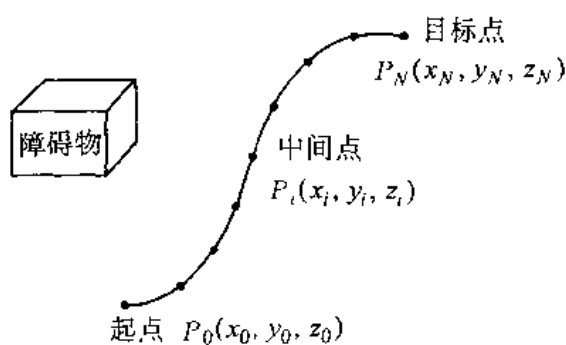


图 3.65 用一系列路径点来表示的路径

常数项。中间层到顶层的连接权均为 1, 顶层结点的阈值取为不等式的个数减去 0.5 后的负数。

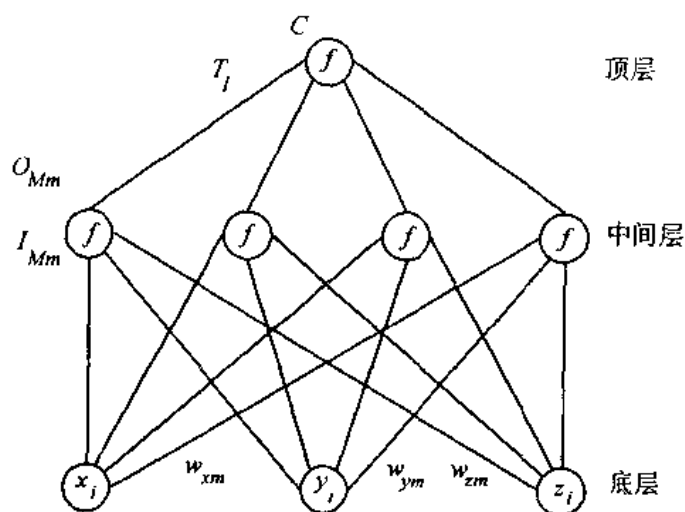


图 3.66 计算到一个障碍物的罚函数的神经网络

该连续网络的运算关系为

$$\begin{aligned}
 C &= f(T_I) \\
 T_I &= \sum_{m=1}^M O_{Mm} + \theta_T \\
 O_{Mm} &= f(I_{Mm}) \\
 I_{Mm} &= w_{xm}x_i + w_{yi}y_i + w_{zm}z_i + \theta_{Mm}
 \end{aligned}$$

其中各符号的含义为

C : 顶层结点输出

T_I : 顶层结点输入

θ_T : 顶层结点阈值

O_{Mm} : 中间层第 m 个结点的输出

I_{Mm} : 中间层第 m 个结点的输入

θ_{Mm} : 中间层第 m 个结点的阈值

w_{xm}, w_{ym}, w_{zm} : 第 m 个不等式限制条件的系数

当空间的一个点的坐标 (x, y, z) 从底层输入时, 且设中间层和顶层结点的激发函数为阶跃函数, 则中间层的每个结点便决定该点是否满足它的限制条件, 若满足, 输出为 1, 否则输出为 0。若所有中间点均满足, 则顶层输出为 1, 它表示该点在障碍物内。若中间点检测出其中至少有一个不满足限制条件, 顶层输出便为 0, 它表示该点在障碍物外。激发函数采用阶跃函数可检测空间点是否与障碍物相碰, 但不能检测出它距离障碍物的远近程度。因此, 通常取激发函数为常用的 S 形函数, 即

$$f(x) = \frac{1}{1 + e^{-x/T}}$$

这样顶层输出将在 0 到 1 之间连续变化, 它反映了空间给定点与障碍物的碰撞程度, 输出数越大, 说明该点越接近障碍物的中心; 数越小, 说明该点越远离障碍物。

图 3.67 给出了一具体例子。其中图(a)表示了一个矩形的障碍物; 图(b)是描述该障碍物的 4 个不等式限制条件; 图(c)表示相应的神经网络, 连接线旁的数字表示相应的连接权系数, 圆圈中的数字表示相应结点的阈值; 图(d)表示该障碍物的碰撞罚函数的三维图形, 其中取 $T=0.05$ 。通过改变 T , 可用来调整罚函数的形状, 它对改善寻优算法的性质有很重要的作用, 这一点下面还要讨论。

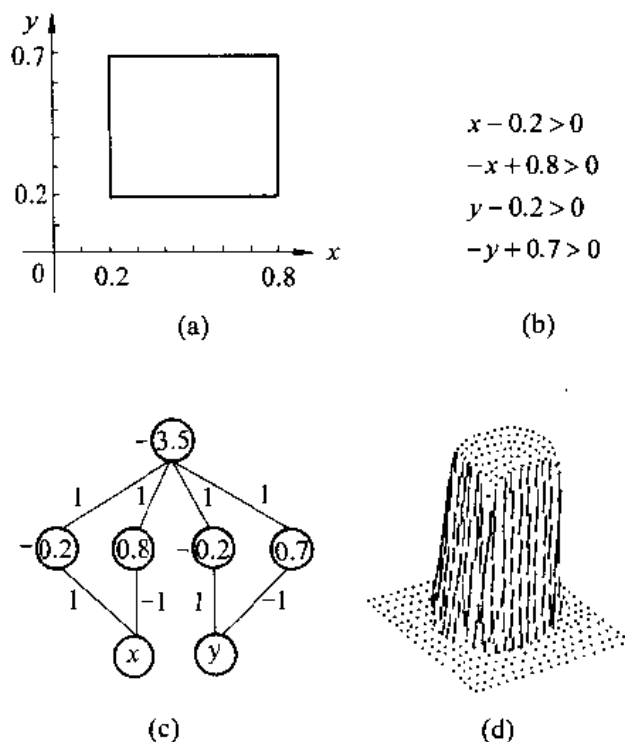


图 3.67 矩形障碍物的神经网络举例

2. 物体表示为质点时的路径规划

无碰路径规划问题可以等效为具有两个约束条件的优化问题。一个是避免物体与障

碍物相碰撞,另一个是要求规划的路径尽量短。基于上述关于障碍物的网络表示,这两个约束条件可以加以量化。这样,路径规划问题便可变为一个极值问题或优化问题。其优化的能量函数由路径的长度及障碍物的罚函数两部分组成。

如果物体用一个点表示,则路径的碰撞罚函数定义为所有路径点的碰撞函数之和。每个点的碰撞罚函数可通过相应的连结网络来计算。相应于碰撞罚函数的这部分能量可表示为

$$E_c = \sum_{i=1}^N \sum_{k=1}^K C_i^k$$

其中 K 是障碍物的个数, N 是路径点的个数, C_i^k 表示第 i 个路径点 P_i 对第 k 个障碍物的碰撞罚函数。

图 3.68 表示了计算 E_c 的神经网络结构。可以看出,该网络的计算对于每个路径点和每个障碍物都是并列的,即每个 C_i^k 可以同时计算。

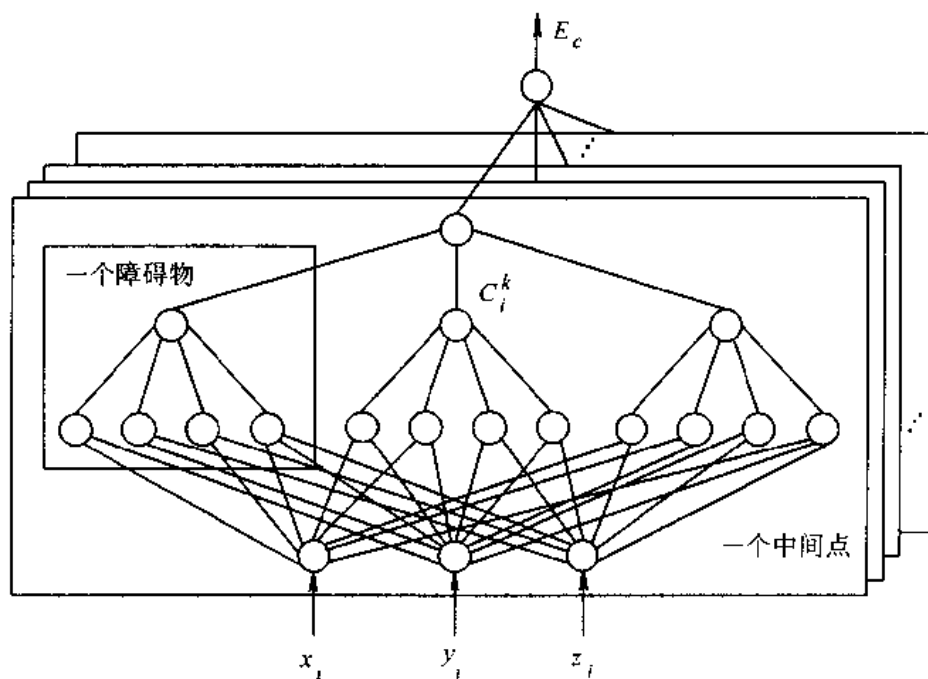


图 3.68 碰撞罚函数联结网络结构

相应于路径长度这部分能量定义为所有线段长度的平方之和。即,对于所有路径点 $P_i(x, y, z), i=1, 2, \dots, N$, 定义

$$E_l = \sum_{i=1}^{N-1} L_i^2 = \sum_{i=1}^{N-1} [(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2]$$

其中 L_i 表示第 i 个线段的长度。 E_l 反映了整个路径的长度。

整条路径的总能量定义为

$$E = w_l E_l + w_c E_c$$

其中 w_l 和 w_c 分别表示对每一部分的加权。

使整个能量 E 极小意味着该路径的长度较短,并较少可能与障碍物相碰撞,这正是我们所希望的目标。由于整个能量是各个路径点函数,因此通过移动每个路径点,使其朝着能量减小的方向运动,最终便能获得总能量最小的路径。求 E 对时间的导数得

$$\begin{aligned}\dot{E} &= \sum_i (\nabla_{P_i} E)^T \dot{P}_i \\ &= \sum_i \left\{ \left[w_i \left(\frac{\partial L_i^2}{\partial x_i} + \frac{\partial L_{i-1}^2}{\partial x_i} \right) + w_c \sum_k \frac{\partial C_i^k}{\partial x_i} \right] \dot{x}_i + \left[w_i \left(\frac{\partial L_i^2}{\partial y_i} + \frac{\partial L_{i-1}^2}{\partial y_i} \right) + w_c \sum_k \frac{\partial C_i^k}{\partial y_i} \right] \dot{y}_i \right. \\ &\quad \left. + \left[w_i \left(\frac{\partial L_i^2}{\partial z_i} + \frac{\partial L_{i-1}^2}{\partial z_i} \right) + w_c \sum_k \frac{\partial C_i^k}{\partial z_i} \right] \dot{z}_i \right\}\end{aligned}$$

若取

$$\begin{aligned}\dot{x}_i &= -\eta \left[w_i \left(\frac{\partial L_i^2}{\partial x_i} + \frac{\partial L_{i-1}^2}{\partial x_i} \right) + w_c \sum_k \frac{\partial C_i^k}{\partial x_i} \right] \\ \dot{y}_i &= -\eta \left[w_i \left(\frac{\partial L_i^2}{\partial y_i} + \frac{\partial L_{i-1}^2}{\partial y_i} \right) + w_c \sum_k \frac{\partial C_i^k}{\partial y_i} \right] \\ \dot{z}_i &= -\eta \left[w_i \left(\frac{\partial L_i^2}{\partial z_i} + \frac{\partial L_{i-1}^2}{\partial z_i} \right) + w_c \sum_k \frac{\partial C_i^k}{\partial z_i} \right]\end{aligned}$$

其中 η 取为正数,则

$$\dot{E} = -\frac{1}{\eta} \sum_i (\dot{x}_i^2 + \dot{y}_i^2 + \dot{z}_i^2) < 0$$

可见 E 将逐渐减小,直至 $\dot{x}_i=0, \dot{y}_i=0$ 和 $\dot{z}_i=0$ 时才有 $\dot{E}=0$,这时 E 取得最小值。所得结果即为要求的路径。

根据上面的式子有

$$\begin{aligned}\frac{\partial L_i^2}{\partial x_i} + \frac{\partial L_{i-1}^2}{\partial x_i} &= -2x_{i+1} + 4x_i - 2x_{i-1} \\ \frac{\partial C_i^k}{\partial x_i} &= \left(\frac{\partial C_i^k}{\partial (T_I)_i^k} \right) \left(\frac{\partial (T_I)_i^k}{\partial x_i} \right) = \left(\frac{\partial C_i^k}{\partial (T_I)_i^k} \right) \sum_{m=1}^M \left(\frac{\partial (O_{Mm})_i^k}{\partial (I_{Mm})_i^k} \right) \left(\frac{\partial (I_{Mm})_i^k}{\partial x_i} \right) \\ &\quad - f'[(T_I)_i^k] \sum_{m=1}^M f'[(I_{Mm})_i^k] w_{xm}^k\end{aligned}$$

从而得关于点 $P_i(x_i, y_i, z_i)$ 的动态运动方程为

$$\begin{aligned}\dot{x}_i &= -\eta \left[2w_i(2x_i - x_{i-1} - x_{i+1}) + w_c \sum_k f'[(T_I)_i^k] \sum_{m=1}^M f'[(I_{Mm})_i^k] w_{xm}^k \right] \\ \dot{y}_i &= -\eta \left[2w_i(2y_i - y_{i-1} - y_{i+1}) + w_c \sum_k f'[(T_I)_i^k] \sum_{m=1}^M f'[(I_{Mm})_i^k] w_{ym}^k \right] \\ \dot{z}_i &= -\eta \left[2w_i(2z_i - z_{i-1} - z_{i+1}) + w_c \sum_k f'[(T_I)_i^k] \sum_{m=1}^M f'[(I_{Mm})_i^k] w_{zm}^k \right]\end{aligned}$$

其中

$$f'(\cdot) = \frac{1}{T} f(\cdot) [1 - f(\cdot)]$$

以上推证过程非常类似于 BP 算法。其差别在于,该算法优化的变量是网络的输入,而在

标准的 BP 算法中优化的变量是连接权系数。

3. 物体表示为多面体时的路径规划

上述算法很容易地推广到物体表示为多面体的情况。这时有两个不同的特点需要考虑：(1) 当物体沿路径运动时，不仅要考虑物体的移动，同时也要考虑物体的转动；(2) 在计算物体关于障碍物的碰撞罚函数时，应该考虑物体上的许多点，而不只是一个点。

当只用一个点来表示物体时，只需一个点 $P_i(x_i, y_i, z_i)$ 即可表示物体的位置，现在需要用固结在物体上的一个坐标系来描述该物体的位置和姿态。定义 $P_i(x_i, y_i, z_i)$ 为该坐标系的原点在基坐标中的位置，采用滚动角 γ 、俯仰角 β 或偏转角 α 表示该坐标系的姿态或方位。以后所说路径点即指的是该坐标系的位置和姿态，它共 6 个分量。图 3.69 表示了物体沿路径的位姿表示。

为了确定物体与障碍物的碰撞程度，可在物体上选择测试点，则该物体的碰撞罚函数即为所有这些测试点的碰撞罚函数之和。这些测试点相对于物体坐标系是固定的，而相对基坐标的位姿可由下式确定。

$$q_{i,j} = R_i p'_j + p_i$$

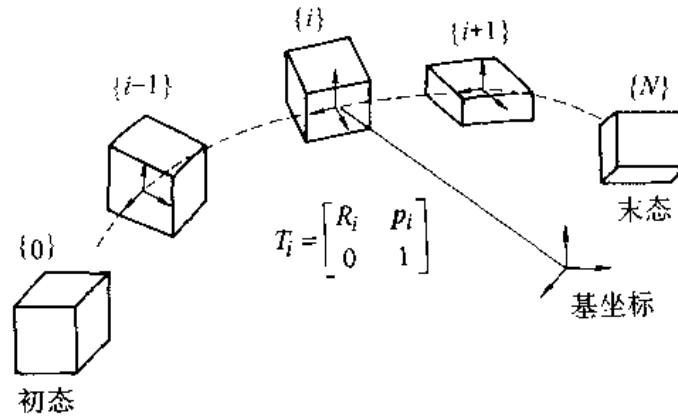


图 3.69 物体沿路径的位姿表示

其中 $p'_j = (x'_j \ y'_j \ z'_j)^T$ 表示物体上第 j 个测试点相对于物体坐标系的位置向量； $q_{i,j} = [X_{i,j} \ Y_{i,j} \ Z_{i,j}]^T$ 表示物体上第 j 个测试点在第 i 个路径点时相对于基坐标的位置向量； $p_i(x_i, y_i, z_i)$ 是物体坐标系原点在第 i 个路径点时的位置向量； R_i 是物体坐标系在第 i 个路径点时的姿态矩阵，容易求得

$$R_i = R_z(\gamma)R_y(\beta)R_x(\alpha) = \begin{bmatrix} c\gamma c\beta_i & c\gamma s\beta_i s\alpha_i - s\gamma c\alpha_i & c\gamma s\beta_i c\alpha_i + s\gamma s\alpha_i \\ s\gamma c\beta_i & s\gamma s\beta_i s\alpha_i + c\gamma c\alpha_i & s\gamma s\beta_i c\alpha_i - c\gamma s\alpha_i \\ -s\beta_i & c\beta_i s\alpha_i & c\beta_i c\alpha_i \end{bmatrix}$$

式中采用了简记符号 $c\alpha \triangleq \cos \alpha$, $s\alpha \triangleq \sin \alpha$ 。这样根据前面的式子便可计算出物体上各测试点在基坐标中的位置，将它作为神经网络的输入便可计算出所对应的碰撞罚函数。

基于上面的分析，可求得该物体沿路径的总的能量函数为

$$E = w_l E_l + w_c E_c = w_l \sum_{i=1}^{N-1} L_i^2 + w_c \sum_{i=1}^N \sum_{j=1}^J \sum_{k=1}^K C_{i,j}^k$$

其中 L_i 表示第 $i-1$ 个路径点物体坐标系原点到第 i 个路径点物体坐标系原点间的距离。 $C_{i,j}^k$ 表示第 i 个路径点上第 j 个测试对第 k 个障碍物的碰撞罚函数,类似前面将物体作为质点时的分析,通过对 E 求导,我们可以建立起使总能量趋向极小的各变量的动态运动方程如下:

$$\begin{aligned}\dot{x}_i &= -\eta_i [2w_i(2x_i - x_{i-1} - x_{i+1}) + w_c \sum_j \sum_k \frac{\partial C_{i,j}^k}{\partial x_i}] \\ \dot{y}_i &= -\eta_i [2w_i(2y_i - y_{i-1} - y_{i+1}) + w_c \sum_j \sum_k \frac{\partial C_{i,j}^k}{\partial y_i}] \\ \dot{z}_i &= -\eta_i [2w_i(2z_i - z_{i-1} - z_{i+1}) + w_c \sum_j \sum_k \frac{\partial C_{i,j}^k}{\partial z_i}] \\ \dot{\alpha}_i &= -\eta_i [w_c \sum_j \sum_k (\nabla_{q_{i,j}} C_{i,j}^k)^T R_\alpha p_j^i] \\ \dot{\beta}_i &= -\eta_i [w_c \sum_j \sum_k (\nabla_{q_{i,j}} C_{i,j}^k)^T R_\beta p_j^i] \\ \dot{\gamma}_i &= -\eta_i [w_c \sum_j \sum_k (\nabla_{q_{i,j}} C_{i,j}^k)^T R_\gamma p_j^i]\end{aligned}$$

其中

$$\begin{aligned}\nabla_{q_{i,j}} C_{i,j}^k &= \begin{bmatrix} \frac{\partial C_{i,j}^k}{\partial X_{i,j}} & \frac{\partial C_{i,j}^k}{\partial Y_{i,j}} & \frac{\partial C_{i,j}^k}{\partial Z_{i,j}} \end{bmatrix}^T \\ \frac{\partial C_{i,j}^k}{\partial X_{i,j}} &= f'[(T_1)_{i,j}^k] \sum_{m=1}^M f'[(I_{Mm})_{i,j}^k] w_{xm}^k \\ \frac{\partial C_{i,j}^k}{\partial Y_{i,j}} &= f'[(T_1)_{i,j}^k] \sum_{m=1}^M f'[(I_{Mm})_{i,j}^k] w_{ym}^k \\ \frac{\partial C_{i,j}^k}{\partial Z_{i,j}} &= f'[(T_1)_{i,j}^k] \sum_{m=1}^M f'[(I_{Mm})_{i,j}^k] w_{zm}^k \\ R_\alpha = \frac{\partial R_i}{\partial \alpha_i} &= \begin{bmatrix} 0 & c\gamma_i s\beta_i c\alpha_i + s\gamma_i s\alpha_i & -c\gamma_i s\beta_i s\alpha_i + s\gamma_i c\alpha_i \\ 0 & s\gamma_i s\beta_i c\alpha_i - c\gamma_i s\alpha_i & -s\gamma_i s\beta_i s\alpha_i - c\gamma_i c\alpha_i \\ 0 & c\gamma_i c\alpha_i & -c\beta_i s\alpha_i \end{bmatrix} \\ R_\beta = \frac{\partial R_i}{\partial \beta_i} &= \begin{bmatrix} -c\gamma_i s\beta_i & c\gamma_i c\beta_i s\alpha_i & c\gamma_i c\beta_i c\alpha_i \\ -s\gamma_i s\beta_i & s\gamma_i c\beta_i s\alpha_i & s\gamma_i c\beta_i c\alpha_i \\ -c\beta_i & -s\beta_i s\alpha_i & -s\beta_i c\alpha_i \end{bmatrix} \\ R_\gamma = \frac{\partial R_i}{\partial \gamma_i} &= \begin{bmatrix} -s\gamma_i c\beta_i & -s\gamma_i s\beta_i s\alpha_i - c\gamma_i c\alpha_i & -s\gamma_i s\beta_i c\alpha_i + c\gamma_i s\alpha_i \\ c\gamma_i s\beta_i & c\gamma_i s\beta_i s\alpha_i - s\gamma_i c\alpha_i & c\gamma_i s\beta_i c\alpha_i + s\gamma_i s\alpha_i \\ 0 & 0 & 0 \end{bmatrix}\end{aligned}$$

4. 避免局部极值问题的模拟退火方法

在前面所介绍的寻优算法中很可能存在局部极值问题。也就是说,由路径长度及碰撞罚函数所组成的总能量函数可能有多个极值点,因而有可能停留在某个局部极小点,而不能确保总能量一定达到全局的极小点。局部极小点所对应的路径可能比最优路径要长很多,或者不能完全躲避障碍物。为此需要设法找到一种能够跳出局部极值的方法。

模拟退火方法是一种可以跳出局部极值的有效方法,它能够解决诸如旅行商等多种优化问题。所谓模拟退火就是模拟金属退火的过程,即首先用高温将金属熔化,然后逐渐缓慢冷却,直到形成良好的晶体结构,也就是进入一种具有最小能量的状态。

模拟退火方法也可用于路径规划算法以避免局部极值。具体实现时是通过改变前面式子所示的 S 型激发函数中的参数 T ,它相当于金属退火中的温度。如图 3.70 所示,当 T 很大时, S 型曲线比较平坦, T 较小时, S 型曲线比较陡峭,当 $T \rightarrow 0$ 时, S 型曲线趋向阶跃函数。 S 型激发函数影响总的碰撞罚函数,当 T 较大时,罚函数能量曲面在障碍物的边界处变化较平缓,这时它只是粗略地反映障碍物的形状。这时在障碍物内部,罚函数能量曲面有一定的斜度。因此,当路径点位于障碍物内部时,由于能量函数曲面有一定斜度,它将驱使该路径点向低洼的方向运动。而当 T 非常小时,罚函数网络像一个开关:路径点在障碍物内部时它输出 1,路径点在障碍物外部时它输出 0。罚函数能量曲面除了在障碍物的边界处很陡峭外,其余地方均很平坦,这就使得路径点很难沿着表面运动。正是利用了参数 T 与罚函数能量曲面的这种关系,通过开始用较高的“温度” T ,然后逐渐减小 T ,从而达到了模拟退火的效果。

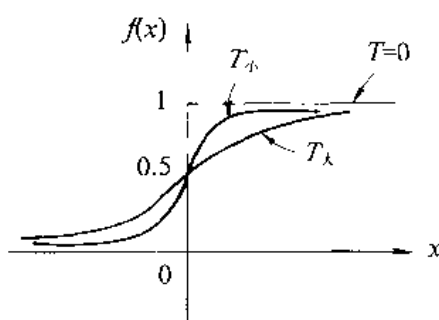


图 3.70 S 型激发函数 $f(x)=1/(1+e^{-x/T})$ 随 T 的变化

可以证明,当“温度” T 按以下规律

$$\frac{T_a(t)}{T_0} = \frac{1}{\log(1+t)}$$

变化时,退火一定能达到全局的极小值。其中 T_0 是起始的高温, $T_a(t)$ 是随时间变化的人为设置的温度。由于按上式变化收敛速度较慢,所以可采用如下的模拟退火规律:

$$\frac{T_a(t)}{T_0} = \frac{1}{1+t}$$

采用该规律大大加快了收敛速度,缩短了路径规划的计算时间,但不一定能确保获得全局的最优。

5. 仿真举例

图 3.71 所示为一较为简单的情况:平面中有两个障碍物,物体表示为质点,初始路径可以任意选择,这里选为从起点到终点的直线,如图 3.71(a)所示。图 3.71(b)表示了碰撞罚函数的形状。由于罚函数在障碍物边界处变化很陡。因此靠近边界处的点的运动远远快于其它地方的点。图 3.71(c)说明了路径点的收敛过程,图 3.71(d)显示了最终的无碰

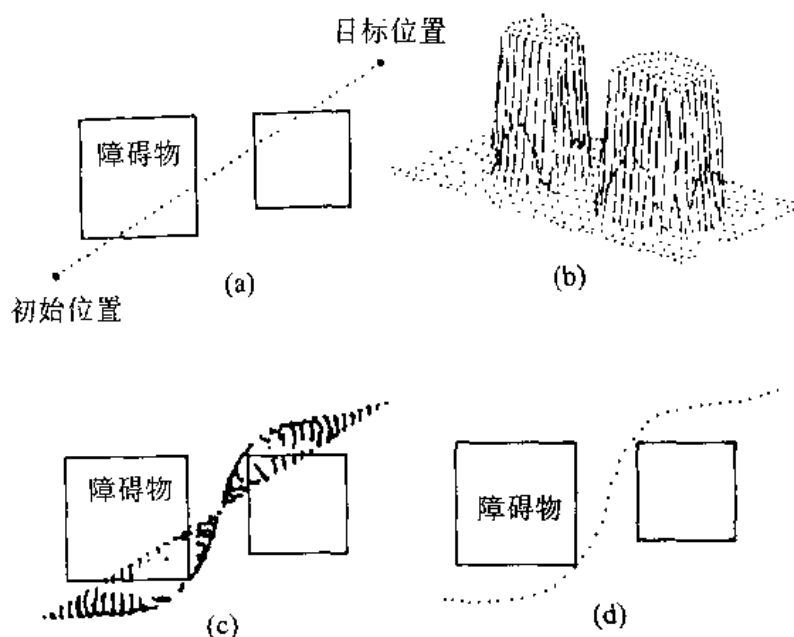


图 3.71 一个物体点与两个障碍物的路径规划

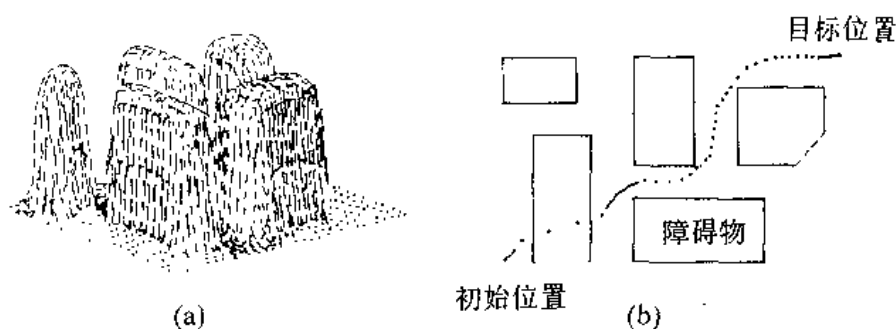


图 3.72 路径规划收敛到局部极值

撞路径。

图 3.72 是另外一个仿真的例子,说明了若不采用模拟退火方法,它将收敛到局部极值。这里共有 5 个障碍物,并且靠得比较近。物体也看作为质点。图 3.72(a)显示了当时的碰撞罚函数,图 3.72(b)显示了最后寻优得到的结果,显然它们留在一个局部极值上。这是因为初始的路径点有一部分是在障碍物的内部,那里的碰撞罚函数曲面非常平坦,因而不足以驱使位于其上的路径点运动到障碍物之外。

对于上面同样的问题,若采用模拟退火方法,开始用足够高的“温度” T ,然后逐渐减小 T ,那么那些原在障碍物内的路径点即可走出障碍物,并最终收敛到最优的无碰撞路径。图 3.73 显示了碰撞罚函数及相应的路径点的运动在退火过程中随时间的变化。起始 $T_0=0.5$,逐渐变化到 $T=0.1$ 。

以上例子中物体均视为质点。图 3.74 显示了将物体视为多面体时进行路径规划的仿

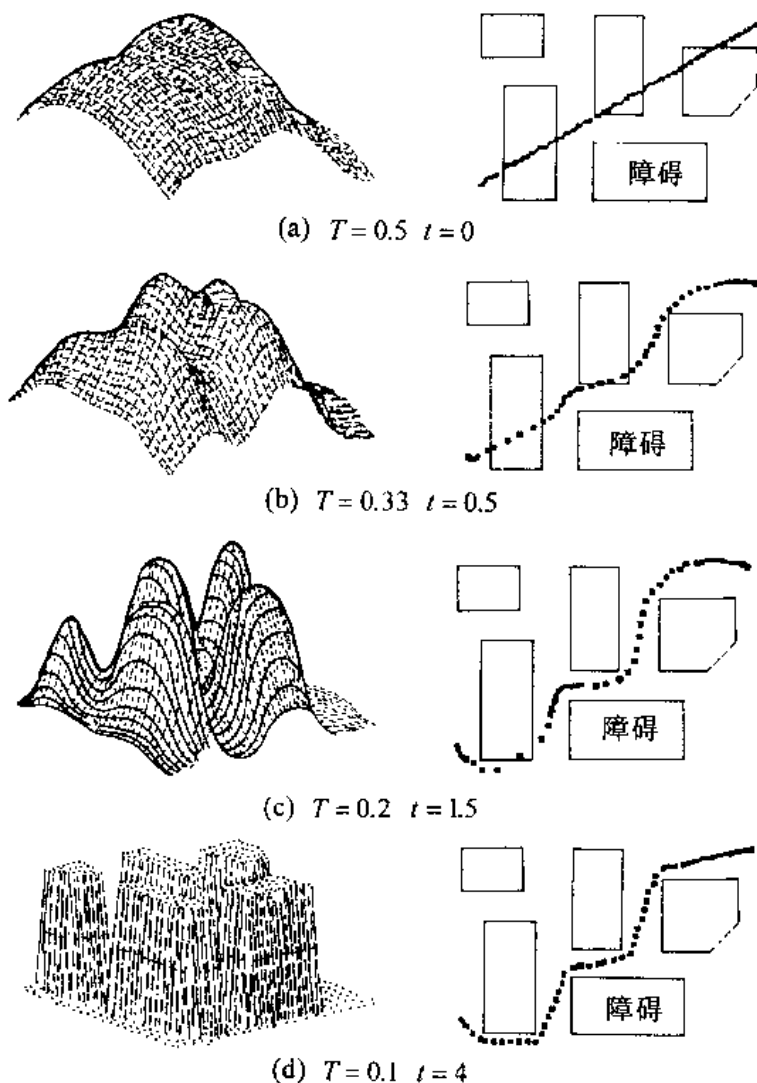


图 3.73 利用模拟退火路径规划收敛到全局极值

真结果。图 3.74(a)表示了一个矩形物体只容许平移的情况,图 3.74(b)表示了既可平移又可旋转的路径规划结果。为了获得较为安全的路径,物体上应选取足够多的测试点。

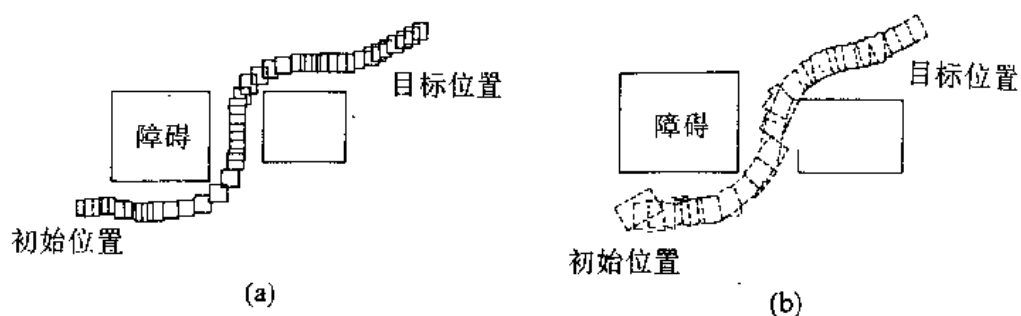


图 3.74 物体作为多面体时的路径规划

6. 小结

这里介绍的用神经网络进行路径规划提供了一种大规模并列计算的方法。因此,增加

路径点及障碍物的个数只是增加网络的规模,而并不增加总的计算时间。为了提高所规划的路径的精度,可以增加路径点的个数。也可以适当地将物体放大,设置一个虚拟的边界,以求获得更大的安全系数。

处理三维路径规划比之二维路径规划要多处理一个位置变量和两个姿态变量。由于上面所说的并列计算的特点,所以处理三维路径规划问题也并不带来额外的复杂性。因此二维路径规划问题可看成是三维路径规划的特例,即将其中的一个位置变量和两个姿态变量取为常数即可。因而上述规划算法既可适用三维也可适用于二维情况,而无需做任何修正。这是与常规的路径规划算法完全不同的。

如前所述,采用如前面式子所示的快速模拟退火步骤在障碍物较密集的情况仍可能停留在局部极值,这时一部分路径可能穿越障碍。这种情况可以通过检查路径点间的距离检测出来,因为无碰撞路径段的路径点大致上是均匀分布的。对于所检测出的穿越障碍物的部分路径,可再次使用上面的寻优算法,直至最终找到满足要求的无碰撞路径。

总之,将神经网络方法用于路径规划具有以下优点:(1) 算法所固有的并列性可用并列硬件来实现,对于有较多障碍物,有较多路径点以及物体上有较多测试点的情况,该路径规划方法也可达到实时应用的程度;(2) 算法的并列性也使得所规划的路径可以达到任意高的精度而并不增加计算时间;(3) 该算法既可用于二维规划,也可用于三维规划,而无需附加的修正;(4) 由于采用了模拟退火方法,从而可解决局部极值问题。

同时也应看到,该神经网络的路径规划方法也有一定的局限性,它只适用于环境是已知的情况,且障碍物必须是静止的。对于环境信息部分已知或障碍物运动的情况,必须对上述算法加以修正,这是需要进一步研究的问题。

本节讨论了用神经网络实现机器人运动学控制,动力学控制及路径规划。神经网络在机器人控制中的应用尚不止这些,其它如基于传感器的机器人控制、机器人的手—眼系统等也可用神经网络来实现,并已有不少人在这方面做了很多研究工作。

利用神经网络方法控制机器的很大优点是它不需要预先知道机器人的确切参数。但如果能够将神经网络方法与获得的知识及有关启发信息结合在一起,并加以充分利用,则有可能产生更好的机器人智能控制方法。

参 考 文 献

- [1] McLulloch W, Pitts W. A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 1943, 7
- [2] Hebb D. *Organization of Behavior*. New York, John Wiley & Sons, 1949
- [3] F. Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, 1958, 65
- [4] Minsky M, Papert S. *Perceptrons*. Cambridge: MIT Press, 1969
- [5] Hopfield J. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Science*, 1982, 79
- [6] Hopfield J. Neurons with Graded Response Have Collective Computational Proper-

- ties Like Those of Two-State Neurons. *Proceedings of the National Academy of Science*, 1984, 81
- [7] Rumelhart D, McClelland J. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Cambridge: Bradford Books, MIT Press, 1986, 1,2
- [8] Simpson P K. *Artificial Neural Systems*. Pergamon Press, 1990
- [9] 杨行峻, 郑君里. *人工神经网络*. 高等教育出版社, 1992
- [10] 张立明. *人工神经网络的模型及其应用*. 复旦大学出版社, 1993
- [11] Hammerstrom D. *Working with Neural Networks*. *IEEE Spectrum*, July, 1993
- [12] 邓志东, 孙增圻. 利用线性再励的自适应变步长快速 BP 算法. *模式识别与人工智能*, 1993, 6(4)
- [13] Lin C T, Lee C S G. Neural-Network-Based Fuzzy Logic Control and Decision System. *IEEE Trans. on Computers*, 1991, 40(12)
- [14] Albus J S. A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller(CMAC). *Trans. ASME, J. Dyn. Syst. Meas. Control*, 1995, 97
- [15] Page G F, Gomm J B, Williams D. *Application of Neural Networks to Modelling and Control*. Chapman and Hall, 1993
- [16] Lane S H, Handelman D A, Gelfand J J. Higher-Order CMAC Neural Networks—Theory and Practice. *American Control Conference*, Boston, 1991
- [17] Jin Y, Pip A G, Winfield A. *Neural Networks for Manipulator Control: Methodology, Stability and Simulations*. *World Congress on Neural Networks*, 1994
- [18] 孙增圻, 邓志东. 一种类似 CMAC 模糊神经网络及其在控制中的应用. 第二届全国智能控制专家讨论会, 1994
- [19] Lee S, Bekey G A. *Application of Neural Networks to Robotics. Control and Dynamic Systems*, 1991; 39
- [20] 吴晓涛, 孙增圻, 邓志东. 基于网络的并行路径规划算法. 第二届全国智能控制专家讨论会, 1994
- [21] Pham D T, Liu X. *Neural Networks for Identification, Prediction and Control*. Springer-Verlag London Limited, London, 1995
- [22] Harris C J, Moore C G, Brown M. *Intelligent Control: Aspects of Fuzzy Logic and Neural Nets*. World Scientific Publishing Co. Pte. Ltd., Singapore, 1993
- [23] Hunt K J, *et al.* *Neural Networks for Control Systems-A Survey*. *Automatica*, 1992, 28(6)
- [24] Narendra K S, *et al.* *Identification and Control of Dynamical Systems Using Neural Networks*. *IEEE Trans. Neural Networks*, 1990, 1(1)
- [25] Miller T W, *et al.* (eds.). *Neural Networks for Control*. MIT Press, Cambridge, MA, 1990
- [26] Antsaklis P J (ed.). *Special Issue on Neural Networks in Control Systems Maga-*

zine, 1990, 10(3)

- [27] Cotter N E. The Stone-Weierstrass Theorem and Its Application to Neural Networks. IEEE Trans. Neural Networks, 1990, 1(1)
- [28] 张恭庆,林源渠. 泛函分析讲义. 北京大学出版社,1987
- [29] 柯罗夫金 И И. 线性算子与逼近论. 人民教育出版社,1960
- [30] Deng Z D, Zhang Z X, Jia P F. A Neural-Fuzzy BOXES Control System with Reinforcement Learning and Its Applications to Inverted Pendulum. Proc 1995 IEEE International Conference on SMC, Vancouver, Canada, Oct. 22—25, 1995
- [31] Sun Z Q, Deng Z D. A Control Method Based on a Fuzzy CMAC Neural Network. Proc International Symp. of Young Investigators on Control, 1994
- [32] 邓志东,孙增圻,刘建伟. 神经网络异步自学习控制系统. 自动化学报,1995, 21(5)
- [33] 邓志东,孙增圻,张再兴. 一种模糊 CMAC 神经网络. 自动化学报,1995, 21(3)
- [34] 邓志东,孙增圻. 异步自学习控制的频域稳定性分析. 信息与控制,1994, 23(1)
- [35] 邓志东,孙增圻. 神经网络控制的研究现状与展望. 中国计算机报,1994 年 5 月 3 日;第 17 期(总第 503 期)

第4章 专家控制

专家控制是智能控制的一个重要分支。

专家控制的实质是使系统的构造和运行都基于控制对象和控制规律的各种专家知识,而且要以智能的方式来利用这些知识,求得受控系统尽可能地优化和实用化。因此,专家控制又称作基于知识的控制或专家智能控制。

专家控制目前尚未形成系统的理论体系。本章主要介绍专家控制的基本思想,典型实例的工作原理,以及有关的研究课题。

4.1 概 述

4.1.1 专家控制的由来

传统的自动控制学科从古典控制理论发展到现代控制理论,并出现了自适应控制等高级控制技术,取得了巨大的进展。这些进展主要源于数学分析和数值计算这两个方面的理论和技术。例如,从控制系统的实践环节看,辨识、分析、仿真、设计等都产生了许多新的思想、概念、方法和算法,而“实现”这一环节基本上都从模拟方式发展为数字方式。

然而,我们可以看到,传统控制系统的结构基本没有改变,仍然是机器单独作用的反馈控制。在设定值确定之后,系统的运行排斥了人的干预,人-机之间缺乏交互。系统的性能是离线监控的。如果出现故障,只能停机,进行离线处理。更值得注意的是,传统控制系统的机制基本上也没有改变,仍然是单纯地执行各种控制规律算法。系统对于控制对象在环境中的参数、结构的变化缺乏应变能力,对于控制器的参数、结构缺乏合适的调整方法。

传统控制理论的不足,在于它必须依赖于受控对象或过程的严格的数学模型,试图针对精确模型来求取最优的控制效果。而实际的受控对象或过程存在着许多难以建模的因素。完善的模型一般都难以解析表示,模型过于简化往往又不足以解决实际问题。

80年代初,正当人工智能中的专家系统技术方兴未艾之时,自动控制领域的学者和工程师开始把专家系统的思想和方法引入控制系统的研究及其工程应用。专家系统是一种基于知识的系统,它主要面临的是各种非结构化问题,尤其能处理定性的、启发式或不确定的知识信息,经过各种推理过程达到系统的任务目标。专家系统技术的特点为解决传统控制理论的局限性提供了重要的启示,二者的结合导致了专家控制这种新颖的控制系统设计和实现的方法。

4.1.2 专家系统

专家系统是一种人工智能的计算机程序系统,这些程序软件具有相当于某个专门领域的专家的知识 and 经验水平,以及解决专门问题的能力。

自从1977年Feigenbaum在第5届国际人工智能大会上提出“知识工程”的概念以

来,知识获取、知识表示和知识利用等技术逐渐形成人工智能研究的一大分支。Feigenbaum 认为,人的智能活动,包括理解、解决问题的能力,甚至学习能力,都完全依靠知识。特定知识的处理和运用是智能行为的中心问题。基于知识推理的专家系统,作为知识工程的研究范例,已经开创了人工智能中尤其活跃而且富有成效的应用领域。目前,专家系统的开发不但早已走出了实验室,而且成为软件产业的一个新分支——知识产业。知识工程的理论,技术和方法,与其他人工智能的研究成果一样,对智能控制的形成和发展提供了重要的借鉴。

1. 专家系统的基本组成

专家系统的基本组成结构如图 4.1 所示。

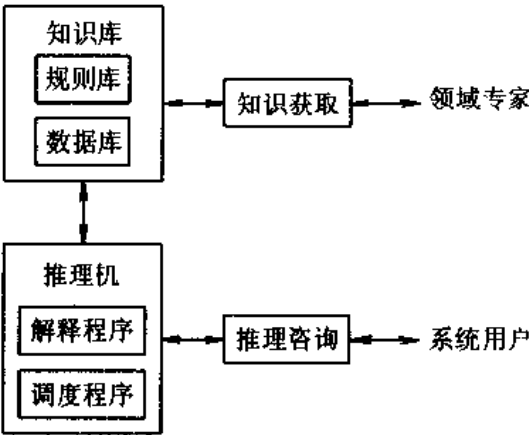


图 4.1 专家系统的基本组成

由图可知,知识库和推理机是专家系统中两个主要的组成要素。

知识库存放着作为专家经验的判断性知识,例如表达建议、推断、命令、策略的产生式规则等,用于某种结论的推理、问题的求解,以及对于推理、求解知识各种控制知识。知识库中还包括另一类叙述性知识,也称作数据,用于说明问题的状态,有关的事实和概念,当前的条件以及常识等。

完整的知识库还应包括具有管理功能的软件系统,主要用于对知识条目的查询、检索、增删、修改、扩充等操作。

知识库通过“知识获取”机构与领域专家相联系,形成了专家系统与领域专家的人-机接口。知识获取的过程即为建立和更新知识库,并完成对知识条目的测试和精炼的过程。知识获取的手段可以采用“专题面谈”、“口语记录分析”等人工移植方式,也可以采用机器学习的方法。

推理机实际上是一个运用知识库中提供的两类知识,基于某种通用的问题求解模型,进行自动推理、求解问题的计算机软件系统。它包括一个解释程序,用于决定如何使用判断性知识推导新的知识,还包括一个调度程序,用于决定判断性知识的使用次序。推理机的具体构造取决于问题领域的特点,及专家系统中知识表示和组织的方法。

推理机的运行可以根据不同的控制策略:从原始数据和已知条件推断出结论的方法

称为正向推理或数据驱动策略:先提出结论或假设,然后寻找支持这个结论或假设的条件或证据,如若成功则结论成立,否则再重新假设,这种方法称为反向推理或目标驱动策略;运用正向推理帮助系统提出假设,再运用反向推理寻找证据,这种方法即为双向推理或混合控制。

推理机通过“推理咨询”机构与系统用户相联系,形成了专家系统与系统用户之间的人-机接口。系统可以输入并“理解”用户有关领域问题的咨询提问,再向用户输出问题求解的结论,并对推理过程作出解释。人-机之间的交互信息一般要在机器内部表达形式与人可接受的形式(如自然语言、图文等)之间进行转换。

2. 专家系统的特点

专家系统通过某种知识获取手段,把人类专家的领域知识和经验技巧移植到计算机中,并且模拟人类专家的推理、决策过程,表现出求解复杂问题的人工智能。它与传统的计算技术和常规的软件程序相比,具有显著的特点。

在功能上,专家系统是一种知识信息处理系统,而不是数值信息计算系统。它依靠知识表示技术确定问题的求解途径,而不是基于数学描述方法建立处理对象的计算模型;它主要采用知识推理的各种方法求解问题,制订决策,而不是在固定程序控制下通过执行指令完成求解任务。

在结构上,专家系统的两个主要组成部分——知识库和推理机是独立构造、分离组织,但又相互作用的。这不能简单地看作是一种编程技巧,而是说明一个知识基系统的首要特征是它具有一个知识体的核心部分。维持专家系统的知识是明确的、可存取的,而且是可积累的。常规的软件程序尽管也包含许多领域知识,但这些知识往往是隐含的,它们与求解问题的方法混杂在一起,无法得到单独的操作和控制。

在性能上,专家系统具有启发性,它能够运用专家的经验知识对不确定的或不精确的问题进行启发式推理,运用排除多余步骤或减少不必要计算的思维捷径和策略;专家系统具有透明性,它能够向用户显示为得出某一结论而形成的推理链,运用有关推理的知识(元知识)检查导出结论的精度、一致性和合理性,甚至提出一些证据来解释或证明它的推理;专家系统具有灵活性,它能够通过知识库的扩充和更新提高求解专门问题的水平或适应环境对象的某些变化,通过与系统用户的交互使自身的性能得到评价和监护。

3. 知识的表示

知识表示、知识获取和知识推理是人工智能知识工程的重要课题。其中,知识表示是专家系统在构造方法上区别于常规程序系统的特征。专家知识的表达形式反映领域问题的性质,影响到知识的获取,知识的操作和利用。

有关知识表示的详尽讨论可见人工智能的各种专著,以下扼要列举专家系统中知识表示的常用形式。

(1) 产生式规则

产生式规则的一般形式为“条件→行动”或“前提→结论”,即用“IF-THEN”语句表示一个知识项。

产生式规则的左半部一般为若干事实的逻辑积,确定了规则可应用的先决条件,右半部描述了规则的先决条件得到满足时所采取的行动或得出的结论,据此可对数据库进行

操作,生成新的状态。产生式规则的先决条件不断与数据库中的事实进行匹配,在顺序执行规则的同时就形成推理链。产生式规则的推理机制是以演绎推理为基础的。

以著名的化学专家系统 DENDRAL(E. Feigenbaum, 1968)、医学专家系统 MYCIN (E. Shortliffe 等, 1972)和地质探矿专家系统 PROSPECTOR(SRI, 1981)为典范,产生式规则已成为专家系统中最流行的知识表示方法,这类系统一般又称为基于规则的系统或产生式系统。

(2) 框架

框架是一种主要表示叙述性知识的数据结构,通常用于描述事物、概念的固定不变的若干方面。一个框架由各个描述方面的槽组成,每个槽可有若干侧面,而每个侧面又可有若干个属性值,如此形成一个具有嵌套的连接表:

(框架名)

```
(槽 1)  (侧面 11) (值 111) (值 112)
        (侧面 12) (值 121) (值 122) ...
        :
(槽 2)  (侧面 21) (值 211) (值 212) ...
        (侧面 22) (值 221) (值 222) ...
        :
:
:
```

框架的内容可根据需要取舍。框架的侧面可以是“值”侧面(属性值已知的侧面),或者是“默认”侧面(填入默认值供属性不明确时用),或者是“如果需要”侧面(填入计算属性值的过程信息),或者是“如果加入”侧面(填入说明是否启动“如果需要”侧面中的过程)。框架可以链接起来组成具有层次结构的框架系统。基于框架表示的专家系统有肺病诊断系统 WHEEZE(D. Smith 等, 1980)。数学专家系统 AM(D. B. Lenat, 1976)采用框架和产生式规则相结合表示知识。

(3) 语义网络

语义网络是通过概念及其相互间语义关系,图解表示知识的网络。其中,结点表示事物或事件的概念,结点间用弧连接,弧上加有标记说明语义关系。另外,结点可以是变量,通过增加中间结点可以使语义网络表示多元关系。基于语义网络的最简单的推理是通过继承关系得到结点事物的属性值。

基于语义网络表示的专家系统有自然语言问答系统 NLQS(Simmons, 1973)。PROSPECTOR 系统用语义网络和规则共同表示知识。

(4) 过程

知识的过程表示法是将某一专门知识及其使用方法表达为一个求解子问题的过程,即子程序。在进行知识推理时,只需调用这些子程序。

应用过程表示的有回答系统 SIR(B. Raphael, 1968), SHRDLU(T. Winograd, 1972)等。

知识表示方法还有谓词逻辑、状态空间、概念从属、脚本、知识表达语言 KRL 等。

4.1.3 专家控制的研究状况和分类

1. 研究状况

知识工程思想和专家系统技术推动了传统控制的发展。

1984年,在布达佩斯召开的 IFAC(International Federation of Automatic Control)第9届世界大会上,J. Zaborszky 提出了系统科学的一般结构(图 4.2)。这种概念结构明确地从知识的观点改变了对控制系统的传统描述,认为系统的功能和构成实际上主要是一个专家系统。1986年,美国 52 位专家教授在加州桑塔卡拉拉大学召开了控制界的“高峰”会议,发表了共同的观点。

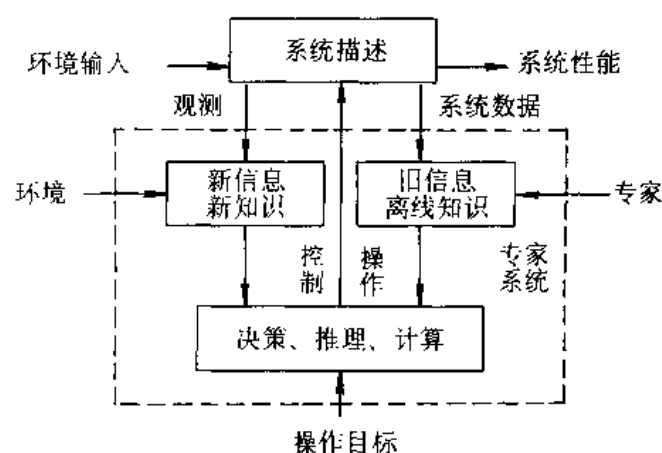


图 4.2 系统科学的一般结构

1980年以后,专家系统技术在控制问题中的应用研究逐渐增多。例如 LISP 机公司研制的用于蒸馏塔过程控制的分布式实时专家系统 PICON(R. L. Moore 等,1984),用于核反应堆环境辅助决策的专家系统 REACTOR(M. Gallanti 等,1982—1983),利用专家系统对飞行控制系统控制规律进行再组合的研究(T. L. Trankle 和 L. Z. Markosian,1985)等。另外专家系统的技术还被应用于传统的 PID 调节器和自适应控制器,例如性能自适应 PID 控制器 EXACT(E. H. Bristol,1977;T. W. Kraus 和 T. J. Myron, 1984),PI 控制器的实时专家调节器(B. Porter 等,1987)等。

在 1984 年的 IFAC 第 9 届世界大会上仅有 6 篇有关专家系统用于控制问题的研究文章,而到 1987 年 IFAC 的第 10 届世界大会上就有了 49 篇文章,而且设专门会议讨论有关问题。随后一些著名刊物纷纷增设专刊。专家系统以及其他人工智能技术在实时控制中的应用研究逐渐成为近 10 年来控制工程领域的一个新潮流。

随着智能控制学科方向的发展,我国有关专家控制技术的研究工作也非常活跃。例如,基于专家知识的智能控制研究及其在造纸过程控制中的应用(胡恒章,倪先锋等,1988—1989),智能控制器与锅炉专家控制系统的研究(郭晨,1991)专家控制系统在精馏控制中的应用(王建华、刘鸿强、潘日芳,1987)。特别是,在多方面研制实用系统的基础上,还提出了仿人智能控制理论(周其鉴、李祖枢等,1983 起)。

一般认为,专家控制研究的突出代表应首推瑞典学者 K. J. Åstrom。他对于自动控制理论的研究深有造诣,尤其在自适应控制方面。1983 年,他发表“Implementation of an Autotuner Using Expert System Ideas”一文,明确建立了将自动控制技术引入自动控制的思想,随后开展了原型系统的实验。1986 年,他在“Expert Control”一文中正式提出了“专家控制”的概念,阐述了比较深入、完整的见解。Åstrom 的研究成果对于专家控制技术的发展应用起到重要的先导作用,体现着专家控制技术的原理本质和功能特点。本章有关专家控制的主要内容以 Åstrom 的这方面研究工作为背景。

2. 类型

对于专家控制及其实现的研究可以有不同的分类看法。

根据专家系统技术在控制系统中的功能结构,可分为直接式专家控制和间接式专家控制。直接式专家控制系统中,领域专家的控制知识和经验被用来直接控制生产过程或调节受控对象,常规的控制器或调节器被代之以一个模拟手动操作功能的专家系统,直接给出控制信号。这种控制方法适用于模型不充分、不精确,甚至不存在的复杂过程。而在间接式专家控制系统中,各种高层决策的控制知识和经验被用来间接地控制生产过程或调节受控对象,常规的控制器或调节器受到一个模拟控制工程师智能的专家系统的指导、协调或监督。专家系统技术与常规控制技术的结合可以非常紧密,二者共同作用方能完成优化控制规律,适应环境变化的功能;专家系统的技术也可以用来管理、组织若干常规控制器,为设计人员或操作人员提供辅助决策作用。一般认为,紧密型的间接式专家控制研究具有典型的意义。

根据专家系统技术在控制系统中应用的复杂程度,可以分为专家控制系统和专家式智能控制器。专家控制系统具有全面的专家系统结构、完善的知识处理功能,同时又具有实时控制的可靠性能。这种系统知识库庞大、推理机复杂,还包括知识获取子系统和学习子系统,人-机接口要求较高。而专家式智能控制器是专家控制系统的简化,针对具体的控制对象或过程,专注于启发式控制知识的开发,设计较小的知识库,简单的推理机制,甚至采用“case by case”的方式,省去复杂的人-机对话接口等。当专家控制系统功能的完备性、结构的复杂性与工业过程的控制的实时性之间存在矛盾时,专家式智能控制器是合适的选择,但它与专家控制系统在基本功能上是没有本质的区别的。

还可以根据专家系统的知识表示技术或推理方式对专家控制的实现系统进行分类,例如产生式、框架式,串行推理、并行推理等。专家系统技术与大系统理论相结合,还可以设计多级、多层、多段专家控制系统。

基于模糊规则的控制也可以与专家系统技术相结合,形成所谓专家式模糊控制的研究,例如,利用一个专家控制器根据系统动态特性知识去修改模糊控制表的参数等。

有关仿人智能控制的一些研究旨在从宏观结构和行为功能上对“人”(控制专家)进行模拟,其中结合运用了专家系统等人工智能技术与传统控制理论、方法。这类研究与专家控制有关,本章也给以扼要的介绍。

4.2 专家控制的基本原理

到目前为止,专家控制并没有明确的公认定义。粗略地说,专家控制是指将专家系统的设计规范和运行机制与传统控制理论和技术相结合而成的实时控制系统设计、实现方法。

4.2.1 专家控制的功能目标

专家控制的功能目标是模拟、延伸、扩展“控制专家”的思想、策略和方法。

所谓“控制专家”,既指一般自动控制技术的专门研究者、设计师、工程师、也指具有熟练操作技能的控制系统操作人员。他们的控制思想、策略和方法包括成熟的理论方法,直觉经验和手动控制技能。专家控制并不是对传统控制理论和技术排斥、替代,而是对它的包容和发展。专家控制不仅可以提高常规控制系统的控制品质,拓宽系统的作用范围,增加系统功能,而且可以对传统控制方法难以奏效的复杂过程实现闭环控制。

专家控制的理想目标是实现这样一个控制器或控制系统:

(1) 能够满足任意动态过程的控制需要,包括时变的,非线性的,受到各种干扰的控制对象或生产过程;

(2) 控制系统的运行可以利用对象或过程的一些先验知识,而且只需要最少量的先验知识;

(3) 有关对象或过程的知识可以不断地增加、积累,据以改进控制性能;

(4) 有关控制的潜在知识以透明的方式存放,能够容易地修改和扩充;

(5) 用户可以对控制系统的性能进行定性的说明,例如:“速度尽可能快”,“超调要小”等;

(6) 控制性能方面的问题能够得到诊断,控制闭环中的单元,包括传感器和执行机构等的故障可以得到检测;

(7) 用户可以访问系统内部的信息,并进行交互,例如对象或过程的动态特性,控制性能的统计分析,限制控制性能的因素,以及对当前采用的控制作用的解释等。

专家控制的上述目标可以看作是一种比较含糊的功能定义,它们复盖了传统控制在一定程度上可以达到的功能,但又超过了传统控制技术。作一个形象的比喻,专家控制试图在控制闭环中“加入”一个富有经验的控制工程师,系统能为他提供一个“控制工具箱”,即可对控制、辨识、测量、监视、诊断等方面的各种方法和算法选择自便,运用自如,而且透明地面向系统外部的用户。

按照专家控制的功能目标,并考虑其具体实现,应当注意到,专家控制虽然引用了专家系统的思想和技术,但它与一般的专家系统还有着重要的差别:

(1) 通常的专家系统只完成专门领域问题的咨询功能,它的推理结果一般用于辅助用户的决策;而专家控制则要求能对控制动作进行独立的、自动的决策,它的功能一定要具有连续的可靠性,较强的抗干扰性。

(2) 通常的专家系统一般处于离线工作方式,而专家控制则要求在线地获取动态反

馈信息,联机完成控制,它的功能一定要具有使用的灵活性,符合要求的实时性。

4.2.2 控制作用的实现

专家控制所实现的控制作用是控制规律的解析算法与各种启发式控制逻辑的有机结合。

可以简单地说,传统控制理论和技术的成就和特长在于它针对精确描述的解析模型进行精确的数值求解。即它的着眼点主要限于设计和实现控制系统的各种核心算法。

例如,经典的 PID 控制就是一个精确的线性方程所表示的算法:

$$u(t) = K_p \left[e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{d}{dt} e(t) \right]$$

式中, $u(t)$ 为控制作用信号, $e(t)$ 为误差信号, K_p 为比例系数, $K_i = \frac{1}{T_i}$ 为积分系数, $K_d = T_d$ 为微分系数。控制作用的大小取决于误差的比例项、积分项和微分项, K_p , K_i , K_d 的选择取决于受控对象或过程的动态特性。适当地整定 PID 的 3 个系数,可以获得比较满意的控制效果,即使系统具有合适的稳定性、静态误差和动态特性。应该指出, PID 的控制效果实际上是比例、积分、微分 3 种控制作用的折衷。

PID 控制算法由于其简单可靠等特点,一直是工业控制中应用最广泛的传统技术。

再考虑作为一种高级控制形态的参数自适应控制。相应的系统结构如图 4.3 所示,其中具有两个回路。内环回路由受控对象或过程以及常规的反馈控制器组成,外环回路由参数估计和控制器设计这两部分组成。参数估计部分对受控模型的动态参数进行递推估计,控制器设计部分根据受控对象参数的变化对控制器参数进行相应的调节。当受控对象或过程的动力学特性由于内部不确定性或外部环境干扰不确定性而发生变化时,自适应控制能自动地校正控制作用,从而使控制系统尽量保持满意的性能。参数估计和控制器设计主要由各种算法实现,统称为自校正算法。

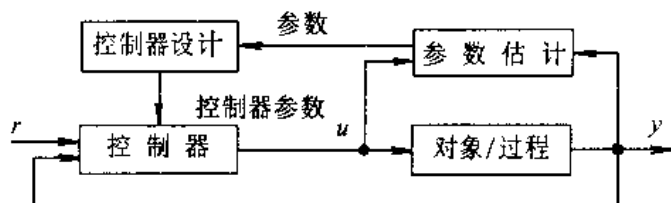


图 4.3 参数自适应控制系统

无论简单的 PID 控制或是复杂的自适应控制,要在很大的运行范围内取得完美的控制效果,都不能孤立地依靠算法的执行,因为这些算法的四周还包围着许许多多的启发式逻辑;而且要使实际系统在线运行,具有完整的功能,还需要并不能表示为数值算法的各种推理控制逻辑。

传统控制技术中存在的启发式控制逻辑可以列举如下。

(1) 控制算法的参数整定和优化

例如对于不精确模型的 PID 控制算法,参数整定常常运用 Ziegler-Nichols 规则,即

根据开环 Nyquist 曲线与负实轴的交点所表示的临界增益(K_c)和临界周期(t_c)来确定 K_p, K_i, K_d 的经验取值。这种经验规则本身就是启发式的,而且在通过试验来求取临界点的过程中,还需要许多启发式逻辑才能恰当使用上述规则。

至于控制器参数的校正和优化,更属于启发式。例如被称为专家 PID 控制器的 EX-ACT(Bristol, 1983; Kraus 和 Myron, 1984; Carmon, 1986),就是通过对系统误差的模式识别,分别识别出过程响应曲线的超调量、阻尼比和衰减振荡周期,然后根据用户事先设定好的超调量,阻尼等约束条件,在线校正 K_p, K_i, K_d 这 3 个参数,直至过程的响应曲线为某种指标下的最佳响应曲线。

(2) 不同算法的选择决策和协调

例如参数自适应控制,系统有两个运行状态:控制状态和调节状态。当系统获得受控模型的一定的参数条件时,可以使用不同的控制算法;最小方差控制、极点配置控制、PID 控制等。如果模型不准确或参数发生变化,系统则需转为调节状态,引入适当的激励,启动参数估计算法。如果激励不足,则需引入扰动信号。如果对象参数发生跳变,则需对估计参数重新初始化。如果由于参数估计不当造成系统不稳定,则需启发一种 K_c-t_c 估计器重新估计参数。最后如果发现自校正控制已收敛到最小方差控制,则转入控制状态。另外 K_c-t_c 估计器的 K_c 和 t_c 值同时也起到对备用的 PID 控制的参数整定作用。由上可知,参数自适应控制中涉及到众多的辨识和控制算法,不同算法之间的选择、切换和协调都是依靠启发式逻辑进行监控和决策的。

(3) 未建模动态的处理

例如 PID 控制中,系统元件的非线性并未考虑。当系统启停或设定值跳变时,由于元件的饱和等特性,在积分项的作用下系统输出将产生很大超调,形成弹簧式振荡,为此需要进行逻辑判断才能防止,即若误差过大,则取消积分项。

又如当不希望执行部件过于频繁动作时,可利用逻辑实现的带死区的 PID 控制等。

(4) 系统在线运行的辅助操作

在核心的控制算法以外,系统的实际运行还需要许多重要的辅助操作,这些操作功能一般都是由启发式逻辑决定的。

例如,为避免控制器的不合适初始状态在开机时造成对系统的冲击,一般采用从手动控制切入自动控制的方式,这种从手动到自动的无扰切换是逻辑判断的。

又如,当系统出现异常状态或控制幅值越限时,必须在某种逻辑控制下进行报警和现场处理。

更进一步,系统应该能与操作人员交互,以便使系统得到适当的对象先验知识,使操作人员了解、监护系统的运行状态等。

传统控制技术对于上述种种启发式控制逻辑,或者并没有作深入的揭示,或者采取了回避的态度,或者以专门的方式进行个别处理。专家控制的基本原理正是面对这些启发式逻辑,试图采用形式化的方法,将这些启发式逻辑组织起来,进行一般的处理,从它们与核心算法的结合上使传统控制表现出较好的智能性。

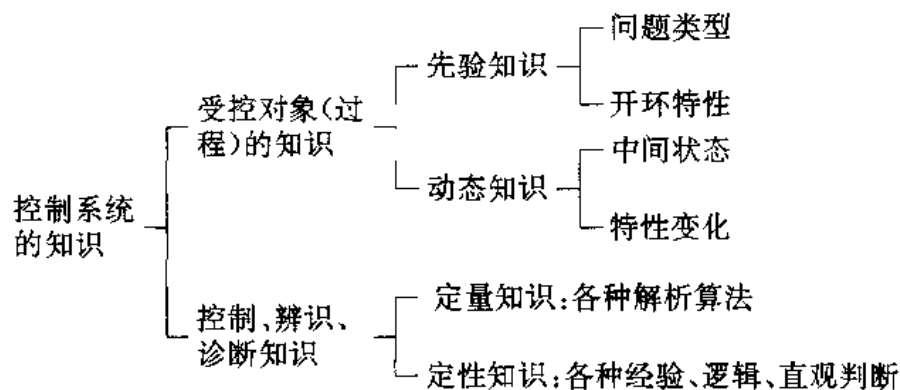
总之,与传统控制技术不同,专家控制的作用和特点在于依靠完整描述的受控过程知识,求取良好的控制性能。

4.2.3 设计规范和运行机制

专家控制的设计规范是建立数学模型与知识模型相结合的广义知识模型,它的运行机制是包含数值算法在内的知识推理。专家控制的设计规范和运行机制是专家系统技术的基本原则在控制问题中的应用。

1. 控制的知识表示

专家控制把控制系统总的看作为基于知识的系统,系统包含的知识信息内容可以表示如下:



按照专家系统知识库的构造,有关控制的知识可以分类组织,形成数据库和规则库。

(1) 数据库

数据库中包括:

事实——已知的静态数据。例如传感器测量误差,运行阈值,报警阈值,操作序列的约束条件,受控对象或过程的单元组态等。

证据——测量到的动态数据。例如传感器的输出值,仪器仪表的测试结果等。证据的类型是各异的,常常带有噪声,延迟,也可能是不完整的,甚至相互之间有冲突。

假设——由事实和证据推导得到的中间状态,作为当前事实集合的补充。例如通过各种参数估计算法推得的状态估计等。

目标——系统的性能目标。例如对稳定性的要求,对静态工作点的寻优,对现有控制规律是否需要改进的判断等。目标既可以是预定的(静态目标),也可以根据外部命令或内部运行状况在线地建立(动态目标)。各种目标实际上形成了一个大的阵列。

上述控制知识的数据通常用框架形式表示。

(2) 规则库

规则库实际上是专家系统中判断性知识集合及其组织结构的代名词。对于控制问题中各种启发式控制逻辑,一般常用产生式规则表示:

IF (控制局势) THEN (操作结论)。

其中,控制局势即为事实、证据、假设和目标等各种数据项表示的前提条件,而操作结论即为定性的推理结果。应该指出,在通常的专家系统中,产生式规则的前提条件是知识条目,推理结果或者是往数据库中增加一些新的知识条目,或者是修改数据库中其他某些原有的知识条目。而在专家控制中,产生式规则的推理结果可以是对原有控制局势知识条目的更新,还可以是某种控制、估计算法的激活。

专家控制中的产生式规则可看作是系统状态的函数。但由于数据库的概念比控制理论中的“状态”具有更广泛的内容,因而产生式规则要比通常的传递函数含义更丰富。

判断性知识往往需要几种不同的表示形式,例如对于包含大量序列成分的子问题,知识用过程式表示就比规则自然得多。

专家控制中的规则库常常构造成“知识源”的组合。一个知识源中包含了同属于某个子问题的规则,这样可以使搜索规则的推理过程得到简化,而且这种模块化结构更便于知识的增删更新。

知识源实际上是基本问题求解单元的一种广义化知识模型,对于控制问题来说,它综合表达了形式化的控制操作经验和技巧,可供选用的一些解析算法,对于这些算法的运用时机和条件的判断逻辑,以及系统监控和诊断的知识等。

2. 控制的推理模型

专家控制中的问题求解机制可以表示为如下的推理模型

$$U = f(E, K, I)$$

其中, $U = \{u_1, u_2, \dots, u_m\}$ 为控制器的输出作用集, $E = \{e_1, e_2, \dots, e_n\}$ 为控制器的输入集, $K = \{k_1, k_2, \dots, k_p\}$ 为系统的数据项集, $I = \{i_1, i_2, \dots, i_q\}$ 为具体推理机构的输出集。而 f 为一种智能算子,它可以一般地表示为

$$\text{IF } E \text{ AND } K \text{ THEN (IF } I \text{ THEN } U),$$

即根据输入信息 E 和系统中的知识信息 K 进行推理,然后根据推理结果 I 确定相应的控制行为 U 。在此智能算子的含义用了产生式的形式,这是因为产生式结构的推理机制能够模拟任何一般的问题求解过程。实际上智能算子也可以基于其他知识表达形式(语义网络、谓词逻辑、过程等)来实现相应的推理方法。

专家控制推理机制的控制策略一般仅仅用到正向推理是不够的。当一个结论不能自动得到推导时,就需要使用反向推理的方式,去调用前链控制的产生式规则知识源或者过程式知识源验证这一结论。

4.3 专家控制系统的典型结构

不少研究工作者认为,专家控制系统没有统一的体系结构,至少目前还没有形成。本节将要介绍的是 Åström 等人建立的原型系统(K. J. Åström 等,1986,K. -E. Årzen, 1990)的结构。这个原型系统是旨在研究专家控制的概念方法的实验床,从中体现的基本原理具有典型性。

4.3.1 系统结构

1. 总体结构及其特点

一个专家控制系统的总体结构如图 4.4 所示,这种典型结构具有以下特点:

(1) 两类知识及其处理过程分离构造。系统的控制器主要有两大部分组成。其中数值算法部分包含的是定量的解析知识,进行数值计算,可按常规编程,它与受控过程直接相连;另外一部分是知识基子系统,所包含的是定性的启发式知识,进行符号推理,按专家系

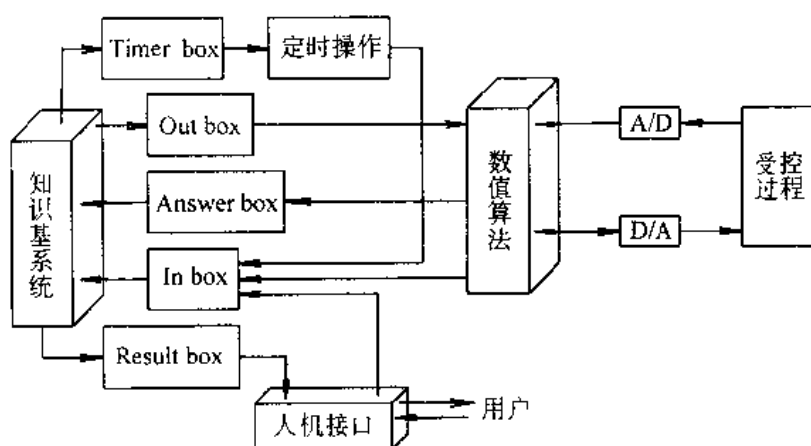


图 4.4 专家控制系统的典型结构

统的设计规范编码,它通过数值算法与受控过程间接相连。算法知识作为一种控制作用,它的具体内容没有必要硬性地转换成符号的逻辑关系存入知识库,而与知识基系统中的定性知识混杂在一起。这种分离构造方式体现了知识按属性分别表示的原则,而且还体现了智能控制系统的分层递阶原则。数值计算快速、精确,在下层直接作用于受控过程,而定性推理较慢、粗略,在上层对数值算法进行决策、协调和组织。

(2) 3 个子过程并发运行。数值算法、知识基系统、人-机通讯为 3 个独立子过程,在具体的计算机实现中是并发运行的,但数值算法拥有最高级的优先权。入-机通讯与知识基系统直接交互,而与数值算法间接联系。控制按采样周期进行,可中断入-机会话的处理。这种并发运行的机制体现了专家控制功能的有机结合,也保证了系统的实时性。

2. 数值算法

数值算法部分实际上是一个算法库,由控制、辨识和监控 3 类算法组成。

控制算法根据控制配置命令(来自知识基系统)和测量信号计算控制信号,例如 PID 算法,极点配置算法,离散滤波器算法,最小方差算法等。控制算法一次运行一种。

辨识算法和监控算法在某种意义上是从数值信号流中抽取特征信息,可以看作是滤波器或特征抽取器,仅当系统运行状况发生某种变化时,才往知识基系统中发送信息。在稳态运行期间,知识基系统是闲置的,整个系统按传统控制方式运行。辨识、监控算法中可包括延时反馈算法,递推最小二乘算法,水平交叉检测器等。

上述 3 类算法都具有一致的编程格式和合适的接口,以便增添新的候选算法,扩充算法库。

3. 内部过程通讯

系统的 3 个运行子过程之间的通讯是通过下列 5 个“邮箱”进行的。

(1) Out box 将控制配置命令、控制算法的参数变更值以及信息发送请求从知识基系统送往数值算法部分。

(2) In box 将算法执行结果、检测预报信号,对于信息发送请求的答案,用户命令,以及定时中断信号分别从数值算法,入-机接口,以及定时操作部分送往知识基系统。这些信息具有优先级说明,并形成先入先出的队列。在知识基系统内部另有一个邮箱,进入的

信息按照优先级排序插入待处理信息,以便尽快处理最重要的问题。

(3) Answer box 传送数值算法对知识基系统的信息发送请求的通讯应答信号。

(4) Result box 传送知识基系统发出的人-机通讯结果,包括用户对知识库的编辑、查询,算法执行原因、推理根据、推理过程跟踪等系统运行情况的解释。

(5) Timer box 用于发送知识基系统内部推理过程需要的定时等待信号,供定时操作部分处理。

4. 知识基系统的内部组织和推理机制

知识基系统主要由一组知识源、黑板机构和调度器 3 部分组成,如图 4.5 所示。整个知识基系统是基于所谓的黑板模型进行问题求解的。

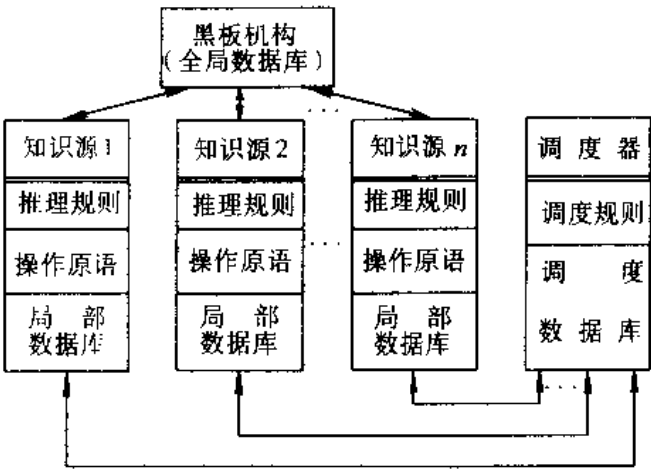


图 4.5 知识基系统的组织

(1) 黑板模型

黑板模型是一种高度结构化的问题求解模型,用于“适时”问题求解,即在最适当的时机运用知识进行推理。它的特点是能够决定什么时候使用知识、怎样使用知识。黑板模型除了将适时推理作为运用知识的策略外,还规定了领域知识的组织方法,其中包括知识源这种知识模型,以及数据库的层次结构等。

黑板模型的基本思想和工作方式可以用一种“拼板游戏”来说明。一群游戏者站在一块大黑板前,每人手里都有一些不同尺寸形状的拼板。开始时已有若干拼板粘附在黑板上,形成一个拼接图形。然后,每个游戏者根据自己手中的拼板以及黑板上的拼接图形,独立判断是否往黑板上粘附手中某块拼板,从而修改拼接图形,直至产生终结游戏的拼接图形。在游戏过程中,并没有事先规定的拼接次序,黑板上拼接图形的每次变更就提供了其他拼板的拼接可能,拼接图形协调着游戏者的配合行为。游戏的秩序可以有一个监督者来维持,他根据一些策略来选择每次走向黑板粘附拼板的游戏者,例如首先举手请求的游戏者先拼接,能使孤立拼接图形相连的拼板拥有者先拼接等。

最早研究应用黑板模型的是著名的语音理解专家系统 HEARSAY- I (L. D. Erman 等,1971—1976)。该系统可以识别在 1000 个词汇范围内的连续语音。对于讲话者通过麦克风输入的要求查询科技文献的口语句子,系统能做出解释的完全正确率和语义正确率

分别达到 74% 和 91%。HEARSAY-I 系统的工作原理是利用符号推理来帮助信号处理。它有关声学、语音学、语义学、词汇、语法等各方面知识组织成 10 多个知识源,对语音信息生成各种局部解释,这些解释在一个“黑板”上逐步得到评价和修改,最后将评价函数具有最大值的解释作为推理结果输出。

黑板模型思想和结构已被广泛运用于专家系统和人工智能的其他领域。

以下介绍黑板模型在本节提出的专家控制系统典型结构中的具体运用。

(2) 知识源

知识源是与控制问题子任务有关的一些知识模块。可以把它们看作是不同子任务问题领域的小专家。一个控制问题的子任务划分是自然的,例如控制器设计(不同的控制算法)、建模及模型验证、各种监控方法、信号历史情况的统计等。应该指出,知识源所表示的是各种数值算法所涉及的启发式逻辑,而不是算法本身的具体内容。

每个知识源都具有比较完整的知识库结构:

推理知识——“IF-THEN”产生式规则,条件部分是全局数据库(黑板)或是局部数据库(知识源内设)中的状态描述,动作、结论部分主要是对黑板信息或局部数据库内容的添加或修改。这些规则可按前向链或后向链方式控制推理。推理知识也可以用过程式表示。

局部数据库——存放与子任务相关的中间推理结果,用框架表示,其中各个槽的值即为这些中间结果。

操作原语——一类是对全局或局部数据库内容的增添、删除和修改操作,另一类是对本知识源或其他知识源的控制操作,包括激活、中止和固定时间间隔等待或条件等待(例如停止知识源的工作,直到黑板上出现某项内容或某个其他知识源结束运行)。

作为以参数自适应控制为背景的专家控制系统,有关的子任务可形成如下几个知识源:

主要控制知识源,包括最小方差控制,最小方差监控器,纹波监测器,阶数监控器;

备份控制知识源,包括 PID 控制, K_c-t_c 估计器;

估计知识源,包括参数估计,估计监控器,激励监控器,扰动信号产生器,跳变检测器;

自校正知识源,即自动调整调节;

学习知识源,即获取调节器参数,平滑并存储调节器参数,测试调度条件;

主监控知识源,包括稳定性监控器,均值和方差计算。

(3) 黑板机构

黑板是一个全局数据库,即各个知识源都可以访问的公共关系数据库。它存放、记录了包括事实、证据、假设和目标所说明的静态、动态数据。这些数据分别为不同的知识源所关注。通过知识源的访问,整个数据库起到在各个知识源之间传递信息的作用;通过知识源的推理,数据信息得到增删、修改、更新。

在 HEARSAY-I 中,黑板被组织成一个层次结构,由下至上为参数、片段、音节、词汇、词汇序列和短语等 6 个信息层。每一层上的主要信息是表示语音理解问题的部分解,即一些在特定层次上解释语音信号的假设。知识源的推理活动是利用本层或其它层上的信息,在本层或层间进行信息转换,即在每一层上找出能够正确解释语音信号的假设。由

于高层信息可以看作是若干较低层上信息的抽象,因此,推理活动一旦在高层上找到了正确的假设,语音理解过程就结束。HEARSAY-I 的黑板层次结构方法具有一般性。

在本节介绍的专家控制系统中,黑板信息类似地被组织成若干数据平面,以下举例说明两种称之为“事件表”和“假设表”的数据平面。

事件表是最重要的数据平面。

按照专家系统技术,可采用“事件驱动”这种处理时变环境的惯常方法。根据进入事件表的事件的特征,在监控作用的引导下将提出合适的动作。事件可以是知识源对原有事件的操作结果,也可以从外部进入处理过程。事件的类型主要有:受控过程的某些阈值,操作人员的指令,对于受控过程状况的新假设,对原有假设的修改,改变控制方式的请求,对于控制方式变化的“通告”,以及操作人员的信息请求等。

不同的事件表为不同的知识源提供数据。面向主要控制知识源的事件表格式如表4.1所示。其中time为时间, u 和 y 分别为控制信号和输出信号的均值, σ_u 和 σ_y 分别为 u 和 y 的标准偏差,stable为稳定性监控器的结果,regulator type为调节器类型。若有需要,最大偏差和最小偏差也可列入该表。当控制方式(手动控制、备份控制、最小方差控制或自校正控制等)改变或设定值改变时,事件表中就生成一个具体条目。根据上述事件表,知识源就可以进行有关受控过程特性的推理,例如:均值 u 与 y 间的关系及其随时间的变化,标准偏差与 u 的关系,控制方式的切换型式,在设置点发生大的变化之后系统是否进入调节方式,大部分时间所用的是哪些控制方式,系统的性能是否随时间和控制方式的变化而出现大的变化。

面向备份控制知识源的事件表如表4.2所示。其中, K_c 为临界增益, t_c 为临界周期,P,I,D为PID参数。当系统进入备份控制方式或处于备份控制方式而进行 K_c-t_c 校正时,就形成事件表。表中包括 K_c, t_c ,又包括PID参数,目的是使系统可以修改PID控制的设计。

在常增益最小方差控制周期内生成的主要数据形成的事件表如表4.3所示。其中包括最小方差控制的最优控制律(参见绪论第1.2节)中多项式 R 和 S 的阶数 n_R 和 n_s ,延迟拍数 h ,采样周期 d 以及调节器参数parameters。

结合表4.1和表4.3可以进行的推理有:所用的最小方差调节器的结构及其与运行状态的关系,所得到的性能中是否存在某种型式等。

面向参数估计知识源的事件表如表4.4所示。其中包括运行状态OC,干扰perturb,以及表4.3中的参数。当参数估计周期性地执行或者按照需要(根据运行状态的变化)时,有关数据就进入表4.4。利用该表可以为考虑参数的变化及其是否与运行状态有关而进行推理。

假设表是另一种重要的数据平面。

假设是对于受控过程运行状态的理解和推测,将各类假设进行适当的组织就形成了假设表。这里采用了逐层抽象的层次结构组织方式。较低层次的假设,主要涉及对于传感器数据的直接推导。例如,根据表4.2列出的当前控制均值和方差,就很容易导出“控制误差较小”之类的假设。较高层次的假设可以是对受控过程当前稳定性程度的估计,这类假设要利用数值计算或启发式经验规则。

对于抽象层次较高的假设,一般要求与控制工程师进行交互,以便将他的推断能力与机器的推断能力相融合。控制工程师应能利用系统赖以推理的理论根据,例如表达推理知识的产生式规则。为此,在构造系统的数据库时,可以在事件表中附上有关的推理规则编号,数据库支持这种技术检索、审查的功能。

黑板数据库的知识表示都采用框架式,复杂的框架系统能提供合适的层次结构。

表 4.1 主监控事件表

#	time	u	σ_u	y	σ_y	stable	regulator type

表 4.2 备份控制事件表

#	time	K_C	t_C	P	I	D

表 4.3 最小方差控制事件表

#	time	n_R	n_i	d	h	parameters

表 4.4 参数估计事件表

#	time	OC	perturb	h	d	n_R	n_i	parameters

(4) 调度器

调度器的作用是根据黑板的变化激活适当的知识源,并形成有次序的调度队列。

激活知识源可以采用串行或并行激活的方式,从而形成多种不同的调度策略。

串行激活方式又分 3 种:

相继触发——一个激活的知识源的操作结果作为另一个知识源的触发条件,自然激活,此起彼伏。

预定顺序——按控制过程的某种原理,预先编一个知识源序列,依次触发。例如初始调节、在检测到不同的报警状态时系统返回到稳态控制方式等情况。

动态生成顺序——对知识源的激活顺序进行在线规划。每个知识源都可以附上一个目标状态和一个初始状态,激活一个知识源即为系统状态的一次转移,通过逐步地比较系统的期望状态与知识源的目标状态,以及系统的当前状态与知识源的初始状态,就可以规划出状态转移的序列,即动态生成了知识源的激活序列。

并行激活方式即为同时激活一个以上的知识源。例如系统处于稳态控制方式时,一个知识源负责实际控制算法的执行,而另外一些知识源同时实现多方面的监控作用。

调度器的结构类似于一个知识库。其中包括一个调度数据库,用框架形式记录着各个知识源的激活状态的信息,以及某些知识源等待激活的条件信息。调度器内部的规则库包括了体现各种调度策略的产生式规则,例如:

“if a KS is ready and no other KS is running then run this KS”。

整个调度器的工作所需要的时间信息(知识源等待激活,彼此中断等)是由定时操作

部分(见图 4.4)提供的。

4.3.2 系统实现

1. 编程语言

知识基子系统总体构架是用面向对象的程序设计语言 Flavors(H. I. Cannon, 1982)以及前向链产生式系统 YAPS(Allen, 1983)实现的。

(1) Flavors

Flavors 是 Lisp 语言的扩充。它允许面向对象的程序设计。在人工智能技术的实现方法中,对于知识基系统的设计,正在从基于规则的机制朝着面向对象与利用规则匹配相混合的机制发展。

使用面向对象的程序设计方法,就是把要构造的软件程序系统表示成对象集。所谓对象,是将常规软件设计中的一组数据和操作这些数据的一组过程,作为整体看待而构成的一个独立单位。对象具有用“属性”表示的状态,还具有“行为”。当一个对象接收到其他对象发送的“消息”时,它就履行指定的操作。对象附有称之为“方法”的操作过程,它们对发送来的消息作出响应,并决定对象的行为。

对象可以分为两种型式:“类”和“实例”。具有相同结构和操作行为的对象可以归并在一起,用类统一描述,称之为类对象(在 Flavors 系统中即为 flavor);由类生成新的对象的过程称为实例化,生成的对象称为该类的实例,或实例对象。一个类对象(flavor)描述了为相应实例对象所共有的属性和方法。一个 flavor 可以是一个或多个其他 flavor 的子类,子类对象继承这个(这些)flavor 的属性和方法。一个类的子类实际上就是比这个类更为具体、特殊的类,类与子类的关系可以形成一个层次结构如图 4.6 所示。

对于面向对象的程序设计方法可详见有关文献。

(2) YAPS

YAPS(E. M. Allen, 1983)是一个前向链产生式系统,它与作为专家系统工具的 OPS 语言家族相仿,带有一种优化的步进式模式匹配算法。YAPS 的重要特点是,它被写成一个类对象 flavor,在它的数据库中可以有实例对象,这些实例对象可以是其它 YAPS 系统。

通常,YAPS 系统仅允许包含数字、原子及 flavor 实例对象的任意嵌套表作为它的数据库元素。在此处的应用中,YAPS 系统已有所扩充和修改。它的数据库中还允许有以对象一属性的集合形式描述的简单框架。另外,它还对数据库元素的增添进行自动解释。

2. 调度器的实现

调度器被实现为一个类对象 flavor,它作为一个 YAPS flavor 的子类而继承其属性和方法。不同型式的知识源在调度器中分别有一些 flavor 与之对应。每个具体的知识源是相应 flavor 的一个实例,在调度器的数据库中以框架的表示形式被存放。在这种框架中包含有说明该知识源表示形式的属性,例如是前向链产生式规则,或是过程式知识,还包含说明该知识源的状态的属性,以及说明实现该知识源的 flavor 实例的属性。

各个具体的知识源与调度器之间的交互是通过消息传递进行的,知识源 flavor 应该提供方法来对这些消息作出响应。

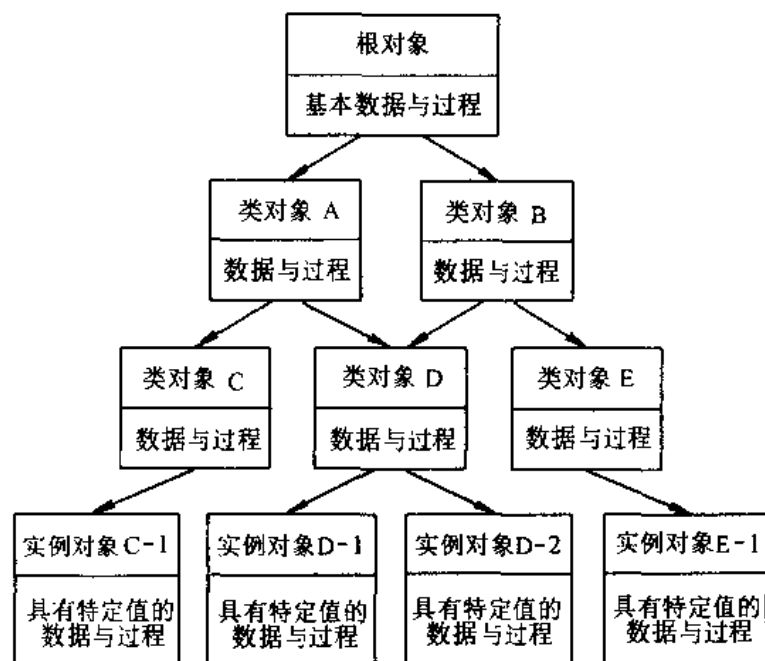


图 4.6 类的层次结构

调度策略是用一些产生式规则表示的，例如：

(P Schedule 1

(frame knowledge-source

status active

state ready

instance -x)

(~(frame knowledge-source

state tunning))

→

(modify 1 state running)

(← -x 'run))

上述这条规则中，条件部分有 2 项，必须同时满足。其中一个条件是：“若存在一个 status 属性值为“active”、state 属性值为“ready”的知识源框架”，第二个条件是：“若不存在 state 属性值为“running”的任何框架”。如果这两个条件都满足，那末第一个条件中所指的框架就变为“running”状态，即将一条“run”消息送给作为 flavor 实例的知识源。在规则中，模式匹配变量-x 是满足第一个条件的框架的 instance 属性值，表示了接受消息的 flavor 实例。

3. 知识源的实现

所有以产生式规则表示的前向链知识源或后向链知识源都被实现为 flavor 实例，这些实例继承 YAPS flavor 的属性和方法。面向对象的程序设计方法为正向推理和反向推理提供了方便。

以过程式表示的知识源是用 Lisp 函数实现的。这些 Lisp 函数的中间计算结果都要被保存起来,这样可以使过程式知识的推理能够根据需要得到中断和挂起的控制。

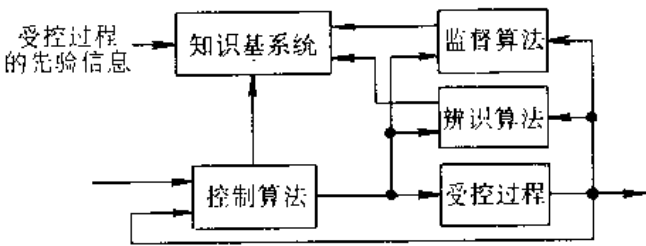
导致知识源变为待执行状态(即其 state 属性值为“ready”)的是各种事件。其中包括外部事件——由定时操作部分、人-机接口或数值算法部分发送来的消息等,以及内部事件——全局数据库元素的增添或修改等。

4.4 专家控制的例示

上一节介绍的专家控制系统已经在 VAX 11/780 上进行了数字仿真,实现了一种基于继电控制的 PID 自动调整(K. J. Åström 和 T. Hagghund, 1984)控制方法。仿真实验说明了这种智能控制方法的可行性,也验证了系统组织结构的必要性和合理性。由于恰当而清楚地表示了这种自动调节器方法中的启发式逻辑,而且逻辑与算法分离组织、共同作用,系统显示了许多优于传统控制方法、传统编程方法的特点。本节将要介绍的是该系统的部分工作原理及其具体实现。

4.4.1 自动调整过程

系统的控制原理结构如图 4.7 所示。



系统所实现的控制分为两种运行方式:自动调整方式和在线适应控制方式。在自动调整方式下,系统完成不同的调节试验从而求得受控过程的动力学特性,然后运用这些特性信息设计合适的控制器。在在线适应控制方式下,系统对控制器进行监视,必要时改变控制器的参数,或者设计新的控制器。

本节仅介绍系统的自动调整运行方式。

1. 概述

系统根据开环 Nyquist 曲线上相位分别为 0° , -90° 和 -180° 的 3 个点的有关知识,按照一种基于继电控制的 PID 自动调整原理进行工作,其工作流程如图 4.8 所示。

系统开始运行时,先询问用户(操作者)有关受控过程的先验知识,其中包括:主导时间常数的估计值,最大允许的相对稳态误差,主要的控制目标,对于设置点的快速响应或者对于负载干扰的良好的抑制性。然后,系统要求用户对过程进行手动控制,直到它在期望的运行工作点上处于稳态。此时,系统进行继电控制实验,即使控制过程发生等幅的自激振荡,完成有关的测试和推断。根据实验结果,受控过程可以得到分类:或者属于二阶或

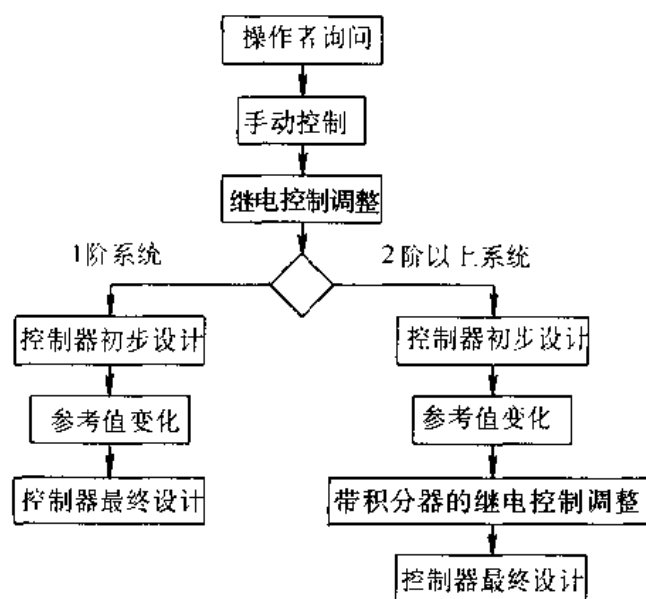


图 4.8 基于 3 点的自动调整过程

高阶模型,或者属于近似的二阶模型。对于二阶或高阶模型,系统将在反馈闭环中插入一个积分器,再进行继电控制实验,以便求取 Nyquist 曲线上的 -90° 相位点。如果改变控制的参考值,系统还可以求得受控过程的静态增益。

自动调整过程包含了大量的启发式逻辑。例如,受控过程的近似模型可以在自动调整过程的不同阶段求取,模型结构的判定依据往往是有冲突的,可以按照不同的原则来计算模型参数而且得到不同的结果等等。这些情况说明,模型的判定完全是启发式的。

2. 手动控制

用户对受控过程的手动控制,是为了启动系统的运行,并且使受控过程达到期望的运行工作点上的稳态。这样,系统就可以在稳态工作点附近对它进行控制。在两个采样时间间隔(采样周期取为主导时间常数)的范围内,比较受控过程的输出平均值,控制器就可以证实是否达到稳态。噪声的量值可确定为受控过程最大输出与最小输出的差值。

3. 继电控制实验

在继电控制特性中,滞后时间 ϵ 由噪声量值决定。继电控制非线性环节造成的自激振荡度也有噪声决定。在振荡的前半个周期内,继电控制特性的幅值从零开始增大到某个默认幅值,或者增大到误差信号超过期望的振荡幅度为止。

在自激振荡稳定后,就可以进行实验测试。如图 4.9 所示,测试的量包括:两个相继半周期的长度,在两个半周期上的振幅峰值,误差信号达到其最大值的时间 τ ,以及误差信号在一个周期内的 6 个等距离采样点 $e_0, e_1, e_2, e_3, e_4, e_5$ 。只要在半周期上取 3 个值,就可以决定自激振荡的二阶脉冲传递函数 $H(z) = (b_1z + b_2)/(z^2 - az)$ (K. J. Åström 和 T. Hägglund, 1987)。

4. 振荡分析

由继电控制自激振荡可以求得受控过程频率特性 $G(j\omega)$ 在振荡频率处的幅值和相

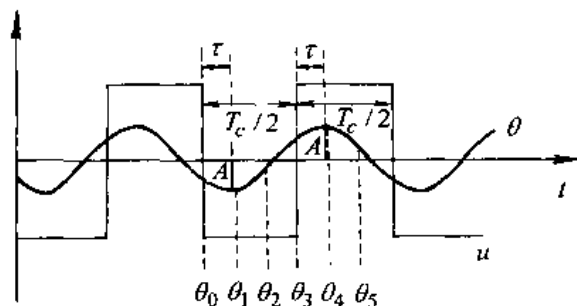


图 4.9 继电控制振荡的测试

位。而由描述函数的幅角可以求取略小于 -180° 的相位估计值。对于具有低阶动态特性的系统,这种估计值往往较差。较好的估计可以按下式计算:

$$\arg G(j\omega_c) = -\frac{\pi}{2} - \frac{2\pi\tau}{T_c}$$

式中 ω_c 为振荡频率, T_c 为振荡周期。

根据描述函数的近似分析,对于正弦输出的过程,上式中的 $\tau = T_c/4$,即相位估计值 $\arg G(j\omega_c) = -\pi$ 。对于一阶系统, $\tau = 0$, $\arg G(j\omega_c) = -\pi/2$ 。继电特性的等效增益为

$$K_c = \frac{4d}{\pi A}$$

其中 d 为继电特性的幅值, A 为测量到的误差峰值。从而受控过程的幅频特性估计为

$$|G(j\omega_c)| = \frac{\pi A}{4d}$$

5. 一阶系统

如果 $\tau/T_c < 0.05$,受控过程就可用一阶系统来近似,即 $G(s) = K_p/(1+Ts)$ 。在这种情况下,连续模型将根据离散模型 $H(z) = b_1/(z-a)$ 来计算。通过比较振荡频率处的相位计算值与量测相位的估计值,模型的合理性可以得到核实。振荡周期的量测值也需要与理论计算值进行比较。对于一阶系统 $G(s) = K_p/(1+Ts)$,振荡周期的计算值为

$$T_c = -2T \ln \frac{K_p d - \epsilon}{K_p d + \epsilon} \approx \frac{4\epsilon T}{K_p d}$$

通过量测得到的离散模型可能具有不实际的参数,例如 $0 < a < 1$ 。这样,连续模型的参数 K_p 和 T 就无法计算了。但是根据上式可算出二者的比值。

为了得到完整的连续模型,必须确定 K_p 和 T 这两个参数中的一个。为此,可以根据已有的知识粗算出一个控制器,并且使参考值作很小的变化。于是就可以通过量测稳态控制信号的差计算出稳态增益 K_c 。具体方法如下:

假设受控过程具有积分环节的特性,即 $G(s) = K_i/s$ 。 K_i 可根据上式确定,即

$$T_c = \frac{4\epsilon}{dK_i}$$

PI 控制器的增益 K 确定为

$$K = \frac{\Delta u}{\Delta n}$$

其中 Δu 为规定的稳态控制信号的允许偏差, Δn 为量测到的噪声变化范围。积分时间 T_i 的确定可根据特征方程

$$s^2 + 2\xi\omega_0 s + \omega_0^2 = s^2 + KK_p s + \frac{KK_i}{T_i}$$

因此
$$T_i = \frac{4\xi^2}{KK_p}$$

当得到稳态增益的一个估计时,就要调整 T_i ,使它满足特征方程

$$s^2 + 2\xi\omega_0 s + \omega_0^2 = s^2 + \frac{(1 + KK_p)}{T}s + \frac{KK_p}{TT_i}$$

上述设计也可用于一开始就得到一个完整的一阶模型的情况。如果带有比例控制的稳态误差 $1/(1+KK_p)$ 小于规定的稳定误差的允许值时,积分项可以省去。

6. 高阶系统

根据受控过程的延时与离散模型的采样间隔之间的关系,可以有 3 种模型,其一般形式为

$$H_i(z) = \frac{b_1 z + b_2}{z^i(z - a)}$$

式中 $i=1,2,3$ 。究竟选取哪一种模型,取决于 τ 的值以及 3 种情况下的模型系数。相应的连续模型为

$$G(s) = \frac{K_p e^{-sL}}{Ts + 1}$$

它与受控过程在振荡频率附近的真实频率特性是一致的。

如同一阶系统的情况,通过比较振荡频率处的相位计算值与量测相位的估计值,比较振荡周期的量测值与理论计算值,模型的合理性就可以得到核实。

按照 Ziegler-Nichols 规则设计一个 PI 控制器,由此也可以确定稳态增益。

7. 带有积分环节的继电控制实验

如前所述,对于高阶系统要进行第二次继电控制实验,即在反馈环中插入一个积分环节。对大多数物理过程来说,随着频率的增加,Bode 图的幅频、相频曲线是递减的。因此,当加入积分环节后,振荡频率减小,而振荡幅度就将变大。

积分环节可以插在继电环节的前面或后面。插在前面的方式有不少好处。实验的鲁棒性变得更强;反馈到继电环节的信号可以在计算机内处理,因而可以取很高的值;积分环节对于量测噪声还具有低通滤波器的效果。

为积分环节选取一个正确的初始值是一个重要的问题。为了使周期性振荡尽快地收敛,应将积分环节初始化为积分误差的峰值。但这个值事先并不知道。在实验中,可以根据在没有使用积分环节时量测到的振荡幅度进行估计。

8. 振荡的进一步分析

当继电控制产生的振荡稳定时,要进行类似以前(见本节 3. 继电控制实验)那样的量测。所不同的是,对于振荡曲线形状的量测 e_0, \dots, e_5 ,是在真实的误差信号上形成的。开环 Nyquist 曲线上新的一点也利用以前的方法来求得。还可以求得一个离散时间的模型,用来近似真实受控过程在振荡频率附近的动态特性。近似模型的合理性也用原先的方法

核实。

使模型 $G(s) = K_p e^{-\tau s} / (Ts + 1)$ 满足被量测的稳态 $G(0)$ 以及相位为 -90° 的振荡点, 可以得到第 3 个模型, 即近似表示真实受控过程在频率接近于 0 时的动态特性模型

9. 模型的最终选取

根据已经得到的信息可以选择系统的最终模型: 或者是一个带有延时的一阶系统 $G(s) = K_p e^{-\tau s} / (Ts + 1)$, 或者是一个二阶系统 $G(s) = K_p / (s^2 + a_1 s + a_2)$ 。一阶模型针对具有主导延时的过程以及具有高阶动态特性的过程, 例如 $G(s) = 1 / (Ts + 1)^n$ (n 有较大的值)。二阶模型针对具有二阶动态特性的过程。

最终模型的选取规则如下:

(1) 如果 $\tau > 0.8T_c/4$, 则选取带有延时的一阶模型。这条规则针对具有主导延时和高阶特性的情况。

(2) 如果频率特性曲线在 -90° 点处的量值与在 -180° 点处的量值比较接近, 而且在 -180° 处的频率大约是在 -90° 处的频率的两倍, 则选取一阶模型。这条规则针对具有主导延时情况。

(3) 如果在 -180° 处的频率大约是在 -90° 处的频率的两倍, 则选取一阶模型。这条规则针对具有高阶特性的过程。

(4) 如果带有延时的 3 种一阶模型的参数值比较接近, 则选取一阶模型。这条规则针对具有一阶动态特性以及具有延时过程。

(5) 否则选取二阶模型。

10. 最终模型的计算

根据选定的最终模型的类别, 可以运用不同的方法来计算模型中的参数。如果是根据上述规则(4)选定的一阶模型, 就取模型参数的平均值。在其他情况下, 模型是有歧义的, 因而并不清楚哪一个模型可以信赖。鉴于此, 最终模型的参数就要根据量测到的频率点来计算。如果模型满足 -180° 点和稳态增益, 那末最终参数就取为该模型中的参数与满足 -90° 点和稳态增益的原先得到的模型参数的平均值。如果最终选定的是二阶模型, 可以使模型满足 -180° 和稳态增益, 从而得到模型参数。

11. 控制的最终设计

模型不同, 控制的设计方法也不同。如果最终模型是带有延时的一阶系统, 而且 $L > 1.5T$, 那末就可以采用极点配置的方法。采样间隔可选为 L , 两个离散的极点位于 0.2 处。对于受控过程的输出 y , 可以使用一个数字低通滤波器。这种离散设计方法是有局限性的。例如采样间隔就相当长, 干扰也不能立即检测出来。

如果最终选取定的是二阶模型, 那就可设计一个 PID 控制器, 其闭环特征方程为

$$(s^2 + 2\xi\omega_0 s + \omega_0^2)(s + \omega_0) = 0$$

其中, ξ 取为默认值 0.707, ω_0 可以根据 PID 控制器中 K 的选择来隐式地求得。稳态增益 K 满足

$$K(1 + N) = \frac{\Delta u}{\Delta n}$$

其中, N 为 PID 控制微分部分的低通滤波器的滤波因子。 N 的默认值为 5。

如果最终模型是带有延时的一阶系统,而且 $L \leq 1.5T$,那可用主导极点的设计方法(K. J. Åström 和 T. Hagglund, 1988)。极点配置可采用 PI, PD 或 PID 控制。若为 PI 控制器 $G_R(s) = (K + K_i/s)$,则要使 K 和 K_i 的选择能得到一对共轭的主导极点,即满足

$$1 + \left[K + \frac{K_i}{- \xi \omega_0 \pm j \omega_0 \sqrt{1 - \xi^2}} \right] \times G(- \xi \omega_0 \pm j \omega_0 \sqrt{1 - \xi^2}) = 0$$

其中, ξ 是指定的,而 ω_0 是设计参数。选取不同的 ω_0 来求解上式,可以得到一套控制器参数。对于 PD 控制器,也能得到类似的方程。对于 PID 控制器,极点选择为 $s = - \xi \omega_0 \pm j \omega_0 \sqrt{1 - \xi^2}$ 和 $s = - \omega_0$,这样得到的方程与 PI 情况类似。

如上所述,为 PI, PD, PID 控制器选择不同的设计参数 ω_0 ,将形成 3 套控制器参数。对于每一套参数,可以确定一个最好的控制器。对于 PI 和 PID 控制器, ω_0 的选择要使得积分增益 K_i 在一定的约束条件下取得最大值,约束条件是:所有的控制器参数必须为正值,而且 ω_0 不能选得太大,否则将导致负实轴上的附加闭环极点向原点移动,从而破坏了所选极点的主导位置。对于 PD 控制器, ω_0 的选择要使得比例增益 K 在同样的约束条件下取得最大值。

究竟应该选择什么类型的控制器,这取决于不同的侧重考虑:控制目标是强调对于设置点的快速响应还是强调对于负载干扰的良好的抑制性。如果最重要的是快速设置点响应,那就应该选择 ω_0 具有最大值的控制器。而且如果是 PD 控制器符合这一条件,那还必须保证稳态误差的值是可接受的。否则就应在最好的 PID 与最好的 PI 控制器之间作如下的选择:如果 $0.7\omega_{0_{pid}} > \omega_{0_{pi}}$,那就选择 PID 控制器,否则选择 PI 控制器。此处规定 0.7 作为比较系数的理由是希望尽量不选择比较复杂的 PID 控制器。

在强调对于负载干扰的抑制性时,也首先考虑 PD 控制器。如果稳态误差可以接受,则选择 PD 控制器。否则就要在最好的 PID 与最好的 PI 控制器之间选择:如果 $0.7K_{pid} > k_{pi}$,那就选择 PID 控制器,否则选择 PI 控制器。

4.4.2 自动调整过程的实现

为了用基于知识的控制方法实现上述自动调整过程,应该把总的问题分解为两部分所涉及到的数值算法以及表示为知识源的控制知识。

1. 数值算法

自动调整过程所用到的数值算法有:

(1) PID 控制器,以及线性离散控制器

$$R(q)u(k) = T(q)y_{ref}(k) - S(q)y(k)$$

其中 R, S, T 为前向平移算子 q 的多项式。

(2) 继电控制算法,以及振荡分析器,用于继电控制实验。

(3) 统计算法,用以计算受控过程输出量的平均值、方差、最大值和最小值、控制误差、以及控制信号。

(4) 水平交叉检测算法,用以指示一个信号与某个水平值的交叉,量测这个信号达到水平值的时间。

(5) 二阶滤波器,可经过参数化而成为低通、高通、带通滤波器或者陷波滤波器。

2. 知识源

自动调整过程所用到的知识源可以组织为:

(1) 询问用户知识源,向用户(操作者)询问有关受控过程特性和控制要求说明的一些问题;

(2) 手动控制监督知识源,监视手动控制;

(3) 继电控制监视知识源,对继电控制实验进行初始化和监视工作,完成继电控制实验的初步分析;

(4) 建模知识源,包含对受控过程建模以及对核实模型合理性的有关知识,例如通过频率特性的测试建立模型、离散模型与连续模型之间的转换、最终模型的选取等;

(5) 控制设计知识源,包含设计控制方法的有关知识,例如控制问题的性能描述、控制结构的选择、控制参数的计算等;

(6) 控制监视知识源,负责处理对于不同控制器的手动改变参数的命令;

(7) 解释知识源,对于受控过程 and 控制器有关问题生成解释;

(8) Y—统计知识源,对过程输出和误差进行统计;

(9) U—统计知识源,对控制信号进行统计。

以上知识源中,除了询问用户知识源是采用后向链产生式规则的表示方式,其余知识表示方式都采用前向链产生式规则。每个前向链知识源中包含 5 至 15 条规则。另外,还有一个专门的过程式知识源负责自动调整过程的知识源组合调度。

4.5 专家控制技术的研究课题

专家控制系统要完全做到实用化,还存在着许多有待研究和解决的技术课题,其中有些课题也反映了专家系统技术本身的发展。

4.5.1 实时推理

专家控制系统必须在线地获取动态信息,实时地进行过程控制。比起通常的专家系统,专家控制系统尤其需要研究实时推理问题。

1. 实时推理的特征

非单调推理(non-monotonic reasoning)。人的思维推理过程具有非单调性,随着认识过程的进行,知识并不是单调地积累,而是有所否定、修正,有所调整、更新。专家控制的推理系统运行在一个动态环境中,所获得的传感器信息,以及经过推导得到的事实都在动态地变化。因此,各种数据知识不可能持久,合法性随时间减弱。而且,由于外来事件的影响,合法的数据知识甚至可能变成不合法。这样,为了维持对于环境认识的一致性,推理系统必须能自动撤回或取消失去时效的推理论断。

异步事件的处理(asynchronous events)。动态环境中的事件往往不是同时发生的,而且没有时间上的规律性。因此,推理系统必须具有接收和处理这些异步事件的能力。例如,中断正在进行的重要性程度较低的事件处理过程,转向新的处理过程;或者将新的事件加

入动态知识库,专注于当前最重要的事件,根据原有的事实继续推理。

按时序推理(temporal reasoning)。动态环境中,时间是一个重要的变量。系统必须能够恰当地表示知识的时效性,在不同的时间区间上推理过去、现在或未来的事件;而且还能对事件发生的时间次序进行推理。

带有时间约束的推理(reasoning under time constraints)。出于实时控制的需要,系统的推理过程必须及时,在需要结论的时候推理过程保证能提供结论;而且推理过程必须限时,在限定的时间内推理过程应能提供尽可能好的解。因而推理系统要能对推理过程所需的时间作出估计,要能对推断结论的优劣以及不同的推理策略的优劣作出度量。

并行推理(parallel reasoning)。并行性是指两个或多个事件在同一时间间隔内发生的现象。问题求解任务一般都可以自然地看作是对一系列并发事件同时进行推理的活动组合,在同一推理活动中也往往需要对多个知识因素(例如产生式规则中的前提项)同时进行确认,这些情况都要求系统具有并行推理的能力。因此,推理机制中要解决不同推理活动的同步问题。在条件不具备时要能提供“挂起”某些事件的操作,等待一段固定时间,或者直至另外某个事件发生,然后再进行“解挂”操作。

不确定推理(uncertain reasoning)。动态过程的实时控制中存在着大量的带有不确定性的知识。例如系统中的随机性信息,启发式逻辑中的定性知识,由于传感器数据丢失造成的不完备信息等。因此,推理系统必须解决证据的不确性问题,结论的不确性问题,以及多个规则支持同一事实时的不确性问题。

其他。由于实际的过程控制都处于复杂、多变的运行环境之下,知识库的规模难免庞大,为缩短知识检索和匹配的时间,实时推理需要建立相应的实时操作知识库。由于处理符号信息的人工智能语言一般都不能快速处理数字信息,针对过程控制中存在大量数字计算的特点,实时推理的软件环境还需要把人工智能语言与过程语言结合起来,统一置于一个实时的多任务操作系统之下。最后实时推理总的要求系统的执行速度足够地快,尽管速度快并不等于一定实时,但快速完成各种操作毕竟是实时系统的重要前提。

2. 实时推理方法的研究举例

完全满足实时推理的各方面需要是极为困难的,但研究实践中也提出了把问题解到一定程度的方法。

对于带有时间效应的知识条目,在表示方法上应标注时间信息,以便于推理机按照时序进行推理。一般常用的方法是把知识条目表示成四元组(事实,特性,取值,时间)的形式,其中的“时间”信息标明了“事实”存在的有效期限。例如在 G2 (Gensym, 1987) 系统中,每个测量数据都附有它的有效期限。这种时间期限可以传播到根据测量数据推导所得的事实。测量数据的有效性随着时间衰退,因而需要得到周期性的修改。这种方法可解决数据有效性所导致的非单调推理问题。G2 系统是美国 Gensym 公司的产品,用 Common Lisp 语言编写,具有时序推理、突发事件处理、知识保持、实时调度、内部过程通讯等能力,可用作专家控制系统的开发工具。

为了更有效地调用大量存在的动态时变数据信息,加快推理速度,可采用在推理机前附加一个数据调度器的方法(Murayame T 等, 1989; Hayes-Roth B 和 Washington R, 1989)。推理机将当前推理状态的信息传送给调度器,调度器根据这些信息对输入数据进

行筛选和优先级排序,然后将推理机所需要的优先级最高的数据送给推理机,数据接收处理过程与推理过程并行工作,提高了推理的实时性。

对于并行推理活动,专家系统开发工具 Muse(CCI, 1987)采用了按知识源形式构造系统的方法。对于不同知识源的控制由一个“议程”机构来处理,它允许知识源之间的相互中断。实际上,本章所介绍的专家控制典型结构的原型系统就运用了这种方法。

针对带有时间约束的推理,一种“递进推理”的策略(Wright M L 等,1986)是很有效的。系统的推理机把一个推理过程按其复杂程度分成几个不同的递进层次。顶层的推理演算最不费事,层次愈深推理费用愈大,而且愈能得到比较精确的推理结论。因此,在实时决策的过程中,推理机首先从顶层出发,如果时间约束允许,就递进地进入较深层次的推理,以便求取精确的解;如果时限已到,当前较高层次的决策就被接受,这样在充分运用推理时间的前提下得到了尽可能精确的推理结论。

由 LISP Machine Inc. 研制的实时专家控制系统 PICON(Moore 等,1984)多方面体现了过程控制中的实时推理功能。PICON 最初用于蒸馏塔上的实时报警和咨询处理。它由一个高速数据处理系统和一个专家系统组成,与分布式过程控制系统相连。PICON 选用 Lambda/PLUS LISP 机,其中 LISP 处理机用于专家系统的运行,完成高层的监控和诊断功能;还有 MC68010 处理器用于高速采集数据,及低层的对过程检测和报警进行规则推理,MC68010 处理器中备有实时智能机器环境 RTIME。上述两个处理机可并行运行。PICON 的硬件结构和软件环境相结合,可监视多达 20 000 个过程变量和报警信号,可支持时序推理、并行推理、多系统通讯、用户通讯、面向图形的知识获取等多项功能。

4.5.2 知识获取

专家控制系统是一种基于知识的系统。与专家系统一样,专家控制系统的性能首先取决于它所拥有的领域知识的水平,其中涉及到这些知识的类型、获取方法以及通过学习得到补充和更新等问题。

1. 浅层知识与深层知识的结合

从知识表达的结构层次看,专家知识可分为浅层知识(shallow knowledge)和深层知识(deep knowledge)两大类。

所谓浅层知识,是指表示数据与行为、激励与响应之间的某种经验联系的知识,也可称为经验知识。熟练的操作人员、工程师和领域专家所具有的各种经验估计、启发式逻辑等都属于浅层知识。浅层知识的常用表达形式是产生式规则。基于浅层知识,可以从已知条件和观察数据出发,直接推断出常见局势特征的最强联想或结论,进行以征兆为依据的浅层推理。因此,浅层知识具有启发性强、计算机表达容易、推理过程短、效率高等优点。

所谓深层知识,是指深入表示事物的结构、行为和功能等方面的基本模型的知识,也可称模型知识。各种物理定律、因果关系都属于深层知识,例如说明电子线路元、器件性质的基本原理(第一原理),欧姆定律和基尔霍夫定理(第二原理)等。深层知识一般难以用规则形式表示。基于深层知识,可以在遇到没有经验可循的新的运行局势时,或者遇到意料之外的运行局势时,从事物的基本概念出发分析、推断,即进行以特性为依据的深层推理。这种推理是间接的,实时性差,但是,深层知识具有知识表示的条理性和完备性,问题求解

的灵活性和精确性。

浅层知识和深层知识都是人类专家认知事物的结果,二者的兼备是基于知识的系统发展的需要,能使系统的功能更接近于人类专家的水平。问题在于如何恰当地在知识的表示和运用方面将浅层知识与深层知识进行有机的结合。一般的思想是将知识结构“由上而下”地按层次组织,“由浅(上)到深(下)”地按需要运用,任何一个层次的知识都可以形成一个问题的求解过程。这种“分层递阶”的思想实际上贯穿于智能控制的各个方面,但是具体的构造和处理方法需要针对问题领域设计,在浅层知识与深层知识的优、缺点之间进行折衷。

应当指出,无论浅层知识或是深层知识,都存在知识获取问题,浅层知识出于专家经验,知识工程师往往不容易从专家那里获得足够的资料,而且按运行局势的分类也很复杂。深层知识依赖于事物的模型,在模型带有不确定性的情况下,深层知识的抽象和提炼就非常困难。

2. 专家经验知识的获取

知识获取是指在人工智能和知识工程系统中机器如何获取知识的问题。获取专家系统工作所需要的知识,并将知识构造成可用的形式是研制专家系统的主要“瓶颈”之一。专家控制系统不但需要控制理论的知识,而且还需要控制专家的经验知识,它同样面临着知识获取这一难题。

控制专家的直觉、技巧和启发式逻辑等经验知识可以有两种状态:显知识状态——知识处于能用语言表达或能用文字描述的状态;潜知识状态——知识蕴含在人的行为感觉或控制过程中,而处于一种“只可意会,不可言传”的状态。处于显知识状态或潜知识状态的经验知识可分别简称为显知识或潜知识。例如骑自行车的直觉经验中就包含许多潜知识。显知识可以通过向控制专家直接咨询来获取,经过编辑形成规则等表达形式。而潜知识由于难以通过文字或语言的明确传授来得到,因而成为“瓶颈”问题的关键。

显知识与潜知识之间没有绝对的界限。如果能将潜知识中的因果关系分析清楚,并能用语言、文字加以表述,那末潜知识就可以转化为显知识。对显知识的深入分析也会使其其中又包含许多潜知识。例如一种明确的控制规律,整体上它属于显知识,但其中某些控制参数的设置问题就可能涉及到一些潜知识。随着显知识中包含的潜知识向显知识转化,对问题的认识也就逐步深化。

潜知识获取的困难主要在于一个复杂问题包含着众多的潜知识,而这些潜知识之间往往彼此关联、相互耦合,而且分不清制约因素的主次,无法进行分析描述。

根据对专家经验中显知识和潜知识的上述研究认识,我国的张明廉、沈程智、何卫东等于1992年提出了一种“归约规则法”,探讨经验获取,仿人控制的解决途径。

归约规则法基于人工智能中的问题归约原理(problem reducing principle),即一个复杂问题的求解过程可以这样来进行:把复杂问题逐步化简分解为一系列次复杂问题,直到若干已有解决方案的简章的本原问题(图4.10)。解决本原问题后,依照逆过程进行综合,就可以解决复杂问题本身。

归约规则法把归约原理的思想用于控制经验潜知识的获取,其间需经过3个步骤:(1)对知识的输入输出信息进行形式化描述;(2)对融入复杂问题中的潜知识进行归约

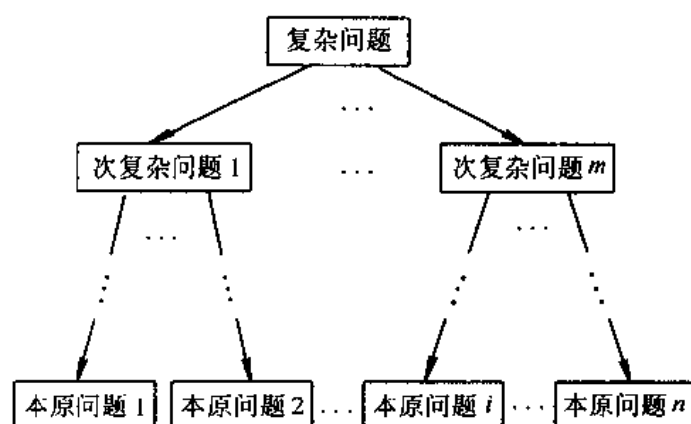


图 4.10 问题归约原理

化简；(3) 对知识的输入输出关系进行因果分析,最后得出某种映射规则。归约规则法获取经验知识并用于求解控制问题的原理如图 4.11 所示。

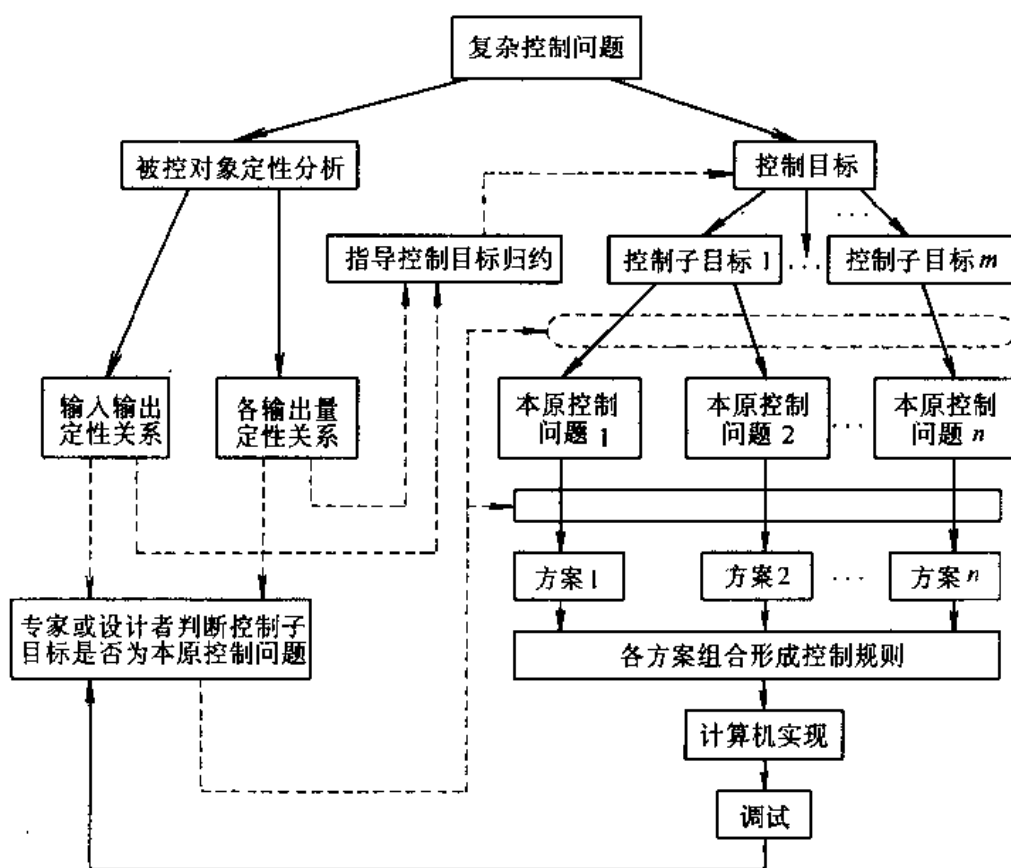


图 4.11 归约规则法原理框图

图 4.11 中,含有大量潜知识的复杂控制问题的求解体现在两个方面。一方面是控制目标的归约分解,直至化简为本原控制问题,这些本原问题中包含的潜知识较少,或者可

以根据控制专家的经验得到解决方案,或者可以通过对输入输出信息进行形式化描述直接得出某种映射关系,从而形成解决方案。另一方面,还需要对复杂控制问题中的被控对象进行定性分析,以便确定实现控制目标的约束。被控对象的定性分析可以分解为输入输出定性关系以及各输出量之间定性关系的因果分析。这两种分析的结果要用来指导控制目标的归约,以便得到相关度较小的各本原问题,使它们的求解方案之间的制约影响尽可能小。这两种分析的结果同时还要提供给控制专家(或系统设计者),用以判断依次分解的控制子目标是否成为本原控制问题。这种判断还要参考控制规则的调试结果。

由上可知,复杂控制问题求解过程的两个方面——被控对象定性分析和控制目标归约是相互作用、相互影响的。定性分析过程把被控对象的定性知识与控制目标相结合,融入归约过程,使一些原来不易表述的控制经验潜知识化为显知识。而随着目标问题归约过程的进行所得到的有关问题的认识也会影响对被控对象的定性分析,即随着矛盾分析的化简,对被控对象有更深入的了解。

总之,归约规则法通过把复杂控制问题化简为本原问题,针对本原问题向控制专家咨询,这为经验知识获取提供了一种有效途径。这种有效性已在倒立摆控制的复杂问题求解中得到了验证。在归约最优性(使本原控制问题相关度最小)、本原问题解决方案与控制规则间关系、归约深度等方面,归约规则法还值得进一步研究。

3. 动态知识获取

专家控制系统是一种基于知识的系统。它得益于所具有的专家知识,提供有效的控制,但在系统出现超出已有知识范围的异常情况时,就可能发生失控。专家控制系统不是专家,由于知识存储容量的限制、知识获取方法的困难等原因,它不可能包含专家的全部知识,因此需要具有某种在线、实时的学习能力,即动态环境中的知识获取能力,实际上,领域专家的知识也是要通过学习来积累的。

专注于学习功能的学习控制系统将在下一章讨论。一般的专家控制系统的学习功能可以通过在线获取信息以及通过人-机交互接受新的知识条目来进行,在系统内部主要体现为知识库的自动更新和扩充,以及根据新的情况自动生成新的规则。这里涉及的实际上也是专家系统技术需要研究的难题。

专家控制系统与一般的专家系统不同,它是一种动态系统。通常情况下,专家控制系统的运行节奏是由它自身决定的,而不为操作人员所左右。因此,在人-机交互的过程中,操作人员必须能够随时“跟上”系统或“超前”于系统,而不是让受控过程停下来等待操作人员的作用。这样,人-机交互就需要在一种中断机制下工作,具备产生于人、机两方面的高级中断能力。其次,工业过程的现场环境中往往有多个操作人员,他们从不同侧面监视系统的运行,而又在同一个人-机接口下工作。因此,必须对“多人-机”的人-机交互提供动态知识的协调、组织能力。另外,过程控制系统常常有大量的检测点,通过在线获取的信息还必须保证知识的实时性和一致性。上述有关动态知识的获取和处理问题在 PICON 系统中得到了一定程度的研究。

在知识库的自动更新和扩充方面,主要涉及到知识的同化和调整等知识库的管理问题。例如,当新获取的知识与知识库里原有的知识发生矛盾、冲突时,就需要根据某种原则进行取舍,或者通过人-机会话进行裁决;当发现新旧知识形成冗余时,就需要通过某种机

制消除冗余;当表明新知识在语义上独立,在形式、规范上一致时,就需要自动地把这些知识加入到知识库中,而且需要首先通过某种方法把它们变为规则等知识表示形式。知识的调整问题包括知识的重新整理、语义精炼、知识的分组和排序等操作。上述有关知识库的管理问题可参见知识工程方面的文献。

总之,专家控制系统的知识获取给实时知识工程技术提出了许多新的课题。

4.5.3 专家控制系统的稳定性分析

稳定性是工程控制系统最基本、最重要的品质属性。简单地说,系统受到扰动而偏离平衡状态时,如在扰动消失后系统能由初始偏差状态回复到平衡状态,则是稳定的,否则是不稳定的。传统的控制理论对稳定性给予极大的重视,认为这是系统的分析与综合设计中的重要原则和方法。基于严格的数学模型,控制理论对于线性系统的稳定性已有了一般的分析方法和判据,而对于非线性系统,只限于特定的系统、特定的方法。

专家控制系统的稳定性分析是一个研究中的难题。这是因为,它涉及的对象具有不确定性、或非线性,它实现的控制基于知识模型,而其中的经验知识具有启发式逻辑、模糊逻辑,因此,专家控制系统本质上是非线性的。控制理论中已有的稳定性分析无法直接用于专家控制系统。

以下列举几种专家控制系统稳定性分析的研究方法。

1. 不稳定指示器

从某种意义上说,专家控制的基本根据就是对系统性能的监控,因此对于不稳定性的及时提早检测尤为重要。但问题是对于不稳定性的出现和发生缺乏明确的定义,因而只能用启发式的方法加以研究。一种“稳定性指标”法(G. T. Russel 和 M. Malcolm, 1976)把平方控制误差的短期均值与平方参考输入的短期均值进行比较,如果前者的增长比后者慢,则认为是不稳定的。但这种方法的决策方程中包含一种“任意”系数,它实际上决定于系统参数。另外一种方法(C. G. Nelser, 1985)通过输出的短期均值与输出的绝对值均值之间的关系导出一种指标,但只是表明了振荡而不是不稳定。

J. Gertler 和 H-S. Chang 指出,不稳定性可通过对象输出量的连续增加幅值是否具有振荡性来描述。对象输出幅值有 4 种类型(图 4.12):增幅振荡——表明不稳定;等幅振荡(系统内存在极限环)——也表明不稳定;衰减振荡——表明过渡过程;恒值输出——表明稳态。

针对上述 4 种类型的输出,可以设计一种不稳定性指示器(J. Gertler 和 H-S. Chang, 1986),对系统是否稳定进行启发式的分析。这种指示器中包括:趋势分析器——分析对象输出幅度;“整流”器——对检测信号进行绝对值运算或平方运算;平滑器——对“检波”信号进行滤波。“整流”器和平滑器的作用是对振荡或非振荡输出给出一种统一的表示。

(1) 输出趋势的获取和分析

如图 4.13 所示,对象输出 $y(t)$ 的趋势可以由两个不同时间区间上的均值 $h_1(t)$ 与 $h_2(t)$ 之差 $h(t)$ 来测量,即

$$h(t) = h_1(t) - h_2(t)$$

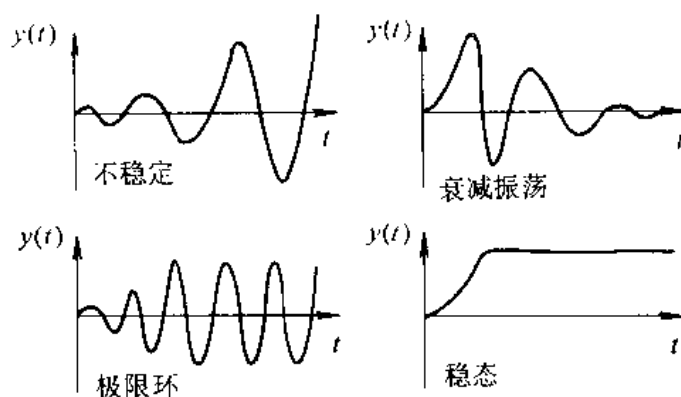


图 4.12 典型的对象输出

$$= \frac{1}{T_1} \int_{t-T_1}^t y(\tau) d\tau - \frac{1}{T_2} \int_{t-T_2}^t y(\tau) d\tau$$

式中时间区间 $T_2 > T_1 > 0$, 称 $h(t)$ 为“趋势量”, 它具有以下两条重要性质。

首先, 趋势量 $h(t)$ 与输出量 $y(t)$ 的导数具有近似的比例关系。设 $y(t)$ 可近似为线性系统的输出, 如果系统不存在多重极点, 那末 $y(t)$ 为指数函数及其与三角函数的组合。对于指数分量 $y(t) = e^{at}$, 就有

$$h(t) = \frac{e^{at}}{\alpha \left[\frac{1}{T_1} (1 - e^{aT_1}) - \frac{1}{T_2} (1 - e^{aT_2}) \right]}$$

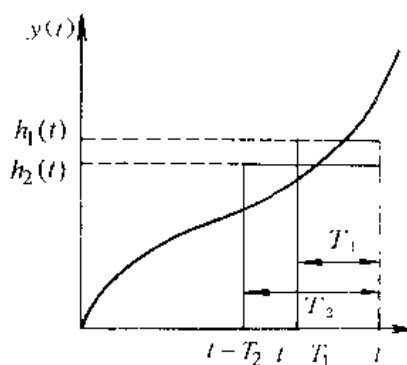


图 4.13 趋势量的原理

式中 α 为 e^{at} 的时间常数, 如果 T_1, T_2 相对于 $\frac{1}{\alpha}$ 较小, 那末指数函数可近似为它的 Taylor 级数的前三项, 即取 $e^a = 1 - aT + \frac{a^2 T^2}{2}$, $T = T_1$ (或 T_2), 这样上式可变为

$$h(t) = \frac{1}{2} (T_2 - T_1) \alpha e^{at}$$

而 αe^{at} 可视为 e^{at} 的导数, 于是

$$h(t) = \frac{1}{2} (T_2 - T_1) \frac{dy(t)}{dt}$$

对指数余弦函数分量 $y(t) = e^{at} \cos(\omega t + \phi)$ 在写成复数形式 $y(t) = \frac{1}{2} [e^{(a+j\omega)t+j\phi} + e^{(a-j\omega)t-j\phi}]$ 后, 仍然可得到同样的结论

$$h(t) = \frac{1}{2} (T_2 - T_1) [\alpha e^{at} \cos(\omega t + \phi) - \omega e^{at} \sin(\omega t + \phi)] = \frac{1}{2} (T_2 - T_1) \frac{dy(t)}{dt}$$

此处, 不但要求 T_1, T_2 相对小于 $\frac{1}{\alpha}$ (包络线时间常数), 而且要小于 $\frac{2\pi}{\omega}$ (余弦函数周期), 即为了得到较好的近似, 与 T_1, T_2 相对应的两个“窗口”只能覆盖输出振荡的周期的很小

一段。

其次,趋势量 $h(t)$ 消除了作用在输出量 $y(t)$ 上的瞬时干扰(例如突然的跳变或脉冲),因而,从不稳定性指示的观点看, $h(t)$ 比观察 $y(t)$ 的导数更合适。瞬时干扰对输出量导数的影响是非常严重的,但对 $h(t)$ 的影响却可得到极大的平滑。例如,对于图 4.14(a)

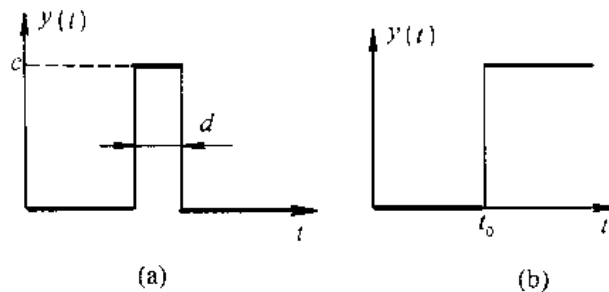


图 4.14 典型的瞬时干扰

所示的脉冲对趋势量的影响为

$$h(t) = \begin{cases} cd(T_2 - T_1)T_1T_2, & \text{若脉冲跨越两个“窗口”} \\ -cd/T_2, & \text{若脉冲仅位于 } T_2 \text{“窗口”} \end{cases}$$

而对于图 4.14(b)所示的阶跃对趋势量的影响为

$$h(t) = \begin{cases} c(t - t_0)(T_2 - T_1)/T_1T_2, & \text{若阶跃跨越两个“窗口”} \\ c[1 - (t - t_0)/T_2], & \text{若阶跃仅位于 } T_2 \text{“窗口”} \end{cases}$$

最大的影响是当阶跃发生于小“窗口”的左极限处,即 $t - t_0 = T_1$,这时有

$$h(t) = c(T_2 - T_1)/T_2$$

趋势量的分析可以用数字滤波器来实现。

(2) “整流”和平滑

“整流”和平滑操作可以先于或者后随趋势分析,如图 4.15 所示。对于图中(a),趋势分析器接收的是经“整流”和平滑的输出量。如果是非振荡的,“整流”不起作用,而平滑将造成某种时延;如果是振荡,则被“整流”为单边波形,而其高次谐波、噪声、干扰就被衰减。这样输入趋势分析器的是与对象输出成比例的信号。上述方式无法检测极限环。对于图中的(b),趋势量要被“整流”和平滑。如果是非振荡的,则造成某种时延;如果是振荡,趋势量还取决于“窗口”长度(相对于振荡周期)。为了有较好的灵敏度,大“窗口”(T_2)不能超过振荡周期的一半。这种方式可以检测极限环,但掩盖了趋势量的方向(导数的正负号)。

2. 智能控制系统的稳定性监控*

这种研究认为,包括专家控制在内的智能控制系统都含有记忆的非线性和变结构控制,而且由于对象和环境的不确定性,难于对系统的稳定性作出离线分析,无法得到系统稳定裕度的解析表达,因此难以研究稳定性特征与系统稳定程度之间的定量关系。但是,系统的输出是控制作用与被控对象内部特征的综合反映,系统不稳定趋势的出现总是以

* 李祖权、秦安松,1993

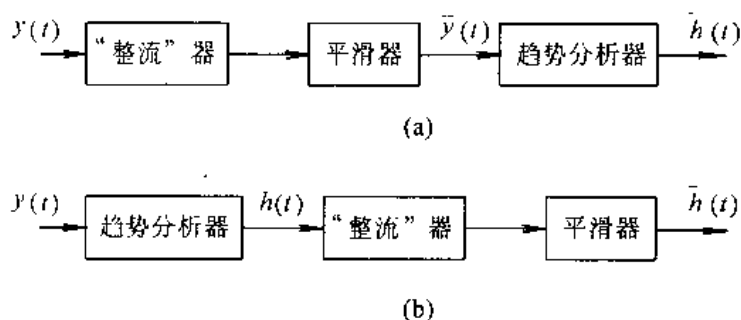


图 4.15 不稳定指示器的组合

一定的特征反映到系统的响应之中,在手动控制中,控制操作者总是能根据输出的不稳定特征,作出在线的预估判断,并以相应的控制策略消除这种不稳定趋势,在线保证系统的稳定性,稳定性监控就是对人的这种控制行为的一种模拟。为此,必须解决两个问题:(1)抽取反映系统稳定性趋势的特征信息,建立系统的不稳定特征模型;(2)基于不稳定特征模型建立相应的监控模态集和推理规则集。

稳定性监控的研究对于动态响应中不稳定趋势特征的识别给出了以下定理:

如果智能控制系统在每一时刻满足:(1)在误差相平面 $e-\dot{e}$ 上,误差相轨迹绕原点运动,或直接收敛于原点;(2)若在闭环系统 $\dot{e} = f(e, t, u)$ 中存在 $t_{2n}, n \in$ 非负整数,使 $\dot{e} = f(e, t_{2n}, u) = 0$,则在区间 $[t_{2n}, t_{2(n+1)}]$ 内,存在唯一的 t_{2n+1} ,使

$$\frac{d}{dt}e(t) = \frac{d}{dt}f(e, t_{2n}, u) = 0$$

且

$$|e(t_{2n+1} - \tau)| \geq |e(t_{2n+1} + \tau)|, \quad \tau \in [t_{2n+1}, t_{2(n+1)}]$$

(3) $\frac{d}{de}\dot{e}(t) < 0$, 那么系统是稳定的。

上述定理是系统稳定的充分条件,当系统输出不满足定理条件时,系统具有不稳定趋势。由于 $\dot{e}(t)$ 在相位上超前 $e(t)$,因而定理的稳定性判断是一种超前的预估判断,这为在线校正系统提供了可能。例如,在系统的零状态阶跃响应中,当 $e(t)$ 接近零时,就可以判断系统是否出现不稳定趋势,而无需等到 $e(t)$ 的第二个极值点出现或 $e(t)$ 单调升降超过允许阈值。因此特征量 $d\dot{e}(t)/de$ 是反映系统稳定性的一个重要特征

3. 全局分析的稳定性指标*

J. Aracil 等认为,任何基于规则控制的系统都是一种非线性系统,这主要是因为控制闭环中包含了逻辑单元。这类系统可以表示为具有非线性控制律 $u = \phi(x)$ 的如图 4.16 所示的闭环结构。

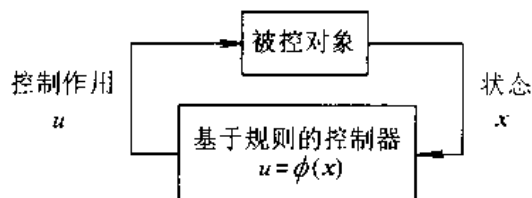


图 4.16 基于规则控制的系统

* J. Aracil 等,1989

系统的状态方程可表示为

$$\dot{x} = f(x) + B\phi(x)$$

上式中 $f(x)$ 为单调递增的对象函数(线性或非线性), 系统以原点为平衡点, 即 $f(0) = 0$, $\phi(0) = 0$ 。利用一种全局矢量场的非线性系统分析方法, 可导出系统在平衡点处的全局稳定性指标和相对稳定性指标。J. Aracil 等人的研究针对模糊控制系统的情况进行了仿真验证, 并认为这种方法可以推广到基于自动调整算法的专家控制系统。

4.6 一种仿人智能控制

广义上, 各种智能控制方法研究的共同点就是使工程控制系统具有某种“仿人”的智能, 即研究人脑的微观或宏观的结构功能, 并把它移植到工程控制系统。事实上, 控制理论本身的研究就是从模仿人的控制行为开始的, 迄今为止世界上最高级的控制器还是人类自身。K. S. Fu 阐述智能控制的研究背景时, 首先提出的是人作为控制器的系统。

早在 1980 年以前, 我国的周其鉴、李祖枢、陈民岫等就提出了仿人智能控制的研究方向。他们认为: 应将对人脑的宏观结构功能模拟与对人控制器的行为功能模拟结合, 仿人智能控制器的研究应从分层递阶智能控制系统的最低层次(运行控制级)着手, 直接对人的控制经验、技巧和各种直觉推理逻辑进行测辨、概括和总结, 编制成各种简单实用、精度高、鲁棒性强、能实时运行的控制算法, 用于实际控制系统。这种仿人智能控制的研究与专家控制密切相关, 本节将介绍它的理论方法要点(李祖枢等, 1990)。

4.6.1 概念和定义

1. 特征模型

智能控制系统的特征模型 Φ 是对系统动态特性的一种定性与定量相结合的描述, 是针对控制问题求解和控制指标的不同要求对系统动态信息空间 \sum 的一种划分。如此划分出的每一区域分别表示系统运动的一种特征动态 ϕ_i , 特征模型为全体特征状态的集合。

$$\Phi = \{\phi_1, \phi_2, \dots, \phi_n\} \quad \phi_i \in \sum$$

在图 4.17(a) 表示的系统动态信息空间 \sum 中, 每一块区域都对应于图 4.17(b) 中系统偏差响应曲线上的一段, 表明系统正处于某种特征运动状态。例如特征状态

$$\phi_1 = \{e \cdot \dot{e} \geq 0 \cap |\dot{e}/e| > \alpha \cap |e| > \delta \cap |\dot{e}| > \delta_2\}$$

就表明系统正处于受扰动作用以较大速度偏离目标值的状态。式中 α, δ_1 和 δ_2 为阈值。

从上式可以看出, 特征状态由一些特征基元的组合描述。设特征基元集为

$$Q = \{q_1, q_2, \dots, q_n\}$$

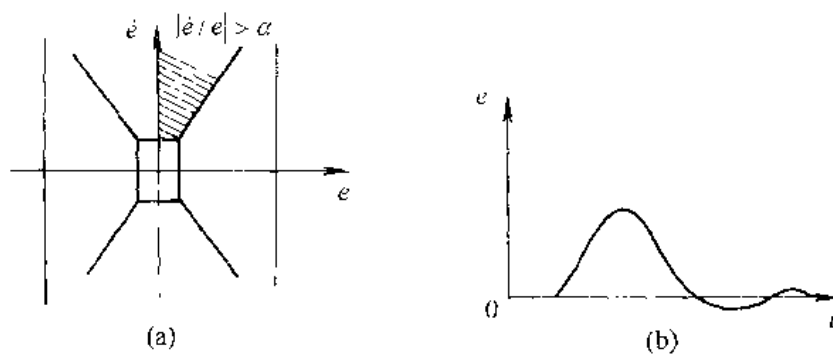
基元 q_i 的常用表示设为:

$$q_1: e \cdot \dot{e} \geq 0; \quad q_2: |\dot{e}/e| > \alpha;$$

$$q_3: |e| < \delta_1; \quad q_4: |e| > M_1;$$

$$q_5: |\dot{e}| < \delta_2; \quad q_6: |\dot{e}| > M_2;$$

$$q_7: e_{m-1} \cdot e_m > 0; \quad q_8: |e_{m-1}/e_m| \geq 1; \dots$$



(a) 一种简单的特征模型 (b) 偏差响应曲线

图 1.17 特征状态的例

其中 $\alpha, \delta_1, \delta_2, M_1$ 和 M_2 均为阈值; e_{m_i} 为误差的第 i 次极值。

若特征模型和特征基元集分别以向量表示为

$$\Phi = (\phi_1 \phi_2 \cdots \phi_n), Q = (q_1 q_2 \cdots q_n)$$

则二者的关系可表示为

$$\Phi = P \odot Q$$

式中 P 为一关系矩阵, 其元素 p_{ij} 可取 -1, 0 或 1 三个值, 分别表示取反, 取零和取正。符号 \odot 表示“与”的矩阵相乘关系。例如

$$\phi_i = [(p_{i1} \cdot q_1) \cap (p_{i2} \cdot q_2) \cap \cdots \cap (p_{im} \cdot q_m)]$$

当除 $p_{i1}=1, p_{i2}=-1$ 之外的 P 的第 i 行元素全为零时, $\phi_i = [e \cdot \dot{e} \geq 0 \cap |\dot{e}/e| \leq \alpha]$ 。

总之, 反映系统运动状态的所有特征信息构成了系统的特征模型, 成为控制器应有的先验知识。

2. 特征辨识

特征辨识是仿人智能控制器依据特征模型 Φ 对采样信息进行在线处理、模式识别, 从而确定系统当前处于什么样的特征状态的过程。

3. 特征记忆

特征记忆是指仿人智能控制器对一些特征信息的记忆, 这些特征信息或者集中地表示了控制器前期决策与控制的效果, 或者集中地反映了控制任务的要求以及被控对象的性质。所记忆的特征信息称为特征记忆量, 其集合记为

$$A = \{\lambda_1, \lambda_2, \cdots, \lambda_r\}, \quad \lambda_i \in \Sigma$$

特征记忆量的常用表示设为

- λ_1 : 误差的第 i 次极值为 e_{m_i} ;
- λ_2 : 控制器前期输出保持值 u_H ;
- λ_3 : 误差的第 i 次过零速度 \dot{e}_{α_i} ;
- λ_4 : 误差极值之间的时间间隔 t_{e_m} ;
-

特征记忆量的引入可使控制器接受的大量信息得到精炼, 消除冗余, 有效地利用控制

器的存储容量。

特征记忆可直接影响控制与校正输出,可作为自校正、自适应和自学习的根据,也可作为系统稳定性监控的依据。

4. 控制(决策)模态

控制(决策)模态是仿人智能控制器的输入信息和特征记忆量与输出信息之间的某种定量或定性的映射关系。控制(决策)模态的集合记为

$$\Psi = \{\psi_1, \psi_2, \dots, \psi_l\}$$

其中,定量映射关系 ψ_i 可表示为

$$\psi_i: u_i = f_i(e, \dot{e}, \lambda, \dots), \quad u_i \in U \quad (\text{输出信息集})$$

定性映射关系 ψ_j 可表示为

$$\psi_j: f_i \rightarrow \text{IF(条件)THEN(操作)}$$

人的控制策略是灵活多变的,不仅因对象而异,而且对同一对象在不同的动态响应状态下或不同的控制要求下也会采取不同的控制策略。这种多变的策略表现为多样的控制(决策)模态,称为多模态控制(决策)。

4.6.2 原理和结构

1. 仿人智能控制的基本原理

在人参与的控制过程中,经验丰富的操作者不是依靠对象的数学模型,而是根据对象的某些定性知识以及自己积累的操作经验进行推理,并且在线确定或变换控制策略。简而言之,人(控制专家)的控制(决策)过程实质上是一种启发式的直觉推理过程。

仿人智能控制方法的基本原理是模仿人的启发式直觉推理逻辑,即通过特征辨识判断系统当前所处的特征状态,确定控制(决策)的策略,进行多模态控制(决策)。

特征辨识和多模态控制实际上是一种具有二次映射关系的信息处理过程。其中的一次映射是特征模型到控制(决策)模态集的映射,即

$$\Omega: \Phi \rightarrow \Psi, \quad \Omega = \{\omega_1, \omega_2, \dots, \omega_i, \dots, \omega_r\}$$

这是一种定性映射,模仿人的启发式直觉推理,可用产生式规则表示

$$\omega_i: \text{if } \phi_i \text{ then } \psi_i$$

再一次映射是控制(决策)模态本身所包含的映射,即

$$\Psi: R \rightarrow U, \quad \Psi = \{\psi_1, \psi_2, \dots, \psi_i, \dots, \psi_r\}$$

式中 R 为控制器输入信息集合 E 与特征记忆量集合 Δ 的并, U 为控制器输出信息的集合。如第 4.6.1 节所述,这一次映射或者表示为定量映射,或者表示为定性映射。

为简化起见,上述二次映射关系可定义为两个三重序元关系。设仿人智能控制过程表示为 IC,则有

$$\text{IC} = (\Phi, \Psi, \Omega), \quad \Psi = (R, U, F)$$

式中 F 表示控制(决策)模态包含的映射关系。

控制模态集 Ψ 也应是仿人智能控制器的先验知识。实际上每一控制模态都可由一些模态基元构成。例如,在运行控制这一层次上,相应的控制模态集中,常用的模态基元表示有

- m_1 : 比例控制模态基元 $K_p e$;
 m_2 : 微分控制模态基元 $K_d \dot{e}$;
 m_3 : 积分控制模态基元 $K_i \int e dt$;
 m_4 : 峰值误差和控制基元模态 $K \sum_{i=1}^n e_{m_i}$;
 m_5 : 保持控制模态基元 u_H ;
 m_6 : 磅-磅控制模态基元 $\pm u_{max}$;
。

设输出信息集 U 也表示输出量的向量, m 为由控制模态基元组成的向量, 则有

$$\Psi; U = Lm$$

式中 L 为关系矩阵, 元素只有 $-1, 0, 1$ 三种, 由此构成的控制模态就可以有

- ϕ_1 : 比例微分加保持 $u = u_H + K_p e + K_d \dot{e}$;
 ϕ_2 : 开环保持 $u = K \sum_{i=1}^n e_{m_i}$;
 ϕ_3 : 磅-磅控制 $u = \pm u_{max}$;
。

2. 仿人智能控制器的结构

仿人智能控制器可表示为一种高阶产生式系统结构, 它由目标级产生式和间接级产生式组成, 具体的结构是分层递阶的, 并遵照层次随“智能增加而精度降低”的原则。较高层解决较低层中的状态描述、操作变更以及规则选择等问题, 间接影响整个控制问题的求解。这种高阶产生式系统结构实际上是一种分层信息处理与决策机构。

图 4.18 表示了一个单元控制器的二阶产生式系统结构。

在图 4.18 中, 运行控制层 MC 是目标级产生式, 直接面对实时控制问题, 构成 0 阶产生式系统; 参数校正层 ST 属间接级产生式, 解决 MC 中控制模态的自校正问题, 对实时控制起间接作用, ST 与 MC 一起构成一阶产生式系统; 任务适应层 TA 也属间接级产生式, 解决 MC、ST 中特征模型、推理规则、控制模态的选择和修改, 及自学习生成的问题, 它更间接地影响实时控制, TA 与 MC 和 ST 一起构成二阶产生式系统。系统的每层结构都有各自的数据库 DB, 规则库 RB, 特征辨识器 CI 和推理机 IE, 层间信息交换通过公共数据库完成。这种紧耦合的并行运行机制便于实现快速的自适应和自学习控制。

图 4.18 所示的单元控制器各层都用二次映射关系来描述, 即

$$IC = (\Phi, \Psi, \Omega) \quad \Psi = (R, U, F)$$

其中

$$\begin{aligned}
 \Phi &= \{\Phi_1, \Phi_2, \Phi_3\}; R = \{R_1, R_2, R_3\}; \Psi = \{\Psi_1, \Psi_2, \Psi_3\}, U = \{U_1, U_2, U_3\}; \\
 \Omega &= \{\Omega_1, \Omega_2, \Omega_3\}, F = \{F_1, F_2, F_3\}.
 \end{aligned}$$

式中各量的下标“1”, “2”, “3”分别相应于 MC, ST, TA 的层次序号(下同)。

在 MC 和 ST 中, 特征辨识器 CI_1 和 CI_2 按照各自的特征模型 Φ_1 和 Φ_2 , 根据给定输入 X , 经滤波器 FM 处理后得到系统输出 Y 及系统输出误差 Y 等在线信息, 判别系统当前

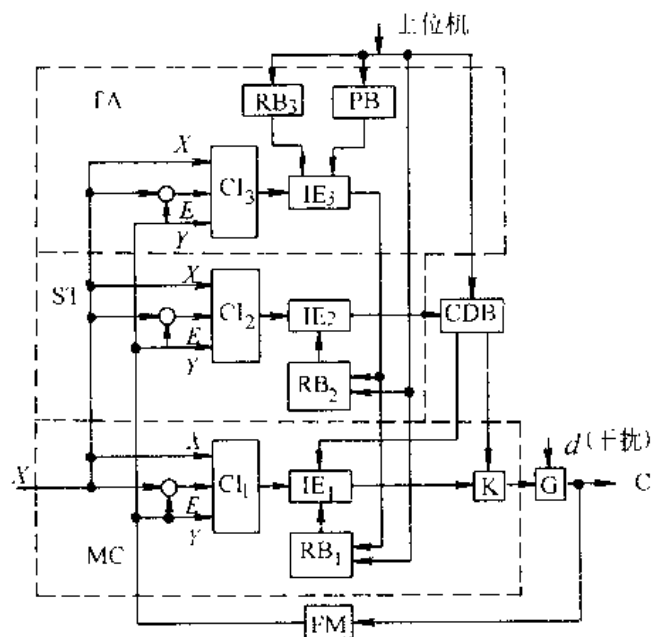


图 4.18 单元控制器

所处的特征运动状态,再通过推理机 IE_1 和 IE_2 ,分别由规则集 Ω_1 和 Ω_2 直接映射到控制模态集 Ψ_1 和参数校正模态集 Ψ_2 ,激励相应的控制模态和参数校正模态,进而完成输入到控制或校正输出的映射。若对象 G 是多变量系统,需经协调器 K 协调后完成控制任务。

在 TA 中,总规则库 RB_3 存放着有关领域控制专家的先验知识,以及实现自学习功能的元知识(Φ_3, Ψ_3, Ω_3)。 PB 是控制性能指标库,存放着预先设置好的各种直观的性能指标(如上升时间,超调量等)和瞬态指标。

TA 的主要功能是任务适应和自学习寻优,即完成(Φ_1, Ψ_1, Ω_1)和(Φ_2, Ψ_2, Ω_2)的自组织、自修正和自生成过程,初投入运行时, TA 首先通过试探控制来积累足够的信息,按照已有的先验知识选择或初始划分特征信息空间,形成初始的($\Phi_{10}, \Psi_{10}, \Omega_{10}$)和($\Phi_{20}, \Psi_{20}, \Omega_{20}$)。然后在控制中完成 MC 和 ST 的任务适应和自学习寻优,其过程可描述为产生式

$$\text{IF } \{\Phi_3, A, P_b\} \text{ THEN } \{\Phi_1, \Psi_1, \Omega_1\} \text{ and } \{\Phi_2, \Psi_2, \Omega_2\}$$

式中 Φ_3 为任务适应或自学习的特征模型, A 为特征记忆集, P_b 为问题求解的目标集。运行中若任务或对象类型发生变化, TA 首先通过变化了的特征模型 Φ_3' 确定新的目标集 P_b' ,并结合控制中形成的新的特征记忆集 A' ,对 MC 和 ST 进行增删修改,这种过程可表示为产生式

$$\text{IF } \Phi_3' \text{ THEN } P_b'$$

$$\text{IF } \{\Phi_3', A', P_b'\} \text{ THEN } \{\Phi_1', \Psi_1', \Omega_1'\} \text{ and } \{\Phi_2', \Psi_2', \Omega_2'\}$$

任务适应和自学习寻优过程结束后, TA 成为一个稳定性监控器。它根据对系统不稳定的特征模型的判别,确定消除不稳定因素的措施。

4.6.3 仿人智能控制的特点

总结上述仿人智能控制方法的原理和结构,可归纳为如下特点:

- (1) 分层的信息处理和决策机构;
- (2) 在线的特征辨识和特征记忆;
- (3) 开、闭环控制,正、负反馈控制和定性决策与定量控制结合的多模态控制;
- (4) 启发式直觉推理逻辑的运用。

具体地,启发式直觉推理逻辑可由人工智能的产生式规则描述;在线特征辨识和特征记忆依据特征模型进行,而特征模型的建立与模式识别、知识获取和表示的技术密切相关;多模态控制建立在经典控制理论上,各种控制模态的设计充分利用了控制理论的成果;分层递阶的信息处理和决策机构可依靠计算机硬件和软件的发展得到支持。

仿人智能控制的方法还表明,智能控制的研究目标不是被控对象,而是控制器如何对控制专家的经验行为和知识结构的模仿;辨识和建模的目标不是对象的数学模型,而是整个系统的动态特征模型和控制器定性定量结合的知识模型。

对于本节介绍的仿人智能控制方法,重庆大学智能控制研究室已对一些控制对象的控制器进行了设计和仿真,并对基于特征辨识和特征记忆的多模态控制(决策)所导致的新问题进行了研究,例如:多性能指标优化,控制器的能控性,推理的可达性和可信度,稳定性监控,以及系统智商的量测等(李祖枢,1991)。

参 考 文 献

- [1] Barr A, Feigenbaum E A. Handbook of Artificial Intelligence, Vol. 1. William Kaufmann, Inc., 1981
- [2] Barr A, Feigenbaum E A. Handbook of Artificial Intelligence, Vol. 2. William Kaufmann, Inc., 1982
- [3] Cohen P R, Feigenbaum E A. Handbook of Artificial Intelligence, Vol. 3. William Kaufmann, Inc., 1982
- [4] 唐纳德 A 沃特曼(美). 周洪泽, 谢学堂, 李玉峰译. 专家系统指南. 东北林业大学出版社, 1989
- [5] 林尧瑞, 马少平. 人工智能导论. 清华大学出版社, 1989
- [6] Moor R L, *et al.* A Real-time Expert System for Process Control, Proc. First Conf. on Artificial Intelligence Applications, 1984
- [7] Gallanti M, Guida G. Intelligence Decision Aids for Process Environments: An Expert System Approach. NATO ASI Series, Vol. F21, Edited by Hollnagel E, *et al.* 1986
- [8] Trankle T L, Markosian L Z. An Expert System for Control System Design. Proc. IEE Int. Conf. Control, Cambridge, UK, 1985
- [9] Bristol E H. Pattern Recognition: An Alternative to Parameter Identification in

Adaptive Control. Automatica, 1977

- [10] Porter B, *et al.* Real-time Expert Tuners for PI Controllers. IEEE Proc. 1987
- [11] 郭晨. 智能控制器与锅炉专家控制系统的研究. 大连海事大学博士学位论文, 1991
- [12] 王建华, 刘鸿强, 潘日芳. 专家控制系统在精馏塔控制中的应用. CAAI'87. 1987
- [13] 周其鉴, 李祖枢, 陈民铀. 智能控制及其展望. 信息与控制, 1987
- [14] Åström K J. Implementation of an Auto-tuner Using Expert System Ideas. Report CODEN: LUTFD2/TFRT-7256. Lund, Institute of Technology, Lund, Sweden, 1983
- [15] Åström K J, Anton J J, Årizen K -E. Expert Control. Automatica, 1986
- [16] Årizen K -E. An Architecture for Expert System Based Feedback Control. Automatica, 1989
- [17] Erman L D, *et al.* The Hearsay- I Speech Understanding Knowledge to Resolve Uncertainty. Computing Surveys, 1980
- [18] Cannon H I. Flavors: a Non-hierarchical Approach to Object-oriented Programming. 1982
- [19] Allen E M. YAPS: Yet Another Production System. TR-1146, Department of Computer Science, University of Maryland, 1983
- [20] Åström K J, Hägglund T. Automatic Tuning of Simple Regulators with Specifications on Phase and Amplitude Margins. Automatica, 1984
- [21] Åström K J, Hägglund T. A New Auto-tuning Design. Technical Report TFRT-7368, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1987
- [22] Chantler M J. Real-time Aspects of Expert System in Process Control. Colloquium on Expert System in Process Control, IEEE, Savoy Place, London, UK, 1988
- [23] 关守平, 柴天佑. 实时专家系统的现状及展望. 第二届全国智能控制研讨会论文集, 1994
- [24] Gensym. G2 User's Manual. Gensym Corp. Cambridge, Massachusetts, 1987
- [25] Murayama T, *et al.* An Inference Mechanism Suited for Real-time Control. 2nd Int. Conf. Application of AI&ES, 1989
- [26] CCL. Muse Product Description. Cambridge Consultants Limited, Cambridge, UK, 1987
- [27] Wright M L, *et al.* An Expert System for Real-time Control. IEEE Software, 1986
- [28] 孙昌龄, 王京平. 专家系统的浅层知识与深层知识. 第一届全球华人智能控制与智能自动化大会论文集, 1993
- [29] 张明廉, 何卫东, 沈程智. 归约规则法仿人控制. 第一届全球华人智能控制与智能自动化大会论文集, 1993

- [30] Gertler J, Chang H-S. An Instability Indicator for Expert Control. IEEE Control System Magazine, 1986
- [31] Russel G T, Malcolm M. Stability Index. Proc. IEEE, 1976
- [32] Nesler C G. American Control Conf. Boston, MA, 1985
- [33] 李祖枢, 秦安松. 智能控制系统的稳定性监控. 第一届全球华人智能控制与智能自动化大会论文集, 1993
- [34] Aracil J, *et al.* Stability Indices for the Global Analysis of Expert Control Systems. IEEE Trans. on SMC, 1989
- [35] 李祖枢, 徐鸣, 周其鉴. 一种新型的仿人智能控制器(SHIC). 自动化学报, 1990
- [36] 李祖枢. 智能控制理论研究. 信息与控制, 1991

第5章 学习控制

学习是人类获取知识的主要形式,是人类具有智能的显著标志,是人类提高智能水平的基本途径。因此,学习也是智能控制的重要属性。这里主要指自学习,即自动获取知识、积累经验、不断更新和扩充知识,改善知识性能。

学习控制是智能控制的一个重要的研究分支。K. S. Fu 把学习控制与智能控制相提并论,从发展学习控制的角度首先提出智能控制的概念(K. S. Fu, 1971)。他推崇在控制问题中引入拟人的自学习功能,研究各种机器系统可以实现的学习机制。

学习控制与自适应控制一样,是传统控制技术发展的高级形态,但随着智能控制的兴起和发展,已被看作是脱离传统范畴的新技术、新方法,可形成一类独立的智能控制系统。

学习的概念含义丰富而又难于确切界定,因而学习控制的研究目前也缺乏系统的理论表达。

5.1 概 述

5.1.1 学习控制问题的提出

智能控制的任务也可以这样来表达:要使闭环控制系统在相当广泛的运行条件范围内,在相当广泛的运行事件范围内,保持系统的完善功能和期望性能。而实现这一任务的困难是,受控对象和系统的性能目标具有一定的复杂性和不确定性。例如,受控对象通常存在非线性和时变性;尤其是受控对象的动力学特性往往建模不良,也可能是设计者主观上未能完整表达所致,或者是客观上无法得到对象的合适模型;其它还有多输入多输出、高阶结构、复杂的性能目标函数、运行条件有约束、测量不完全、部件发生故障等因素。

学习控制的作用是为了解决主要由于对象的非线性和系统建模不良所造成的不确定性问题,即努力降低这种缺乏必要的先验知识给系统控制带来的困难。

K. S. Fu 指出,在设计一个工程控制系统时,如果受控对象或过程的先验知识全部是已知的,而且能确定地描述,那么从合适的常规控制到最优控制的各种方法都可利用,求得满意的控制性能;如果受控对象或过程的先验知识是全部地或者局部地已知,但只能得到统计的描述(例如概率分布、密度函数等),那么就要利用随机设计或统计设计技术来解决控制问题;然而如果受控对象或过程的先验知识是全部未知的或者局部未知的,这时就谈不上完整的建模,传统的优化控制设计方法就无法进行,甚至常规控制方法也不能简单地使用。

对于先验知识未知的情况,可以采取两种不同的解决方法。一种是忽略未知部分的先验知识,或者对这些知识预先猜测而把它们视同已知,这样就可以基于知识“已知”来设计控制,采取保守的控制原则,安于低效和次优的结果;另一种方法是,在运行过程中对未知

信息进行估计,基于估计信息采用优化控制方法,如果这种估计能逐渐逼近未知信息的真实情况,那么就可与已知全部先验知识一样,得到满意的优化控制性能。

由于对未知信息的估计逐步改善而导致控制性能的逐步改善,这就是学习控制。

应当指出,学习控制所面临的系统特性在一定环境条件下实际上是确定的,而不是不确定的,只是在于事先并不清楚,但随着过程的进展可以设法弄清楚。换言之,不可知的信息无法学习,学习是对事先未知的规律性知识的学习。

5.1.2 学习控制的表述

学习这一概念在日常生活中使用极其广泛,非常通俗,目前没有公认的统一定义。人们从不同的学科角度、不同的理解层次来表述学习、学习控制和学习控制系统。

Wiener 从物种随时间变异的现象给出了学习的最一般的定义(Wiener,1965):具有生存能力的动物,是那些在它的个体的一生中,能被它所经历的环境所改造的动物;一个能繁殖的动物,至少能够产生与它自己大略相似的动物,虽然这种动物不会完全相似到随时间的推移而不再发生变化的程度;如果这种变化是自我可遗传的,则就有了一种能受自然选择的原料;如果这种变化以某种行为形式显现出来,则只要该行为不是有害的,则这种变化就会一代代地继续下去。这种从一代到下一代的变化形式就称为种族学习或系统发育学习,而特定个体中发生的行为变化或行为学习,则称为个体发育学习。

Shannon 对于学习的定义考虑了所有可能的个体发育学习中的一个子集(R. M. Glorioso,1975):假定一个有机体或一台机器处于某类环境中,或者同该类环境有联系;而且假定存在一个对该环境是“成功”的量度或“自适应”的量度;进一步假定,这种量度在时间上是比较局部的量度,即人们能在比该有机体生命期为短的时间内,测定这个成功的量度。如果对于所考虑的这类环境,这种局部的成功量度有随时间而改善的趋向,那么我们可以说,相对于所选择的成功量度,该有机体或机器正在为适应这类环境而学习着。

Osgood 从生理学角度表述了学习的定义(R. M. Glorioso,1975):所谓学习是指在同类特征的重复情境中,有机体个体靠自己的自适应性,使自己的行为和竞争反应中的选择不断地改变、增强。这类选择变异是由个体的经验形成的。

上述定义对于学习本质的认识,有助于我们在工程控制系统中研究开发学习功能。

K. S. Fu 详细阐述了学习控制的意义,指出学习控制器的任务是在系统运行中估计未知的信息并基于这种估计的信息确定最优控制,逐步改进系统的性能(K. S. Fu,1970)。

Y. Z. Tsypkin 把系统中的学习一词理解为一种过程,通过重复各输入信号并从外部校正该系统,从而使系统对于特定的输入信号具有特定的响应。而自学习就是不具有外来校正的学习,或即不具惩罚和奖励的学习(Y. Z. Tsypkin,1966)。

G. N. Saridis 认为,如果一个系统能对一个过程或其环境的未知特征所固有的信息进行学习,并将得到的经验用于进一步估计、分类、决策或控制,从而使系统的品质得到改善,那就称此系统为学习系统。而学习系统将其得到的学习信息用于控制具有未知特征的过程,就成为学习控制系统(G. N. Saridis,1977)。

综合上述各种解释,有一种比较完整、规范的学习控制表述是值得推荐的:

一个学习控制系统是具有这样一种能力的系统,它能通过与控制对象和环境的闭环

交互作用,根据过去获得的经验信息,逐步改进系统自身的未来性能(L. Walter 和 J. A. Farrell,1992)。

这种表述说明了学习控制的一般特点:

- (1) 有一定的自主性。学习控制系统的性能是自我改进的;
- (2) 是一种动态过程。学习控制系统的性能随时间而变,性能的改进在与外界反复作用的过程中进行;
- (3) 有记忆功能。学习控制系统需要积累经验,用以改进其性能;
- (4) 有性能反馈。学习控制系统需要明确它的当前性能与某个目标性能之间的差距,施加改进操作。

5.1.3 学习控制和自适应控制

自适应控制也是一种解决系统不确定性问题的方法。简单地说,自适应控制系统能适应系统的环境条件或对象特性的变化,它根据对可达的输入输出量的在线观测信息,自动校正或调整控制器的参数和性能,使系统保持在最优的或满意的工作状态。

自适应控制与学习控制处理不确定性问题都是基于在线的参数调整算法,都要使用与环境、对象闭环交互得到的实验信息。但二者对于不确定性问题处理的程度、着重点和目的存在着重要的区别。

自适应控制着眼于瞬时观点,它的目标是针对干扰和动态特性随时间变化的情况,维持某种期望的闭环性能。实际中当系统的工作点出现变化时,动态特性随时间变化的现象可能是由于非线性引起的。大多数自适应控制的规律一般都不能在很广的范围内把控制作用表示为当前运行状态的函数,因此它的控制器是缺乏记忆的,即使是时不变的非线性特性,而且是以以前经历过的特性,它也要重新适应,补偿所有的瞬时变化。进一步,甚至在理想的环境下,每当对象特性变化时,自适应过程的动态特性还会引起期望控制作用的滞后。这就意味着不合适的控制作用也可能持续一段时间。对于时不变的非线性对象特性,不合适的控制作用导致的不必要的过渡过程,从而造成控制性能的下降。而对于线性的动态特性,如果变化非常快,仅依靠自适应作用也可能无法维持期望的控制性能。

相比之下,学习控制要求把过去的经验与过去的控制局势相联系,能针对一定的控制局势来调用适当的经验。学习控制强调记忆,而且记忆的是控制作用表示为运行状态的函数的经验信息。因此,学习控制对于那些单纯依赖于运行状态的对象特性变化具有较快的反应。这种情况典型地表现为非线性特性。

从智能控制的观点看,适应过程与学习过程各具特色,功能互补。自适应过程适用于缓慢的时变特性以及新型的控制局势,而对于非线性严重的问题则往往失效;学习控制适合于建模不良的非线性特性,但不宜用于时变动态特性。为此,有一种看法主张控制系统实际上需要三个子系统组成:一个先验的补偿器(常规反馈环),一个自适应环,一个学习环(J. Sklansky,1966)。

5.1.4 学习控制的研究状况和分类

工程上对于学习的研究起源于人工智能中对学习机制的模拟。一条途径是基于人脑

结构模型来模拟人的形象思维。40年代初,McCulloch 和 Pitts 就提出了一种最基本的神经元突触模型。50多年来,已有数百种神经元模型和神经网络模型被发表。这些学习模型具有联想和分布记忆的特征,与非线性动力学关系密切,导致了非线性问题的学习控制的发展。另一条途径是基于人脑的外部功能来模拟人的逻辑思维。50年代末 Samuel 研制了能与人对奕而且能积累经验的跳棋程序。60年代 Feigenbaum 的语言学习模型表明从参数学习到概念学习的发展。70年代中 Buchanan 和 Mitchell 的 Meta-DENDRAL 系统等研究表明从孤立概念的符号学习到知识基系统的结构学习。80年代以来,示例式、观察式、发现式、类比式等多种学习机制被深入研究,一些工具式学习系统可供应用。以上阶段的研究形成了“机器学习”的人工智能学科分支,它以知识为中心,综合应用知识的表达、存储、推理等技术,是自动知识获取的重要手段。

人工智能对于学习的研究有力地推动了学习控制理论的发展。60年代以来,学习控制的研究方向主要有3类。

1. 基于模式识别的学习控制

这一方向主要起源于人工神经网络的研究,采用的方法基本上是模式识别,着重于参数的自学习控制。

K. S. Narendra 等最先研究了基于性能反馈进行校正的方法(1962年);其后由 F. W. Smith 提出了一种利用自适应模式识别技术的开关控制方法(1964年);F. B. Smith 研究了一种可训练飞行控制系统(1964年),A. R. Bute 推出学习 Bang-Bang 调节器(1964年);A. M. Mendel 等进一步将可训练阈值逻辑(模式分类器)方法应用于控制系统(1968年)。

M. D. Waltz 和 K. S. Fu 将线性再励技术引入学习控制系统,被认为是在控制系统中最早应用了人工智能启发式方法。这类研究是基于模式识别的学习控制的另一个思路。J. M. Mendel 将根据这类方法研究了一个卫星的精确姿态控制问题(1966年)。

K. S. Fu 还首先提出了利用 Bayes 学习估计的方法(1965年)。对于这类自学习控制系统,Y. Z. Tsypkin 等还研究了随机逼近方法,并利用随机自动机构成学习系统的模型,J. S. Riordo 讨论了 Markov 学习模型(1969年)。

总之,随着基于模式识别的参数自学习控制方法的发展,出现了利用模式分类器、再励学习、Bayes 学习、随机逼近、随机自动机、模糊自动机和语义学方法的各种学习控制系统。

2. 基于迭代和重复的学习控制

这一方向主要针对在一定周期内作重复运行的系统,它不但与传统的控制理论相联系,而且可导出易于工程实现的简单的学习控制规律。

这类方法最早见之于日本学者内山的一篇有关机器人控制的论文(1978年)。其后,井上和中野等从频域角度将其发展为重复自学习控制(1980年);有本、川村和宫崎等又将内山的初步研究结果理论化为时域的迭代自学习控制。自此,这类方法主要从时域、频域两方面开始得到独立的研究和发展。

主要在时域中发展的迭代自学习控制应用较广,成果较多。有本等人研究了迭代自学习控制与逆系统,有界实性、灵敏度和最优调节等问题的关系,深刻地指出这种自学习过

程实质上是逼近逆系统的过程。一些研究者随后又提出了多变量系统的最优迭代自学习控制、离散时间系统的迭代自学习控制,自适应迭代自学习控制以及非线性系统的迭代自学习控制等方法。

主要在频域中发展的重复自学习控制只适用于有界连续周期性期望输出的精确伺服跟踪问题,应用面较窄,研究不够深入。这类方法实际上是围绕稳定条件的逐步放宽和学习控制系统的综合这两方面问题展开的,有关在多变量系统、离散时间系统中的运用问题也得到了研究。

迭代与重复自学习控制分别在时域、频域中研究,但基本思想是一致的,都是基于系统不变性的假设、基于记忆系统的间断的重复训练过程。鉴于此,我国的邓志东将这两种方法统一起来,提出了一种异步自学习控制理论,建立了包含更多新内容的体系和框架(算子描述、 L_2 稳定与渐近稳定,系统周期不变性的约束等)。有关方法已针对三轴运动模拟台正弦振动失真的补偿问题进行了仿真实验(邓志东,1991 年)。

3. 联结主义学习控制

主要基于人工神经网络机制的联结主义是人工智能学科领域中近年来蓬勃发展的大学派。联结主义与学习控制的结合已被认为是一种新型的学习控制方法,它与基于知识推理的符号主义学习方法(机器学习)相比,更具有有效性。

W. L. Baker 和 J. A. Farrell 在 1990 年前后提出了联结主义学习控制系统(connectionist learning control systems)的概念,论述了它的理论方法。这种方法把控制系统看作是从对象输出和控制目标到控制作用的映射,学习就是一种自动地综合多变量函数映射的过程,而学习过程的根据则是某种优化原则以及在运行中逐步积累的经验信息,学习过程的实现是通过系统参数和结构的选择调整完成的。

联结主义的学习机制源于生物学和行为科学,在神经网络等方面的研究中也得到了大量实践。本章中有关内容将侧重介绍 W. L. Baker 和 J. A. Farrell 研究的理论方法要点。

5.2 基于模式识别的学习控制

基于模式识别的学习控制方法的基本思想是,针对先验知识不完全的对象和环境,将控制局势进行分类,确定这种分类的决策,根据不同的决策切换控制作用的选择,通过对控制器性能估计来引导学习过程,从而使系统总的性能逐步改善。

5.2.1 学习控制系统的一般形式

J. Sklansky 认为,学习控制系统是具有三个反馈环的层次结构。底层是简单反馈环,包括一个补偿器,它提供控制作用;中间层是自适应环,包括一个模式识别器,它对补偿器进行调整,以响应对象动态特性变化的估计;高层是学习环,包括一个“教师”(一种控制器),它对模式识别器进行训练,以作出最优或近似最优的识别。这种学习控制系统的原理框图如图 5.1 所示(J. Sklansky,1966)。

图 5.1 中,补偿器由多路开关和控制作用的并行单元组成, G_c 的选择由模式识别器

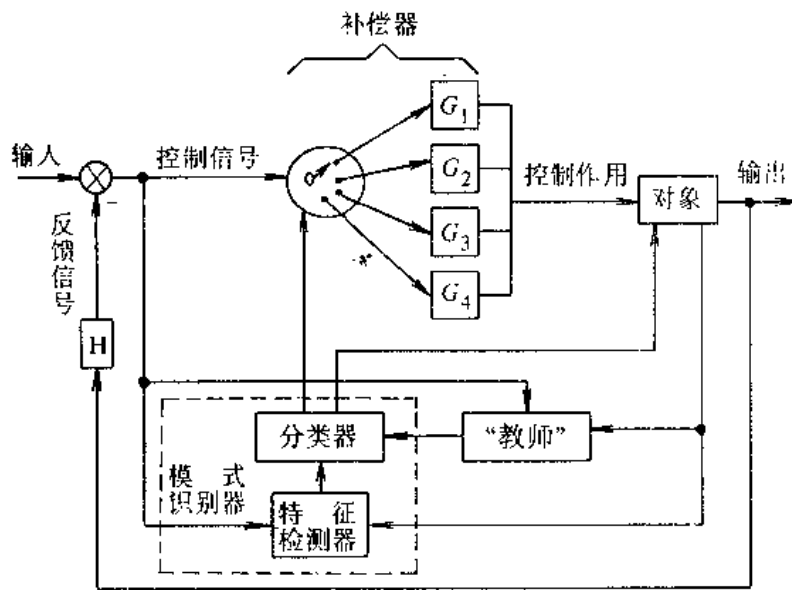


图 5.1 学习控制器的原理框图

的结果信号来确定。模式识别器中的特征检测器敏感对象的动态特性变化,将这些变化转换为—组特征(动态特性参数的估计、状态变量的估计等)。分类器把每一组特征与一个模式类别相联系,这种联系将为 G_i 的选择提供激发信号,也可用来按照某种预定的规则直接激发对受控对象的调节。“教师”监视系统的性能,并调整模式类别在特征空间中的界面,体现学习在控制中的作用。“教师”送往模式识别器的调整作用是一种再励信号,它根据计算所得的性能指标对分类器进行“奖励”或“惩罚”。

在这种学习控制系统中,如果对象的参数在稳定范围内变化,而且外部干扰统计上也是稳态的,那么仅有简单的反馈环就足够了。如对象参数变化剧烈,出现不稳定的干扰,那么借助于模式识别器进行参数估计,启用自适应控制,就能使问题得到缓解。但是在大多数情况下,对象变化和环境干扰的统计特性是未知的,模式识别器并不可能事先得到充分的设计。这样学习环就提供一种“在线”设计模式识别器的能力,整个系统中同时存在学习和控制的作用。

K. S. Fu 更一般地把学习控制系统表示成图 5.2 的形式。

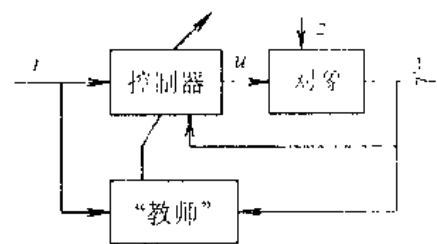


图 5.2 学习控制系统

如图 5.2 所示,受到环境干扰的对象特性设为未知或不全已知,控制器将针对某个最优控制律来学习(估计)所需要的未知信息,如果学得的信息收敛于真实信息,控制器也将逐步地逼近最优控制。“教师”的作用是评估控制器的性能,从而引导控制器完成的学习过程,使得系统总的性能逐渐改进。这里的“教师”具有外部监督的功能。如果具有这种监督,则称为“有监督的学习”,或称“训练”,或称“离线学习”;如果没有(或不需耍)这种监督,则称为“无监督的学习”或称“在线学习”。有监督的学习过程中通常确切地知道期望的系统

输出或期望的最优控制作用,控制器可以据以修改控制策略或控制参数从而改进系统性能。在无监督的学习过程中,或者需要考虑所有可能的答案(例如后面将介绍的 Bayes 学习中的混合密度法),或者利用性能测量的方法来引导学习过程(即所谓的性能反馈法)。

在图 5.2 所示形式的学习控制系统中,学习到的信息同样被看作是控制器的经验,被用于遇到类似控制局势时对控制品质加以改进。因此,对控制局势的分类就成为基于模式识别的学习控制器的一项主要功能。

5.2.2 模式分类

模式分类在学习控制问题中被用于区分不同的控制局势类别。

假设控制局势的未知模式可表示为一组测量值或观察值 x_1, x_2, \dots, x_k , 这 k 个值称为特征。特征可表示为 k 维向量 $\mathbf{x} = (x_1, x_2, \dots, x_k)^T$, 称为特征向量, 相应的向量空间 Ω_x 称为特征向量空间。若控制局势可能有 m 个模式类 $\omega_1, \omega_2, \dots, \omega_m$, 那么模式分类就是对给定的特征向量 \mathbf{x} 指定一种正确的类别隶属关系, 即对特征向量 \mathbf{x} 的分类进行决策。模式分类的操作就是将 k 维特征空间 Ω_x 划分为 m 个互斥子区域的过程。特征空间的这种聚集同类特征向量的子区域称为类区或决策空间, 而分割各类区的界面称为决策面。模式分类确定了从 Ω_x 空间到决策空间的映射。决策面可以用解析的判别函数来表示, 每一个模式类 ω_i 都有一个判别函数 $d_i(\mathbf{x})$ 与之相关联 ($i=1, 2, \dots, m$), 即: 若特征向量 \mathbf{x} 属于模式类 ω_i , 则有

$$d_i(\mathbf{x}) > d_j(\mathbf{x}), \quad \forall j \neq i$$

于是, 模式类 ω_i 与 ω_j 之间的决策面可表示为方程

$$d_i(\mathbf{x}) - d_j(\mathbf{x}) = 0$$

以下介绍几种重要的判别函数。

1. 线性判别函数

判别函数可以选择为特征 x_1, x_2, \dots, x_k 的线性函数, 即

$$d_i(\mathbf{x}) = \sum_{r=1}^k w_{ir} x_r + w_{i,k+1}, \quad i = 1, 2, \dots, m$$

式中 w_{ir} 为相应的阈值权。决策面方程为

$$f(\mathbf{x}) = d_i(\mathbf{x}) - d_j(\mathbf{x}) = \sum_{r=1}^k (w_{ir} - w_{jr}) x_r + (w_{i,k+1} - w_{j,k+1}) = 0$$

这种线性的决策面即为 Ω_x 中的超平面。若令

$$w_r = w_{ir} - w_{jr}, \quad r = 1, 2, \dots, k+1$$

则上式变为

$$\sum_{r=1}^k w_r x_r + w_{k+1} = 0$$

根据上式, 对于 $m=2$ 就可以很容易地借助于阈值逻辑单元实现一个 2-类线性分类器。如图 5.3 所示, 若 \mathbf{x} 属于 ω_1 , 分类器的输出则为 +1, 这是因为

$$d_1(\mathbf{x}) - d_2(\mathbf{x}) = \sum_{r=1}^k w_r x_r + w_{k+1} > 0$$

则若 x 属于 ω_2 , 分类器的输出则为 -1 , 因为

$$d_1(x) - d_2(x) = \sum_{r=1}^k w_r x_r + w_{k+1} < 0$$

仿此, 对于 $m > 2$, 可把若干个阈值逻辑单元并联在一起形成一个 m -类线性分类器 (图 5.4), 其输出值为 $+1$ 与 -1 的组合, 用于区分 m 类不同的模式。

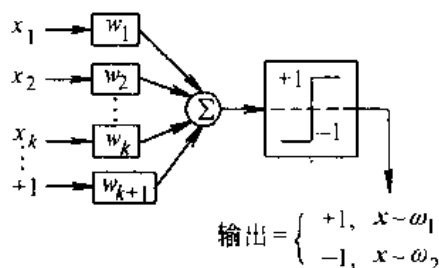


图 5.3 2-类线性分类器

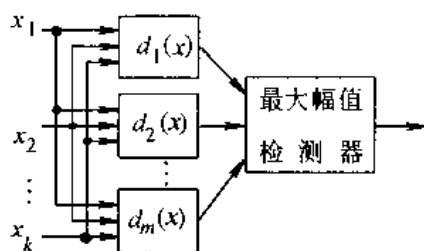


图 5.4 m -类线性分类器

2. 多项式判别函数

判别函数可以选择为特征 x_1, x_2, \dots, x_k 的 n 阶 ($n > 1$) 多项式。当 $n=2$ 时, 有

$$d_i(x) = \sum_{r=1}^k w_r x_r^2 + \sum_{r=1}^{k-1} \sum_{q=r+1}^k w_{rq} x_r x_q + \sum_{r=1}^k w_r x_r + w_{N-1}$$

式中, $N = k + k(k-1)/2 + k = k(k+3)/2$ 。令 A 为元素是 a_{ij} 的矩阵, 其中 $a_{ij} = w_{ij}$, $j = 0, 1, 2, \dots, k$, 而

$$a_{jq} = \frac{1}{2} w_{jq}, \quad j, q = 1, 2, \dots, k, \quad j \neq q$$

令 B 为元素是 $b_j = w_j$ ($j = 1, 2, \dots, k$) 的行向量, 则多项式判别函数可表示为

$$d_i(x) = x^T A x - x^T B + C$$

式中 $C = w_{N+1}$ 。这样 ω_1 与 ω_2 之间的决策面一般为超双曲面。特殊情况下则为超球面或超椭球面。

3. 统计判别函数

如果考虑特征值中混杂噪声的情况, 特征向量 x 则为一随机向量。这时可选择统计判别函数为

$$d_i(x) = p(x|\omega_i)P(\omega_i), \quad i = 1, 2, \dots, m$$

式中 $P(\omega_i)$ 为类 ω_i 的先验概率, $P(x|\omega_i)$ 为 x 属于类 ω_i 的多变量条件概率密度, 利用 Bayes 公式有

$$p(\omega_i|x) = \frac{p(x|\omega_i)P(\omega_i)}{\sum_{i=1}^m p(x|\omega_i)P(\omega_i)}$$

式中 $p(\omega_i|x)$ 称为后验条件概率。

可以证明, 把上述 $d_i(x)$ 作为划分模式类别的判别函数即为统计决策理论中具有 0-1 损失函数的 Bayes 决策规则, 即具有最小错误率或最小风险的 Bayes 决策规则。根据这种判别函数构成的 Bayes 分类器如图 5.5 所示。

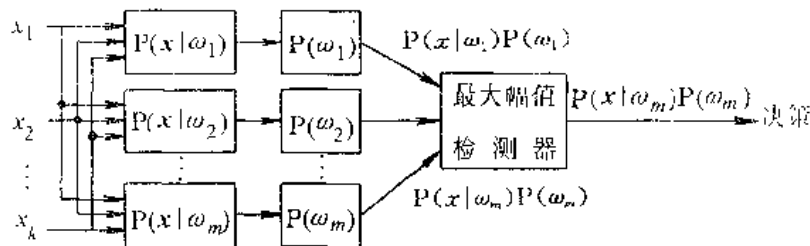


图 5.5 Bayes 分类器

上述统计判别函数也称为联合密度,记为 $p(x, \omega_i) = p(x|\omega_i)p(\omega_i)$, 相应的决策面可以表示为

$$p(x, \omega_i) = \max_{1 \leq j \leq m} p(x, \omega_j)$$

上式称为极大似然条件。对于多个特征的情况,联合密度域为一个多维参数空间,相应的极大似然决策边界出现在联合密度的最大交叉处。图 5.6 表示了二维特征空间的极大似然条件,决策面为一曲面。高维情况的决策面则为超曲面。

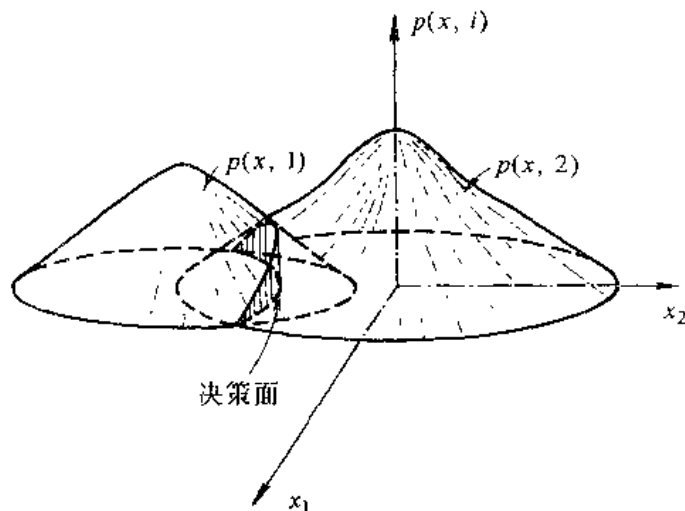


图 5.6 二维极大似然条件

5.2.3 可训练控制器

图 5.3 所示的 2-类线性分类器可作为一种可训练控制器来使用,并实现一种时间最优学习控制系统,如图 5.7 所示。

在此,特征空间 Ω_x 的划分等价于状态空间的划分,状态空间中的开关面相应于特征空间中的决策面,而状态空间或特征空间中的类区则相应于不同的控制局势或模式类。对于时间最优控制系统,2-类线性分类器的输出 $u = +1$ 或 -1 。这种输出结果既表示分类的控制局势,又表示控制作用的切换。开关面的实现可通过一种训练来完成。

时间最优控制的开关面一般是非线性的,分类器可运用分段线性的方法加以近似实

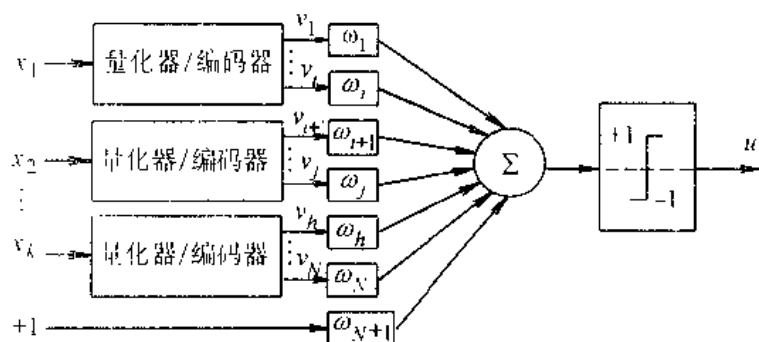


图 5.7 可训练控制器

现。如图 5.7 所示,状态空间(状态向量为 $\mathbf{x} = (x_1, x_2, \dots, x_k)^T$)首先被量化,形成一个个超立方体的基本单元(基本控制局势),单元内对应的控制作用为常量。每一个超立方体单元又用线性无关的码进行编码,构成一个二值的模式(特征)向量。这里所谓线性无关编码是指所有的模式向量都是线性无关的,否则可通过每个模式向量加入一个 +1 来实现。可以证明,经过这样编码之后,图 5.7 所示的控制器将以任意精度(通过增加量化水平)逼近开关面

$$f(x_1, x_2, \dots, x_k) = 0$$

其中 f 不含任何交叉乘积项。

上述可训练控制器的学习能力是通过调整阈值权向量 $\mathbf{w} = (w_1 \ w_2 \ \dots \ w_N \ w_{N+1})^T$ 实现的。若令 $N+1$ 维向量 $\mathbf{v} = (v_1 \ v_2 \ \dots \ v_{N+1})^T$, 则输出为

$$u = \begin{cases} +1, & \text{若 } f(\mathbf{v}) > 0 \\ -1, & \text{若 } f(\mathbf{v}) < 0 \end{cases}$$

式中

$$f(\mathbf{v}) = \mathbf{V}^T \mathbf{w}$$

一般说来,开关面并非先验已知的,但它可由训练样本集隐含确定。这里的训练样本集是由状态空间中有限个点(控制局势)组成的,而状态空间的最优控制 u^* 是已知的。特别是,状态空间中的这些点正好位于最优轨迹 $\mathbf{x}^*(t)$ 。当把这些点从空间 Ω_x 转换到 Ω_v 中时,就确定了一个训练集 $T = \{\mathbf{v}(j), u^*(j)\}, j=1, 2, \dots, L$ 。如果把 T 分解为两个子集 T_1 和 T_2 , 其中 $u^* = +1$ 的 $\mathbf{v}(j)$ 的所有元素都属于 T_1 , 而 $u^* = -1$ 的 $\mathbf{v}(j)$ 的所有元素都属于 T_2 , 那么就有

$$\begin{cases} \mathbf{v}^T \mathbf{w} > 0, & \forall \mathbf{v} \in T_1 \\ \mathbf{v}^T \mathbf{w} < 0, & \forall \mathbf{v} \in T_2 \end{cases}$$

这样,训练集 T 作为实际遇到的控制局势的典型样本,就可用来确定阈值权向量 \mathbf{w} , 而 \mathbf{w} 又可用于分类其他的控制局势。

在实际应用时,训练集 T 是从大量具有代表性的 $\{\mathbf{x}^*, u^*\}$ 中获取的,其中最优控制 u^* 可通过建立高阶模型并且模拟各种环境干扰由计算机仿真获得,它实际上相当于一个“教师”的作用。

此外,在实际训练过程中,阈值权向量 w 是根据训练集中每一新出现的模式向量以及相应的期望输出改变的。这时,训练模式向量被序贯地提供给控制器,一直到所有的模式向量(表示控制局势)被正确地分类,或者一直到分类错误数目趋近于某一稳态值。在每次不正确分类后,权的变化均为 αv 。围绕系数 α 的选择问题,可采用“最小均方误差训练法”或“错误校正训练法”等算法。

5.2.4 线性再励学习控制

心理学家认为,一个系统的具有某种特定目标的性能的任何有规律的变化都是“学习”。一般可用互斥而又完备的响应类 $\omega_1, \omega_2, \dots, \omega_m$ 来描述系统性能的变化。令 P_i 为第 i 类响应 ω_i 发生的概率,系统性能的变化可表示为响应概率集 $\{P_i\}$ 的再励,这种再励的数学表示为

$$P_i(n+1) = \alpha P_i(n) + (1-\alpha)\lambda_i(n), \quad n = 0, 1, 2, \dots; \quad i = 1, 2, \dots, m$$

其中 $P_i(n)$ 表示在观察到输入 x 的时刻 n 出现 ω_i 的概率, $0 < \alpha < 1, 0 \leq \lambda_i(n) \leq 1$, 而且

$$\sum_{i=1}^m \lambda_i(n) = 1$$

由于 $P_i(n+1)$ 与 $P_i(n)$ 之间为线性关系,以上两式常称为线性再励学习算法。

容易证明,若 $\lambda_i(n) = \lambda_i$, 则有

$$P_i(n) = \alpha^n P_i(0) + (1 - \alpha^n) \lambda_i,$$

$$\lim_{n \rightarrow \infty} P_i(n) = \lambda_i$$

由上式, λ_i 即为 $P_i(n)$ 的极限概率。从而 $\lambda_i(n)$ 一般应该与根据时刻 n 的输入 x 估计到的性能信息有关。在学习控制系统中,学习控制器的输入 x 通常是受控对象的输出,而 ω_i 则直接表示第 i 个控制作用。这样 $\lambda_i(n)$ 可看作是第 i 类响应(控制作用)相联系的归一化性能指标。在某些简单的情况下, $\lambda_i(n)$ 可为 0 或 1, 表示由于第 i 个控制作用而导致的系统性能是满意的或不满意的;或者可表示控制器在时刻 n 对输入 x 作出的决策(即分类)正确或不正确。一般说来,如果第 i 个控制作用是期望的控制,则可证明 $P_i(n)$ 将收敛于它的最大值($n \rightarrow \infty$)。

线性再励学习算法已应用于控制系统的设计。在线性再励控制器的设计中,控制器的响应类 $\omega_i (i = 1, 2, \dots, m)$, 即为相应的允许控制作用,而控制器的性能,即对不同控制局势的控制作用的品质,则可根据对象的输入输出进行评估。在对象和环境干扰的先验信息不完全的情况下,所设计的控制器将在每一时刻学习最优控制作用,学习过程可由当时估计的系统性能来引导,因而控制器就能进行“在线”的学习。线性再励学习控制系统的原理框图如图 5.8 所示。

5.2.5 Bayes 学习控制

在利用动态规划或统计决策理论设计随机最优控制器时,通常需要知道系统环境参数或对象输出的概率分布。考虑如下状态方程表示的离散随机系统

$$x(n+1) = g(x(n), u(n))$$

式中 $x(n)$ 为时刻 n 的状态向量(随机向量), $u(n)$ 为时刻 n 的控制作用。问题的提法是:寻

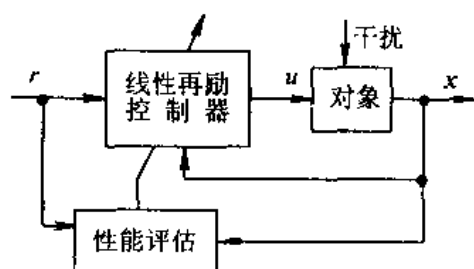


图 3.8 线性再学习控制系统

找最优控制 $u=u^*$, 使如下性能指标

$$I_n(u) = E \left\{ \sum_{n=1}^N F[x(n), u(n-1)] \right\}$$

极小。为此可利用具有已知概率密度 $p(x)$ 的动态规划方法。类似于统计模式识别中的情况, 如果概率分布或密度函数未知或不全已知, 则控制器的设计可以首先估计(学习)未知密度函数, 然后根据估计信息实现控制律。如果这种估计逼近真实函数, 则控制律也逼近最优控制律。所谓 Bayes 学习控制, 就是利用一种基于 Bayes 定理的迭代方法来估计(学习)未知的密度函数信息。

1. 具有监督的 Bayes 学习控制

设要学习的概率密度函数为 $p(x|\omega_i)$, 其中 ω_i 表示第 i 类控制局势。令 $x(1), x(2), \dots, x(n)$ 为已知控制局势类属(设属于 ω_i)的 n 个特征(学习样本)。这是一种具有外部监督(离线训练)的学习。

如果 $p(x|\omega_i)$ 已知, 但某些参数 θ 未知, 那问题就变为对给定特征 $x(1), x(2), \dots, x(n)$ 估计参数 θ 。由于 θ 未知, 因此可假定 θ 为具有某个先验分布的随机变量。

利用 Bayes 定理, 参数 θ 的后验密度可根据其先验密度函数和由样本集提供的信息来计算, 即

$$\begin{aligned} p[\theta|\omega_i, x(1), \dots, x(n)] \\ = \frac{p[x(n)|\omega_i, \theta, x(1), \dots, x(n-1)]p[\theta|\omega_i, x(1), \dots, x(n-1)]}{p[x(n)|\omega_i, x(1), \dots, x(n-1)]} \end{aligned}$$

例如, 若 $p(x|\omega_i)$ 是均值向量为 m 、协方差矩阵为 K 的高斯分布, 而未知参数 θ 是均值向量 m , 设 θ 的先验分布 $p_0(\theta|\omega_i)$ 也是高斯分布, 其初始均值向量为 $m(0)$, 初始协方差矩阵为 $\Phi(0)$ 。那么在取得了第一次样本测量后, 就有

$$p[\theta|\omega_i, x(1)] = \frac{p[x(1)|\omega_i, \theta]p_0(\theta|\omega_i)}{p[x(1)|\omega_i]}$$

由于假设 $p_0(\theta|\omega_i)$ 为高斯分布, 则乘积 $p[x(1)|\omega_i, \theta]p_0(\theta|\omega_i)$ 亦为高斯分布, 这样上式的计算将得到简化。利用高斯分布的上述性质, 重复运用 Bayes 定理, 在几次学习样本之后, 可得到估计 $\theta=m$ 的递推公式:

$$\begin{aligned} m(n) &= K[\Phi(n-1) + K]^{-1}m(n-1) + \Phi(n-1)[\Phi(n-1) + K]^{-1}x(n), \\ \Phi(n) &= K[\Phi(n-1) + K]^{-1}\Phi(n-1) \end{aligned}$$

利用 $p_0(\theta|\omega_i)$ 的先验初始估计 $m(0)$ 和 $\Phi(0)$, 以上两式就变为

$$m(n) = n^{-1}K[\Phi(0) + n^{-1}K]^{-1}m(0) + \Phi(0)[\Phi(0) + n^{-1}K]^{-1}\langle x \rangle,$$

$$\Phi(n) = n^{-1}K[\Phi(0) + n^{-1}K]^{-1}\Phi(0)$$

其中, $\langle x \rangle = \frac{1}{n} \sum_{i=1}^n x(i)$ 为样本均值。

$m(n)$ 的表达式表明, 均值向量 $m(n)$ 的第 n 次估计可解释为先验均值向量 $m(0)$ 和样本信息 $\langle x \rangle$ 的加权平均。当 $n \rightarrow \infty$ 时, $m(n) \rightarrow \langle x \rangle$ 而且 $\Phi(n) \rightarrow 0$, 这就意味着估计 $m(n)$ 将逼近真实的均值向量 m 。类似地, 如果协方差矩阵 K 未知, 或者 m 和 K 都未知, 这时也可应用 Bayes 学习方法。

2. 无监督的 Bayes 学习控制

如果不能得到学习样本 $x(1), \dots, x(n)$ 的正确分类, 那么必须运用一种无监督(在线训练)的学习。此时每一个特征 $x(i)$ 可能属于 m 类控制局势中的任何一类。比较一般的方法是根据各种可能分类的概率密度函数构造一个混合密度(或分布), 即:

$$p(x|\theta, P) = \sum_{i=1}^m P_i p(x|\omega_i, \theta_i)$$

式中 θ_i 为与 $p(x|\omega_i)$ 相关联的未知参数, P_i 为类 ω_i 的先验概率, 而

$$\theta = \{\theta_i; i = 1, \dots, m\}, \quad P = \{P_i; i = 1, \dots, m\}$$

令 $B = (\theta, P)$, 并认为未标明类别的样本序列 $x(1), \dots, x(n)$ 是从混合密度为 $p(x|\theta, P)$ 的混合体中独立抽样得到的, 经过逐次应用 Bayes 定理后就有

$$p[B|x(1), \dots, x(n)]$$

$$= \frac{p[x(n)|x(1), \dots, x(n-1), B]p[B|x(1), \dots, x(n-1)]}{p[x(n)|x(1), \dots, x(n-1)]}$$

这里需要选择一个先验概率 $p_0(B)$, 使其在 B 的真值处不等于 0。

另外, 必须强调对于给定混合密度类型的可识别性, 以便保证能唯一地学习未知参数。混合密度 $p(x|\theta, P)$ 是否能识别实际上是一个唯一性问题, 即对于第 i 个参数条件是密度函数 $\{p(x|\omega_i, \theta_i)\}$ 以及参数 θ 和 P 的集合, 混合密度 $p(x|\theta, P)$ 必须能唯一地确定参数集 $\{\theta_i\}$ 和 $\{P_i\}$ 。显然, 在这种无监督的学习中如果混合密度不能唯一地由 $\{\theta_i\}$ 和 $\{P_i\}$ 来刻画(即不可识别), 那么基于混合密度的估计问题就无解。

5.2.6 基于模式识别的其他学习控制方法

1. 随机逼近法学习控制

这是一种更一般的采用性能反馈的学习方法, 是随机逼近过程在控制器设计中的应用。这种方法的基本思想是控制器利用随机逼近过程, 针对每一类控制局势来学习最优的控制作用。其中首先要对系统性能进行合适的估计, 使这种估计能引导学习过程。由于对象和环境的特征一般是未知的或不全已知的, 当然实际上不可能得到精确的性能指标。为此, 必须适当地选择一个瞬时性能估计, 使得系统的学习过程在这一估计的引导下, 最终能保证相对于总体性能指标的最优性。一些随机逼近算法就用来首先估计这种瞬时性能指标, 然后再度利用随机逼近法, 根据这一指标, 学习相应的最优控制律。

随机逼近法在线学习控制的主要结果最早由 Z. J. Nikolic 和 K. S. Fu 于 1968 年给出,随后由 J. S. Riordon 提出的自适应自动机控制器可认为是这种方法的推广,更一般的和更完全的方法可见于 G. N. Saridis 对于轨道卫星姿态控制问题的研究,所得到的基于随机搜索算法的扩展子空间在线控制方法,对于工作在随机环境而且对象动态特性未知的控制,可给出一个满意的全局渐近最优解。

2. 随机自动机模型学习控制

线性再励学习控制方法实质上是描述了一类在随机环境中具有未知动态特性系统的学习问题。实际上,随机自动机的功能可用再励算法进行描述,而再励学习的模型则可由随机自动机提供。

在控制论中自动机是指在离散时间内,对离散数据符号进行运算的一类抽象系统。而随机自动机即指自动机未来的状态并不由初态和输入信号唯一确定,系统可以从同一状态和输入出发,按不同的概率,转移到不同的状态,得到不同的输出值。

随机自动机可以表示为五元组 (Y, Q, U, F, G) , 其中 $Y = \{y^1, y^2, \dots, y^r\}$ 为输入有限集, $Q = \{q^1, q^2, \dots, q^r\}$ 为状态有限集, $U = \{u^1, u^2, \dots, u^m\}$ 为输出有限集, 而 F 为转移状态的随机函数, G 为输出函数(确定或随机), 即

$$\begin{cases} q(n+1) = F[y(n), q(n)] \\ u(n) = G[q(n)] \end{cases}$$

对于任一输入 $y^k(n)$, F 可表示为一个状态转移概率矩阵 $M^k(n)$, 它的元素定义为随机状态的转移概率

$$\begin{cases} p_{ij}^k(n) = P\{q(n+1) = q^j | q(n) = q^i, y(n) = y^k\} \\ \sum_{j=1}^r p_{ij}^k(n) = 1, \quad i, j = 1, 2, \dots, r \end{cases}$$

上式可理解为试图达到某一预定目标的非线性再励算法, 因而随机自动机可以作为再励学习系统的模型, 这一模型指出了为改进系统的在线性能, 转移概率所应作的修改。而这些概率的修改反过来又是通过再励算法进行的, 整个再励学习正是自动机的功能。即如果外界(“教师”)对自动机的某次状态转移施行奖励(惩罚), 那么相应的状态转移概率就会上升(下降), 而其他的状态转移概率就会下降(上升)。

3. 模糊自动机和模糊学习控制

与形式语言可推广到模糊语言一样, 有限自动机的概念也可推广到模糊自动机。一个模糊自动机可表示为五元组 (I, V, Q, f, g) , 其中 I 为输入有限集 $\{i_1, i_2, \dots, i_p\}$, V 为输出有限集 $\{v_1, v_2, \dots, v_r\}$, Q 为状态有限集 $\{q_1, q_2, \dots, q_n\}$, f 为 $Q \times I \times Q$ 空间中模糊集的隶属函数, 即 $f: Q \times I \times Q \rightarrow [0, 1]$, g 为 $V \times I \times Q$ 空间模糊集的隶属函数, 即 $g: V \times I \times Q \rightarrow [0, 1]$ 。

通常, f 称为直接模糊转移函数, $f_A(q_i, i_j, q_m)$, 表示当输入为 i_j 时, 从状态 q_i 转移到 q_m (模式 A) 的隶属度。若记 $q(k) = q_i, q(k+1) = q_m, i(k) = i_j$, 则有

$$f_A(q_i, i_j, q_m) = f\{q(k) = q_i, i(k) = i_j, q(k+1) = q_m\}$$

显然, 当隶属度为 1 时, 表明这种转移存在; 当隶属度为 0 时, 则表示这种转移不存在。或者, 设 A 定义为给定输入时状态转移过程的模糊集, 且 X 表示三元组 (q_i, i_k, q_m) , 就有: 如

果 $f_A(X) \geq \alpha$, 则 X 属于 A (真); 如果 $f_A(X) \leq \beta$, 则 X 不属于 A (假); 如果 $\beta < f_A(X) < \alpha$, 则 X 相对于 A 不确定, 这里 $0 < \beta < \alpha < 1$ 。

一般说来隶属函数 f 可依赖于 k , 也可与 k 无关。前者称为平稳模糊转移函数, 后者称为非平稳模糊转移函数。通常模糊自动机的学习行为是由非平稳模糊转移函数描述的。

类似于随机自动机模型的学习控制, 也可基于模糊自动机构成一种无监督的 (在线) 学习控制方法。考虑离散时间受控对象

$$x(k+1) = \Phi_{k+1}[x(k), u(k+1)]$$

其中 $x(k), x(k+1) \in \Omega_x = \{x_i | i=1, 2, \dots, p\}$, $u(k) \in \Omega_u = \{u_i | i=1, 2, \dots, p\}$, $x(k+1)$ 为施加控制作用 $u(k+1)$ 时对象的可测响应。假设 Φ_{k+1} 未知, 则控制作用 $u(k+1)$ 的瞬时性能评价函数为

$$Z(k+1) = g[x(k), u(k+1), x(k+1)]$$

这里 $0 < Z(k) < T, k=1, 2, \dots$ 。

控制的目的是要使 Z 的样本平均 $M_{k+1}[Z|u(k), x(k), u(k+1)]$ 极小。为此需要对每一个控制策略的样本平均进行估计。设在观察 $u(k)=u_j$ 和 $x(k)=x_i$ 后, 施加控制作用 $u(k+1)=u_l$, 则有样本平均的估计

$$\hat{M}_{k+1}(Z|u_j, x_i, u_l) = \frac{N}{N+1} \hat{M}_k(Z|u_j, x_i, u_l) + \frac{1}{N+1} Z(k+1),$$

$$\hat{M}_{k+1}(Z|u_j, x_i, u_h) = \hat{M}_k(Z|u_j, x_i, u_h)$$

其中 $h=1, 2, \dots, p, h \neq l, N=N(j, i, l)$ 表示 $u(k)=u_j, x(k)=x_i$ 和 $u(k+1)=u_l$ 出现的次数。这时

$$\hat{f}_{k+1}(u_l|u_j, x_i) = 1 - \frac{\hat{M}_{k+1}(Z|u_j, x_i, u_l)}{T}$$

上式把具有极大隶属度的控制作用与使 Z 的样本均值极小化过程联系起来了。

4. 递阶语义学习控制系统

将句法模式识别中的语义学方法应用于递阶系统的学习控制系统, 就可构成一种递阶语义学习控制系统。

一般说来, 采用形式文法描述的句法模式识别不但兼具统计模式识别处理随机环境的能力, 而且还能给出模式的数学描述。这里的形式文法按照 Chomsky 的分类, 通常包括 0 型、1 型、2 型和 3 型文法, 它们分别与图灵机、线性有界自动机、非确定下推机和有限自动机等价。因此在这种意义上前述基于随机自动机和模糊自动机的学习控制系统, 都可以用形式语言或语义学的方法进行描述分析, 即可发展为相应的语义学习控制系统。

J. H. Graham 和 G. N. Saridis 于 1982 年针对机器人的递阶控制提出了一种统一的递阶语义学习控制方法。整个递阶结构的每一级都以形式文法予以表达, 各级之间控制指令的匹配则利用语义决策图来完成, 其中每个语义决策图本身都包括了某种学习算法 (如线性再励、随机逼近、Bayes 学习等)。这就使得整个系统不但可以在递阶结构的最高层把人的定性指令 (高级语言) 翻译 (细化) 成一系列对象级指令, 而且学习算法还可使系统适应于环境与对象动态特性的随机变化。

5.2.7 研究课题

本节介绍的基于模式识别的各种学习控制算法只是一些基本原理要点,深入的研究一直在进行着。可以看出,这些学习算法的主要区别在于它们需要的先验信息不同,所牵涉到的计算技术不同。所谓有监督的学习方法与无监督的学习方法在实际运用中往往是结合的;有监督的学习首先通过离线的训练(计算机仿真或人-机交互)获取尽可能多的先验信息,然后由无监督的学习负责随机环境下的在线学习。

有些一直在研究中的课题可以列举如下。

(1) 非稳态环境中的学习

大多数学习方法只是适用于稳态环境(估计稳态参数)。由于对象动态特性可能有不稳定的(未知的)环境干扰,因而需要研究这种非稳态环境中的学习问题。例如,如果非稳态环境可以用有限个不同的稳态环境来近似(形成“开关环境”),那么就能通过模式识别和混合分解的技术对这些稳态环境加以辨识,然后再施用相应学习算法。另外,非线性再励算法也可采用,只是要克服在分析方面的数学困难。

(2) 学习速率的改进

基于模式识别的学习算法一般是相当慢的,对于快速反应系统,可以适当利用一些先验知识,例如对象的动力学方程形式、参数变化的范围、环境干扰的类型等,而开发新的快速算法更为必要。

(3) 停止规则

学习控制算法一般都需要证明其渐近收敛性,即当学习样本数趋近于无穷时,应获得真实的参数。实际中,有限时间的运行只能得到很小的样本数。这样学得的信息就显得很重要。因此除了收敛性以外,还必须研究有限次重复学习算法的性质。另一方面,如果能预先说明系统的满意的容许性能(通常不是最优的),那么需要某种停止规则,使得学习过程停止在必要的时刻。

(4) 学习的层次结构

在复杂的学习过程中,可以用几种学习算法构成相关而又不同的信息获取层次。低层学习的性能依赖于高层学习得到的信息。如果高层学习总是产生正确的信息,那么低层学习就仅取决于所采用的算法,而高层学习如果采用与低层学习不同的另一种算法,它一般产生的正确信息只能渐近地趋于完备。在这种情况下,即使每一层的学习算法都是收敛的,也必须特别注意学习系统的总的收敛性问题。利用多种学习算法的系统的性能是深入研究的课题。

5.3 基于迭代和重复的学习控制

基于迭代和重复的学习控制,针对一类特定的系统但又不依赖系统的精确数学模型,它通过反复训练的方式进行自学习,使系统逐步逼近期望的输出。这类方法可导致结构简单的学习控制器。它在时域中的发展即为“迭代自学习控制”,在频域中的发展即为“重复自学习控制”。“异步自学习控制”方法将两者有机地统一起来,提出了一些新的理论观点。

5.3.1 迭代和重复自学习控制的基本原理

以下介绍由日本有本和井上等分别研究的迭代和重复自学习控制的基本原理要点。

1. 迭代自学习控制

考虑如下的线性定常系统

$$\begin{cases} R\dot{x}(t) + Q\dot{x}(t) + Px(t) = u(t) \\ y(t) = \dot{x}(t) \end{cases}$$

其中 $x(t)$ 、 $u(t)$ 和 $y(t)$ 分别为 n 维状态变量、控制变量和输出变量,且均为实变量;而 R 、 Q 和 P 分别为 $n \times n$ 的对称正定实矩阵,它们均为未知的系统矩阵。已知系统的初始条件为

$$x(0) = x_0, \quad \dot{x}(0) = \dot{x}_0 = y_d(0)$$

这里 $y_d(0)$ 是定义在有限区间 $[0, T]$ 上的期望轨迹输出。可以看出,所讨论的系统是一种速度跟踪伺服系统。

迭代自学习控制的基本思想是,基于多次重复训练(运行),只要能保证训练过程的系统不变性,控制作用的确定可在模型不确定的情况下获得有规律的原则,使系统的实际输出逼近期望输出。图 5.9 描述了这种方法的迭代运行结构和过程。

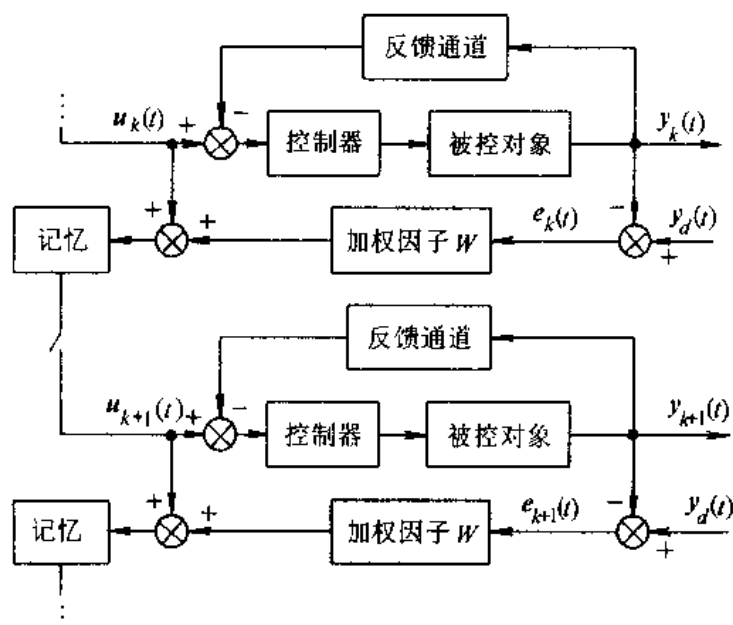


图 5.9 迭代自学习控制的运行

如图 5.9 所示,若第 k 次训练时期望输出与实际输出的误差为

$$e_k(t) = y_d(t) - y_k(t), \quad t \in [0, T]$$

第 $k+1$ 次训练的输入控制 $u_{k+1}(t)$ 则为第 k 次训练的输入控制 $u_k(t)$ 与输出误差 $e_k(t)$ 的加权和:

$$u_{k+1}(t) = u_k(t) + W e_k(t)$$

迭代自学习控制方法已经证明,设每次重复训练时都满足初始条件 $e_k(0) = 0$, 当 $k \rightarrow$

∞ , 即重复训练次数足够多时, 可有 $e_k(t) \rightarrow 0$, 即实际输出能逼近期望输出:

$$y_k(t) \rightarrow y_d(t)$$

在迭代自学习控制系统中, 控制作用的学习是通过对以往控制经验(控制作用与误差的加权和)的记忆实现的。算法的收敛性依赖于加权因子 W 的确定。这种学习系统的核心是系统不变性的假设以及基于记忆单元的间断的重复训练过程, 它的学习控制律极为简单, 可实现训练间隙的离线计算, 因而不但有较好的实时性, 而且对干扰和系统模型的变化具有一定的鲁棒性。

2. 重复自学习控制

考察图 5.10 所示的周期信号产生器, 其中 $e_1(t)$ 为定义在 $[-T, 0]$ 上的初始函数。显然, 任何周期为 T 的周期信号 $e_2(t)$ 都可由这样的纯时延环节 e^{-Ts} 产生。这种周期信号产生器的闭环传递函数为

$$F(s) = \frac{e^{-Ts}}{1 - e^{-Ts}}$$

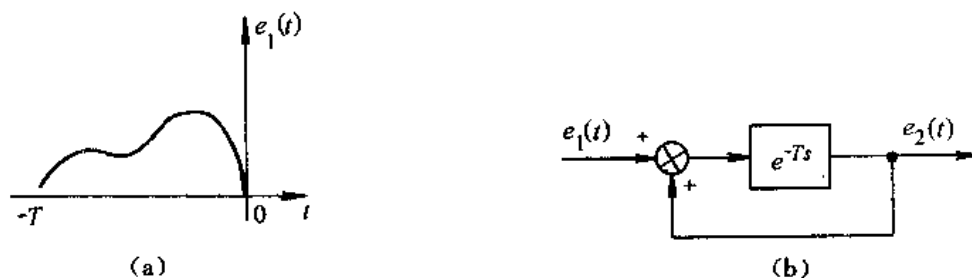


图 5.10 周期信号产生器

由内模原理可知(S. Hara 和 Y. Yamamoto, 1985), 如果将 $F(s)$ 包括在一个闭环系统内, 则可实现对外部周期信号的渐近跟踪性能。这种具有 $F(s)$ 的控制器称为重复控制器, 而具有这种重复控制器的系统称为重复自学习控制系统(图 5.11)。

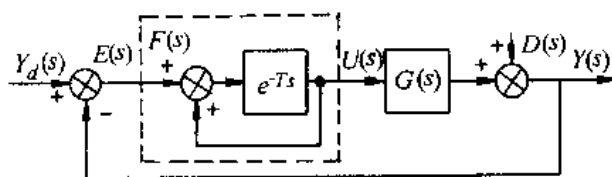


图 5.11 重复自学习控制系统

图 5.11 中, $Y_d(s)$, $Y(s)$ 分别为期望输出和实际输出, $D(s)$ 为未知的有界连续外部干扰, 它与 $Y_d(s)$, $Y(s)$ 同为周期是 T 的周期信号。 $U(s)$ 为控制作用, $G(s)$ 为前馈或反馈补偿后的被控系统的传递函数。

由图 5.11 可得:

$$\begin{aligned} Y(s) &= G(s)U(s) + D(s), \\ U(s) &= e^{-Ts}[U(s) + E(s)], \end{aligned}$$

$$E(s) = Y_d(s) - Y(s)$$

再由以上三式可导出

$$E(s) = e^{-Ts}[1 - G(s)]E(s) + Y_{d0}(s)$$

其中 $Y_{d0}(s)$ 称为等价期望输出,

$$Y_{d0}(s) = (1 - e^{-Ts})[Y_d(s) - D(s)]$$

上式可表示为图 5.12 所示的系统,它与图 5.11 的系统等价。

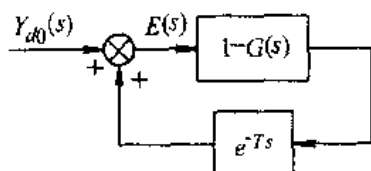


图 5.12 重复自学习控制系统的等价系统

通过讨论图 5.12 的等价系统的有界输入有界输出稳定性,可以说明重复自学习控制系统的误差收敛条件或稳定条件。

对等价系统的表达式两边求 Laplace 逆变换,可得

$$y_{d0}(t) = [y_d(t) - y_d(t - T)] - [d(t) - d(t - T)]$$

由于期望输出 $y_d(t)$ 和干扰输入 $d(t)$ 为具有相同周期 T 的有界连续周期函数 ($t \geq 0$),则由上式可知

$$y_{d0}(t) = \begin{cases} y_d(t) - d(t), & 0 \leq t \leq T \\ 0, & t \geq T \end{cases}$$

这说明等价期望输出 $y_{d0}(t)$ 为 L_2 函数。据此,如果用 $\|G(s)\|_\infty = \sup_\omega |G(j\omega)|$ 表示渐近稳定的传递函数 $G(s)$ 的范数,则由小增益定理可知,若 $G(s) \in R_+$, 且 $\|1 - G(j\omega)\|_\infty < 1$,再考虑到 $|e^{-j\omega T}| = 1$,则必得偏差 $e(t)$ 有界的结论,即

$$e(t) \in L_2$$

进一步,在对纯时延环节 e^{-Ts} 加入低通滤波器和顺馈环节后,就可证明

$$\lim_{t \rightarrow \infty} e(t) = 0$$

即说明误差 $e(t)$ 是一致收敛的。

总之,上述重复自学习控制的基本思想是,基于重复控制器的作用,经过多个周期的重复训练(运行),只要能保证系统的周期不变性,控制作用的确定可在干扰不确定的情况下获得有规律的原则,使系统的实际输出逼近期望输出。具体地,设第 k 个周期训练时期望输出与实际输出的误差为

$$e_k(t) = y_d(t) - y_k(t), \quad t \in [(k-1)T, kT]$$

则第 $k+1$ 个周期训练的输入控制 $u_{k+1}(t)$ 为第 k 个周期训练的输入控制 $u_k(t)$ 与输出误差 $e_k(t)$ 的加权和

$$u_{k+1}(t) = u_k(t) + W e_k(t)$$

根据前面讨论的误差收敛性,当 $k \rightarrow \infty$ 时,系统的实际输出可逼近期望输出。

类似于迭代自学习控制系统,重复自学习控制系统中控制作用的学习也是通过对以往控制经验的记忆实现的。这种方法与迭代自学习控制的区别在于,它仅对周期性期望输出成立;各次训练的系统不变性退化为周期的不变性;记忆功能由重复控制器体现;它对控制作用的修正是连续的,不再具有间断离线计算的特点。

5.3.2 异步自学习控制

1. 基本概念

考虑到迭代自学习控制和重复自学习控制的共同点和区别,我国的邓志东将这两种方法统一起来,提出了一种异步自学习控制的理论框架。他的基本思想是:将第 k 次重复训练的迭代自学习控制系统看成是对第 k 个重复周期的“间断”的重复自学习控制系统,且前者的训练时间等于后者的重复周期;将重复自学习控制系统的重复控制器视为一个记忆系统。这样图 5.9 所示的迭代自学习控制系统与图 5.11 所示的重复自学习控制系统完全等价,而迭代自学习控制系统的期望输出 $y_d(t)$ 可认为是以训练时间为重复周期的周期性有界连续信号。

异步自学习控制系统的基本结构如图 5.13 所示。其中 $y_d(t)$ 是周期为 T 的有界连续期望输出, $u_k(t), u_{k+1}(t)$ 分别是第 k 次、第 $k+1$ 次迭代的参考输入, $y_k(t)$ 是第 k 次迭代的闭环控制系统实际输出, $e_k(t)$ 定义为第 k 次迭代的输出偏差,即 $e_k(t) = y_d(t) - y_k(t)$ 。另外, $t \in [(k-1)T, kT], k=1, 2, \dots, T$ 为迭代学习周期。

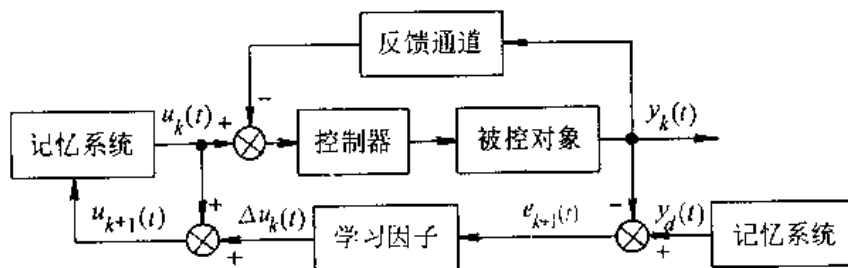


图 5.13 异步自学习控制系统

异步自学习控制基于以下一些假设:(1) 异步自学习控制闭环系统或是定常系统,或是周期时变系统。若将学习周期 T 视为“间断”学习的训练时间,这一假设对周期时变系统的限制可放宽为一般的时变系统,更具一般性;(2) 期望输出 $y_d(t)$ 是学习周期为 T 的有界连续周期信号。这一假设将保证学习控制规律的稳定性或输出偏差 $e_k(t)$ 的收敛性,可通过对学习周期初始时刻的重定位方法得以解决;(3) 学习控制满足初始条件 $e_k(0)=0, k=1, 2, \dots$ 。

可以看出,迭代自学习控制和重复自学习控制实际上是异步自学习控制的特例,即分别是具有“间断”和“连续”学习的异步自学习控制。其中,连续时间系统的异步自学习控制也可表示为二维离散形式。这时图 5.13 中的 $u_k(t), y_k(t), y_d(t)$ 和 $e_k(t)$ 分别由 $u(k, i), y(k, i), y_d(i)$ 和 $e(k, i)$ 替代, $k=1, 2, \dots, i=0, 1, 2, \dots, l-1$ 。而 $T=l\tau$,其中 T 为学习周期, τ 为采样周期。

异步自学习控制的“异步”含意是,第 $k+1$ 步的参考输入总是基于第 k 步以前的经验,即存在不同步或异步。另外在二维离散形式中一般有 $T \neq \tau$,这也含有异步之意。

2. 基本原理

(1) 问题的算子描述

设反馈闭环控制系统的输入输出算子形式为

$$y_k(t) = T_P u_k(t), \quad t \in [(k-1)T, kT], \quad k = 1, 2, \dots \quad (5.1)$$

显然,若此闭环系统稳定,则必对应 T_P 有界。

假设系统的期望输出 $y_d \in C^1[(k-1)T, kT]$, 系统的输出偏差为 $e_k(t) = y_d(t) - y_k(t)$, $t \in [(k-1)T, kT]$, $k = 1, 2, \dots$, 则由图 5.13 可知,在第 k 次学习时,参考输入 $u_k(t)$ 与修正量 $\Delta u_k(t)$ 之和将存储在记忆系统中,并作为第 $k+1$ 次学习时的给定输入,即

$$u_{k+1}(t) = u_k(t) + \Delta u_k(t) = u_k(t) + T_L e_k(t), \quad k = 1, 2, \dots \quad (5.2)$$

式中 T_L 为学习算子, $\Delta u_k(t) = T_L e_k(t)$ 实际上是第 k 次学习时的偏差加权量,也即第 k 次学习时已积累的经验。

异步自学习控制的目标是:第 $k+1$ 次学习时的输入能基于第 k 次学习时的输入和偏差加权量获得,并且随着有效经验的不断积累,最终使

$$e_k(t) \rightarrow 0,$$

$$y_k(t) \rightarrow y_d(t),$$

$$\text{或 } k \rightarrow \infty, (k-1)T \leq t \leq kT$$

即,使闭环系统的实际输出经过学习而逐渐逼近期望输出。

(2) 稳定性与一致收敛性

由式(5.1)可得

$$e_k(t) = y_d(t) - y_k(t) = y_d(t) - T_P u_k(t) \quad (5.3)$$

将式(5.2)代入上式,有

$$\begin{aligned} e_{k+1}(t) &= y_d(t) - T_P [u_k(t) + T_L e_k(t)] \\ &= [y_d(t) - T_P u_k(t)] - T_P T_L e_k(t) \\ &= e_k(t) - T_P T_L e_k(t) \\ &= (1 - T_P T_L) e_k(t) \end{aligned} \quad (5.4)$$

式中“1”为恒同算子。若令 $T_e = 1 - T_P T_L$, 称 T_e 为误差传播算子,则式(5.4)可写为

$$e_{k+1}(t) = T_e e_k(t) \quad (5.5)$$

上式表明,可以对 T_e 讨论 $e_k(t)$ 的一致收敛性。

定义 5.3.1 若学习误差一致减小,即对任何 $k=1, 2, \dots$, 都有

$$\|e_{k+1}(t)\| \leq \theta \|e_k(t)\|, \quad 0 \leq \theta \leq 1, \quad (k-1)T \leq t \leq kT$$

其中 $\|\cdot\|$ 为定义在 Banach 空间 E 上的范数,则称由式(5.1)、(5.2)表示的异步自学习控制系统 Σ_L 是稳定的。

定义 5.3.2 异步自学习控制系统 Σ_L 是 L_2 稳定的,如果 $e_k(t) \in L_2$ 或

$$\int_0^\infty e_{k+1}^T(t) e_{k+1}(t) dt < \int_0^\infty e_k^T(t) e_k(t) dt \quad (5.6)$$

显然,这里的 L_2 稳定实际就是有界输入有界输出稳定。

定义 5.3.3 异步自学习控制系统 Σ_L 是渐近稳定的, 如果对任何 $t \in [(k-1)T, kT]$, 都有

$$e_k(t) \rightarrow 0, \quad k \rightarrow \infty$$

换言之, 学习误差 (输出偏差) $e_k(t)$ 的一致收敛与 Σ_L 的渐近稳定性是完全等价的概念。此外, 若 Σ_L 是渐近稳定的, 则 Σ_L 也必是 L_2 稳定的, 但反之不一定成立。

定理 5.3.1 异步自学习控制系统 Σ_L 为 L_2 稳定的充分条件是

$$(1 - T_e^* T_e)x \geq 0, \quad x \in E \quad (5.7)$$

其中 E 为 $L_2[0, \infty]$ 空间, $T_e \in B(E)$ (有界线性算子空间), 且 T_e^* 为 T_e 的共轭算子。

证明: 假设 Σ_L 为 L_2 稳定, 则由式 (5.6)、(5.4) 可得

$$\int_0^\infty [T_e e_k(t)]^T [T_e e_k(t)] dt < \int_0^\infty e_k^T(t) e_k(t) dt$$

考虑到 $L_2[0, \infty]$ 为自共轭空间, 则有

$$\int_0^\infty e_k^T(t) T_e^* T_e e_k(t) dt < \int_0^\infty e_k^T(t) e_k(t) dt$$

即有

$$\int_0^\infty e_k^T(t) (1 - T_e^* T_e) e_k(t) dt > 0$$

而上式成立的一个充分条件为

$$(1 - T_e^* T_e)x \geq 0, \quad x \in E \quad (\text{证毕})$$

特别地, 若 $L_2[0, \infty]$ 为复赋范线性空间, 即从频域的角度上考虑上述异步自学习控制系统的稳定性问题, 则式 (5.5) 可写为

$$E_{k+1}(s) = S(s)E_k(s)$$

其中, $E_k(s)$ 为 $e_k(t)$ 的 Laplace 变换, 且

$$S(s) = I_m - H(s)L(s) \quad (5.8)$$

式中 $H(s)$ 为 $m \times m$ 阶闭环传递函数矩阵, $L(s)$ 为 $m \times m$ 阶学习矩阵, $S(s)$ 称为 $m \times m$ 阶误差传递矩阵。这样式 (5.7) 就变为

$$1 - S^H(s)S(s) \geq 0$$

可以证明, 误差传递矩阵 $S(s)$ 就是 IQ 问题中的灵敏度矩阵。

(3) 学习过程与逆系统逼近

下面将表明, 异步自学习控制系统的学习过程本质上通过重复学习而逐渐逼近其逆系统的过程。

定理 5.3.2 (川村等, 1985) 设 T_E 为定义在 Banach 空间 E 上的有界线性算子, 则当 $\|1 - T_E\| < 1$ 时, T_E 的唯一有界线性逆算子 T_E^{-1} 存在, 且可由如下 Neumann 级数表示为

$$T_E^{-1}y = \lim_{n \rightarrow \infty} [1 + (1 - T_E) + (1 - T_E)^2 + \cdots + (1 - T_E)^n]y$$

式中 $y \in E_1$ (定理证明从略)。

设 $u_1(t) = T_L y_d(t)$, 则由式 (5.3) 可知

$$e_1(t) = y_d(t) - T_p T_L y_d(t) = (1 - T_p T_L) y_d(t)$$

再由式(5.4)可得

$$e_k(t) = (1 - T_p T_L)^k y_d(t)$$

这样,第 $k+1$ 次学习时的给定输入为

$$\begin{aligned} u_{k+1}(t) &= u_{k+1}(t) + T_L e_k(t) \\ &= u_1(t) + T_L e_1(t) + T_L e_2(t) + \cdots + T_L e_k(t) \\ &= T_L [y_d(t) + e_1(t) + e_2(t) + \cdots + e_k(t)] \\ &= T_L [1 + (1 - T_p T_L) + (1 - T_p T_L)^2 + \cdots + (1 - T_p T_L)^k] y_d(t) \quad (5.9) \end{aligned}$$

进一步假定 $T_E = T_p T_L \in B(E)$ (如 T_L 为常数算子),则由定理 5.3.1 可知

$$\|1 - T_E\|_2 = \|1 - T_p T_L\|_2 = \|T_p\|_2 < 1$$

再对式(5.9)两边取极限,则由定理 5.3.2 可得

$$\begin{aligned} \lim_{k \rightarrow \infty} u_{k+1}(t) &= T_L \lim_{k \rightarrow \infty} [1 + (1 - T_E)^2 + \cdots + (1 - T_E)^k] y_d(t) \\ &= T_L T_E^{-1} y_d(t) \\ &= T_L (T_p T_L)^{-1} y_d(t) \\ &= T_p^{-1} y_d(t) \end{aligned}$$

即当 $k \rightarrow \infty$ 时,

$$u_{\infty}(t) = T_p^{-1} y_d(t)$$

上式与式(5.1)的对照比较表明,异步自学习控制系统的学习过程本质上就是逐渐构成逆系统的过程,且其学习次数相当于 Neumann 级数展开的阶次,因而随着学习次数的增加,异步自学习控制系统就逐步逼近逆系统,或者说实际输出 $y_k(t)$ 愈加逼近期望输出 $y_d(t)$ 。

5.3.3 异步自学习控制时域法

异步自学习控制时域法是指在时间域内的研究,主要以时域表达式为描述与分析的方法,它可包括表示为二维形式的连续与离散时间的两类系统。

1. 学习因子选择的一个充分条件

包括迭代和重复自学习控制在内,异步自学习控制的学习特点主要体现在“记忆系统”和“学习因子”两个环节上。记忆系统将输出误差延时一个学习周期,它对学习系统的影响是结构性的,而学习因子将直接决定系统的学习稳定性(误差的一致收敛性)和学习动态品质(学习收敛速度)。异步自学习控制系统的设计某种意义上就是学习因子的设计。

假设图 5.13 的异步自学习控制系统中的被控对象为完全能控且无零点的系统,其状态方程为

$$\dot{\mathbf{x}}^{(k)}(t) = A\mathbf{x}^{(k)}(t) + B\mathbf{u}_1^{(k)}(t) \quad (5.10)$$

式中, $(k-1)T \leq t \leq kT$, T 为学习周期, $k=1,2,\dots$; A 、 B 分别为 $n \times n$ 实矩阵和 $m \times m$ 实矩阵, (A, B) 完全能控。定义二次型性能指标

$$J = \int_0^\infty [\mathbf{x}^{(k)T}(t) Q \mathbf{x}^{(k)}(t) + \mathbf{u}_1^{(k)T}(t) R \mathbf{u}_1^{(k)}(t)] dt \quad (5.11)$$

其中, Q 、 R 分别为 $n \times n$ 、 $m \times m$ 正定实矩阵。根据现代控制理论中的 LQ 问题可知,使上

述性能指标极小的最优状态反馈为 $u_1^{(k)*}(t) = -K^* x^{(k)}(t)$, 且

$$K^* = R^{-1} B^T P$$

式中 P 为 $n \times n$ 实矩阵, 它是满足一个 Riccati 矩阵方程的唯一正定解, 方程形如

$$PA + A^T P - PBR^{-1}B^T P + Q = 0$$

因而若将 K^* 看成输出矩阵并写成传递函数的形式, 则前向通道的传递函数矩阵为

$$G(s) = K^* G_0(s) = K^* (sI - A)^{-1} B$$

这里 $G_0(s)$ 表示被控对象的 $m \times m$ 传递函数矩阵。从而

$$y^{(k)}(t) = K^* x^{(k)}(t)$$

$$u_1^{(k)}(t) = -y^{(k)}(t)$$

且

$$y_d^{(k)}(t) = K^* x_d^{(k)}(t)$$

其中 $x_d^{(k)}(t)$ 为期望状态, 相应的方框图如图 5.14(a) 所示。

显然, 若将状态 $x^{(k)}(t)$ 看成输出, 则图 5.14(a) 可等价地变换为图 5.14(b) 的形式。这时, 最优状态反馈阵 K^* 实际上是异步自学习控制的学习因子。以下将证明, 由这样得到的学习因子组成的异步自学习控制系统满足定理 5.3.1 的 L_2 稳定。

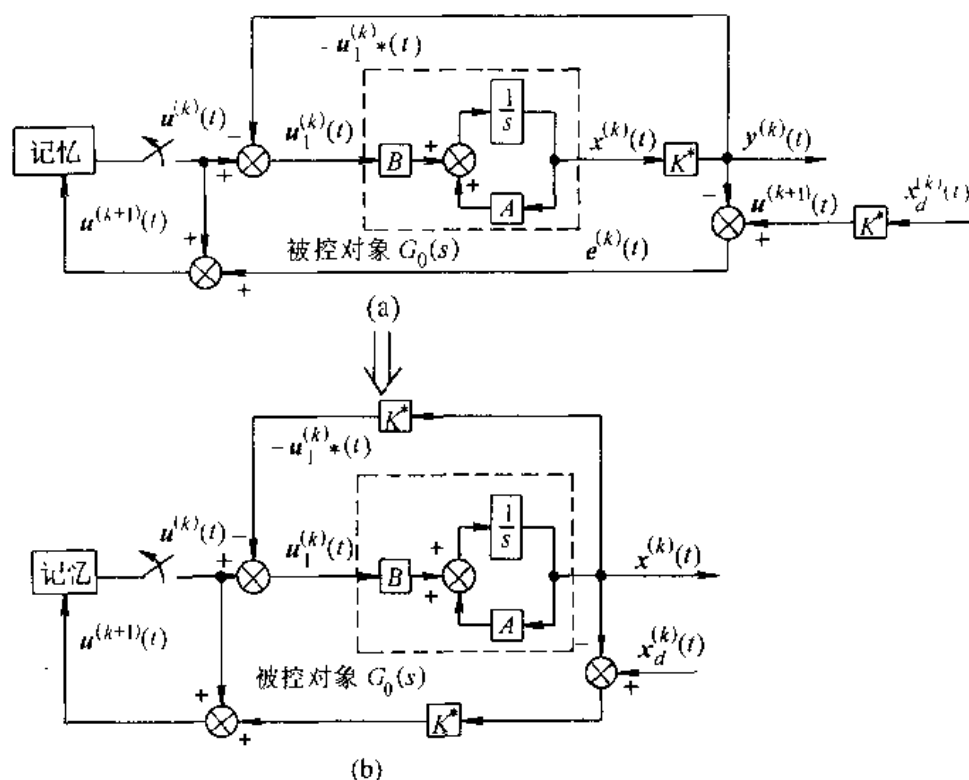


图 5.14 学习因子选择的一个充分条件

定理 5.3.3 对式(5.10)的系统, 使状态反馈

$$u_1^{(k)}(t) = -K x^{(k)}(t)$$

成为式(5.11)二次型性能指标的最优控制的充分条件为: (1) 反馈闭环系统 $\dot{x}^{(k)}(t) = (A - BK)x^{(k)}(t)$ 渐近稳定; (2) $D^H(j\omega)RD(j\omega) \geq R$, 对 $\forall \omega$ 。这里, $D(s) = I_m + G(s)$ 为回差

矩阵。

Kalman 证明了定理 5.3.3 的单输入输出情况(川村等,1985),并表明了最优调节等价于使灵敏度的绝对值减小。B. D. O. Anderson, Cruz 和 Perkins 分别证明了定理 5.3.3 的多输入多输出情况(B. D. O. Anderson 和 J. B. Moore,1971;川村等,1985)。在有关的证明中,回差矩阵的逆被用来定义为灵敏度矩阵

$$\tilde{S}(s) = D^{-1}(s)$$

这样,定理 5.3.3 中充分条件(2)可写为

$$R - \tilde{S}^H(j\omega)R\tilde{S}(j\omega) \geq 0, \quad \text{对 } \forall \omega \quad (5.12)$$

这表明,最优调节就是使灵敏度减小。反之,使灵敏度减小的渐近稳定的状态反馈就是使式(5.11)极小的最优调节。

定理 5.3.4 对式(5.10)的系统,使如图 5.14(b)所示的异步自学习控制系统具有 L_2 稳定的学习因子选择的充分条件为

$$K = K^*$$

其中 K^* 为选取加权矩阵 $Q \geq 0$ 且 $R = I_m$ 的 $m \times m$ 最优状态反馈矩阵。

证明:若选取 $Q \geq 0$ 且 $R = I_m$,则式(5.12)变为

$$1 - \tilde{S}^H(j\omega)\tilde{S}(j\omega) \geq 0, \quad \text{对 } \forall \omega$$

考虑图 5.14(a)的系统,由式(5.8)可知,其误差传递矩阵为

$$\begin{aligned} S(s) &= I_m - H(s) \\ &= I_m - [I_m + G(s)]^{-1}G(s) \\ &= [I_m + G(s)]^{-1}[I_m + G(s)] - [I_m + G(s)]^{-1}G(s) \\ &= [I_m + G(s)]^{-1} \\ &= D^{-1}(s) \\ &= \tilde{S}(s) \end{aligned}$$

即误差传递矩阵等于灵敏度矩阵。于是式(5.12)进一步可改写为

$$1 - S^H(j\omega)S(j\omega) \geq 0, \quad \text{对 } \forall \omega \quad (5.13)$$

由于式(5.13)的学习稳定条件与式(5.12)的最优调节的构成条件完全相同,因而由定理 5.3.1 可知,在式(5.11)的性能指标中选取适当的加权阵 $Q \geq 0$ 且 $R = I_m$,并求出最优状态反馈增益 K^* 后,则图 5.14(a)的最优反馈系统就成为图 5.14(b)的学习因子为 K^* ,而且具有 L_2 稳定的异步自学习控制系统,即有

$$x^{(k)}(t) \rightarrow x_d^{(k)}(t), \quad k \rightarrow \infty$$

2. PID 型异步自学习控制及其收敛性

在式(5.2)中,若将学习算子 T_L 取为 PID 控制律形式,就成为一种 PID 型异步自学习控制系统(图 5.15),其异步自学习控制律为

$$u_{k+1}(t) = u_k(t) + \left[\alpha + \beta \frac{d}{dt} + \gamma \int dt \right] e_k(t) \quad (5.14)$$

式中, α, β, γ 分别为比例、微分、积分“学习因子”, $kT \leq t \leq (k+1)T, k=0,1,\dots,T$ 为学习周期。同样, $e_k(t) = y_d(t) - y_k(t)$, 且 $e_k(0) = 0$, 初始输入 $u_0(t)$ 应使闭环系统保持稳定,期望输出 $y_d(t)$ 应满足周期为 T 的不变性。

在图 5.15 中,若有 $\beta=\gamma=0$,即采用“P”型异步自学习控制时,图 5.15 就成为图 5.14(b)的形式。因此,对于一类广泛的线性定常系统,定理 5.3.4 实际已给出了“P”型学习控制律 L_2 稳定的充分条件。下面不加证明地列出进一步研究 PID 型异步自学习控制渐近稳定性的主要结论,这些结论是针对三类特定系统展开的。

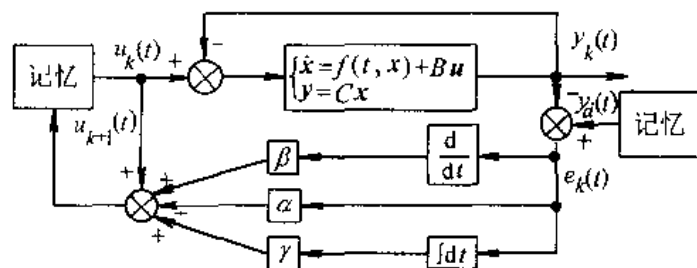


图 5.15 PID 型异步自学习控制系统

(1) 线性定常系统

考虑如下线性定常系统

$$\begin{cases} R\ddot{x}_k(t) + Q\dot{x}_k(t) + Px_k(t) = u_k(t) \\ y_k(t) = \dot{x}_k(t), \quad kT \leq t \leq (k+1)T, \quad k = 0, 1, \dots \end{cases} \quad (5.15)$$

其中 n 维实向量 u, x, y 分别为输入、状态、输出向量, R, Q, P 为 $n \times n$ 对称正定实矩阵。

定理 5.3.5 假设 (I) $x_k(0) = x_0, \dot{x}_k(0) = y_d(0) = \dot{x}_0$; (II) $u_0(t) \in C[0, T], y_d(t) \in C^1[kT, (k+1)T]$; (III) $\alpha > 0$ 且 $2Q > \alpha > 0$ (即 $2Q - \alpha$ 正定), 则如下“P”型异步自学习控制律

$$u_{k+1}(t) = u_k(t) + \alpha e_k(t)$$

关于 $y_d(t)$ 为 L_2 学习稳定, 即当 $k \rightarrow \infty$ 时, 有

$$e_k(t) \in L_2, \quad kT \leq t \leq (k+1)T, \quad k = 0, 1, \dots \quad (5.16)$$

定理 5.3.5 只能保证速度误差 $e_k(t) = y_d(t) - y_k(t) = \dot{x}_d(t) - \dot{x}_k(t)$ 的 L_2 收敛, 而一致收敛的结论可由下面的推论给出。

推论 5.3.1 在与定理 5.3.5 相同的条件下, 式(5.16)的“P”型异步自学习控制律关于 $x_d(t)$ 渐近稳定, 即

$$x_k(t) \xrightarrow{\text{一致地}} x_d(t), \quad t \in [kT, (k+1)T]$$

其中

$$x_d(t) = x_0 + \int_{kT}^t y_d(\tau) d\tau$$

(2) 线性周期性时变系统

考虑线性周期性时变系统

$$\begin{cases} R_k(t)\ddot{x}_k(t) + Q_k(t)\dot{x}_k(t) + P_k(t)x_k(t) = u_k(t) \\ y_k(t) = \dot{x}_k(t), \quad kT \leq t \leq (k+1)T, \quad k = 0, 1, \dots \end{cases}$$

式中符号同式(5.15), 另外对于一切 $kT \leq t \leq (k+1)T, k = 0, 1, \dots$, 都有 $R_k(t) = R_k^T(t) \geq$

$R_0 > 0, Q_k(t) + Q_k^T(t) \geq Q_0 > 0, P_k(t) + P_k^T(t) \geq P_0 > 0$, 且各项连续可微。

定理 5.3.6 假设 (I) $x_k(0) = x_0, \dot{x}_k(0) = y_d(0), = \dot{x}_0$; (II) $u_0(t) \in C[0, T], y_d(t) \in C^1[kT, (k+1)T]$; (III) $\alpha > 0, \gamma > 0, \|\alpha^{-1}\gamma\| = \lambda \ll 1$ 且 $P_0 - \gamma - 2\gamma^2 R_k(t) > 0$, 其中 $\|\cdot\|$ 表示矩阵的谱半径, 则如下“PI”型异步自学习控制律

$$u_{k+1}(t) = u_k(t) + (\alpha + \gamma) \int dt e_k(t) \quad (5.17)$$

关于 $y_d(t)$ 为 L_2 学习收敛。

推论 5.3.2 在与定理 5.3.6 相同的条件下, 式(5.17)的“PI”型异步自学习控制律关于 $x_d(t)$ 渐近稳定。

(3) 非线性系统

考虑一种特殊的二阶非线性微分方程(Cartwright-Littlewood 方程)。

$$\ddot{x}(t) + \mu f(x)\dot{x}(t) + g(x) = \mu u(t) \quad (5.18)$$

假定 $\mu > 0$; f 和 g 为连续函数; g 对所有 x 满足 Lipschitz 条件; 存在正数 a 和 b 使得对于 $x \geq a, g(x) \geq b > 0$, 而对于 $x \leq -a, g(x) \leq -b$ 。若令 $E(t) = \int_0^t u(\tau) d\tau, F(x) = \int_0^x f(\tau) d\tau, G(x) = \int_x^0 g(\tau) d\tau$, 则式(5.18)可表示为如下非线性状态方程

$$\begin{cases} \dot{x}_1 = x_2 - [F(x_1) - E(t)] \\ \dot{x}_2 = -g(x_1) \end{cases}$$

输出方程可为

$$y(t) = x(t)$$

定理 5.3.7 假设 (I) $x_k(0) = y_d(0) = x_0, \dot{x}_k(0) = \dot{x}_0$; (II) $u_0(t) \in C[0, T], y_d(t) \in C^1[kT, (k+1)T]$; (III) $dg(x_1)/dx_1$ 在 $(-\infty, \infty)$ 上有界; (IV) $\beta > 0, 2f(x_1) - \beta > 0$, 且 $\int_{kT}^t d_k(\tau) \int_{kT}^{\tau} \frac{dg(x_1)}{dx_1} d_k(\tau_1) d\tau_1 d\tau \geq 0$, 其中 $d_k(t) = e_k(t) - e_{k+1}(t)$, 则如下“D”型异步自学习控制律

$$u_{k+1}(t) = u_k(t) + \beta \frac{d}{dt} u_k(t)$$

关于 $y_d(t)$ 为 L_2 学习稳定。

3. 其他研究结论

由上可知, PID 型异步自学习控制不需要被控系统的精确模型, 它只需要被控系统的定性知识, 即满足稳定性的某个大致范围。另外, 当非线性对系统的影响较弱且可等价地视为一个外加干扰时, 则只需要这些干扰使实际输出 $y_k(t)$ 具有重复性, 那么就可避开困难的非线性学习控制问题, 而直接利用线性的异步自学习控制方法。

在前述研究内容的基础上, 通过引入 L_2 空间范数指标, 还可将 PID 型异步自学习控制发展为一种具有加权矩阵的最优异步自学习控制, 使控制系统更加平滑地跟踪期望输出。另外还将异步自学习控制推广到范围更大一类线性系统。

由于异步自学习控制系统实质上是在闭环回路之上加入学习回路构成的, 因而闭环系统本身的负反馈可部分抑制系统内部的不确定性, 但对量测噪声仍然无能为力。而如果

量测噪声使学习过程不再具有重复性时,就会由于噪声在重复学习过程中积累、放大而使实际输出愈加偏离期望输出。要解决这一问题可在学习修正时对输出误差或计算出的修正输入施加滤波器,以便减小量测噪声的影响,并使学习稳定条件放宽。为此,根据随机逼近理论,可通过引入一种随机泛函指标,从而使确定性的异步自学习控制推进到随机问题的形式。

5.3.4 异步自学习控制频域法

异步自学习控制频域法是指在频率域内的研究。由于只采用一般的传递函数,因而被控对象通常仅限于线性定常连续系统。

在图 5.13 中,设被控对象和控制器的传递函数矩阵分别为 $G_p(s) \in R^{m \times m}$, $G_c(s) \in R^{m \times m}$, 这里 $R^{m \times m}$ 表示 $m \times m$ 阶稳定真有理传递函数矩阵空间,为了简单起见,就采用单位反馈。如果 (I) 按照重复控制器的思想,将实现 l 步异步自学习的记忆系统考虑为一个非线性的纯时延环节 e^{-lT_s} ($l \geq 1$); (II) 设学习因子为 $L(s) \in R^{m \times m}$, 并称为学习矩阵,那么图 5.13 可等价变换为如图 5.16 的频域形式。

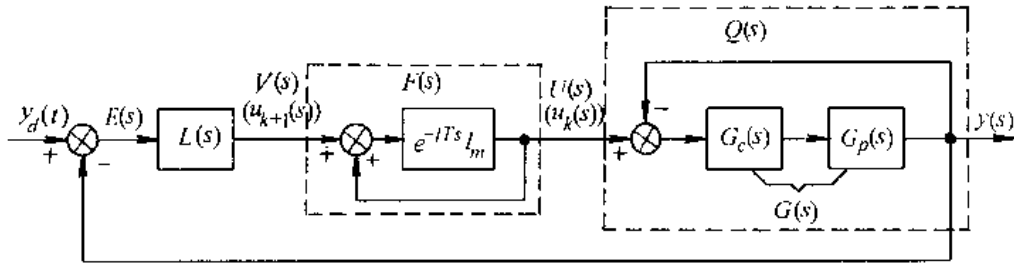


图 5.16 异步自学习控制频域法 (I)

不失一般性,若只考虑 $l=1$,即一步自学习的情况,则由图 5.16 可看出:

$$F(s) = e^{-T_s} I_m / (1 - e^{-T_s}),$$

$$G(s) = G_p(s) G_c(s)$$

且

$$Q(s) = [I_m + G(s)]^{-1} G(s)$$

这时显然有 $G(s), Q(s) \in R^{m \times m}$ 。从而式(5.2)的算子形式异步自学习控制律可写成频域形式

$$U_{k+1}(s) = U_k(s) + L(s) E_k(s) \quad (5.19)$$

其中

$$U_k(s) = e^{-T_s} U_{k+1}(s) \quad (5.20)$$

上式表明,第 $k+1$ 次迭代与 k 次迭代的给定输入之间具有明确的非线性关系,这是与时域法不同之处。由于第 $k+1$ 次的诸量可由式(5.20)折算为第 k 次进行,以下将以无下标的 $V(s), U(s)$ 记法分别代替 $U_{k+1}(s), U_k(s)$ 。应当指出,上述结论只对“连续学习”的异步自学习控制系统而言,对于“间断学习”的情况,记忆系统不可能用 e^{-T_s} 来简单表示,只

能用存储的方法实现记忆。

1. 稳定性分析

(1) 一般情形

定理 5.3.8 对于图 5.16 的系统,取 $l=1$ (下同),如果 (1) $Q(s) \in R_-$, R_- 表示稳定的真有理传递函数矩阵空间; (2) $\|I_m - Q(s)L(s)\|_\infty < 1$, $\|\cdot\|_\infty = \sup \bar{\sigma}(\cdot)$, 而 $\bar{\sigma}(\cdot)$ 表示矩阵的最大奇异值, 那么对于有界连续周期信号 $y_d(t)$ 就有

$$e(t) = L^{-1}[E(s)] \in L_2 \quad (5.21)$$

即有 L_2 学习稳定成立。

证明: 由图 5.16 可知

$$\begin{aligned} E(s) &= Y_d(s) - Y(s), \\ Y(s) &= Q(s)U(s), \\ U(s) &= e^{-Ts}[U(s) + L(s)E(s)] \end{aligned} \quad (5.22)$$

从而可得,

$$E(s) = e^{-Ts}[I_m - Q(s)L(s)]E(s) + Y_{d0}(s)$$

其中,

$$Y_{d0}(s) = (1 - e^{-Ts})Y_d(s) \quad (5.23)$$

于是可作出其等价系统如图 5.17(a)所示。

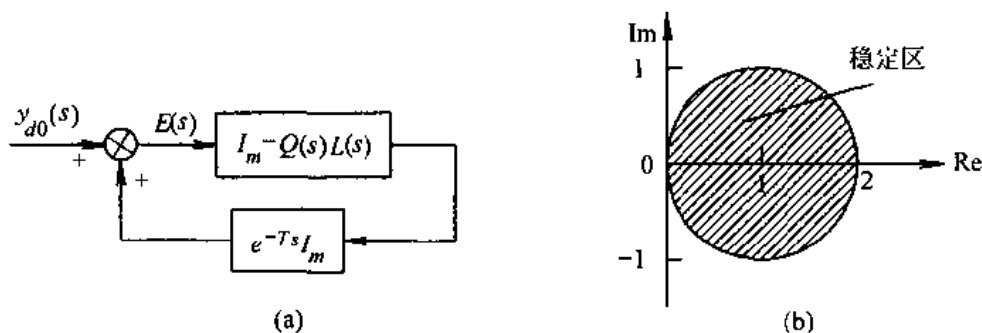


图 5.17 与图 5.16 等价的系统和稳定域

为了给出异步自学习控制系统的误差收敛条件或稳定条件,现考虑图 5.17(a)等价系统的有界输入有界输出稳定性。若对式(5.22)两边求 Laplace 逆变换,则得

$$y_{d0}(t) = y_d(t) - y_d(t - T)$$

易知,若期望输出 $y_d(t)$ 为周期是 T 的有界连续周期函数($t \geq 0$),则由上式可知

$$y_{d0}(t) = \begin{cases} y_d(t), & 0 \leq t \leq T \\ 0, & t \geq T \end{cases}$$

从而 $y_d(t)$ 为 L_2 函数即 $y_{d0}(t) \in L_2$ 。又由于 $Q(s) \in R_-$, $L(s) \in R_-$ 因而 $I_m - Q(s)L(s) \in R_-$, 又注意到 $\|e^{-Ts}I_m\|_\infty = 1$, 则由小增益定理可知,对于图 5.17(a)的等价系统,如果其闭环回路的增益小于 1,即

$$\|[I_m - Q(s)L(s)]e^{-Ts}I_m\|_\infty < 1$$

或

$$\|I_m - Q(s)L(s)\|_{\infty} < 1 \quad (5.24)$$

则必有式(5.21)成立。

(证毕)

在式(5.24)中,若令 $Q'(s) = Q(s)L(s)$,则相应的条件变为

$$\|I_m - Q'(s)\|_{\infty} < 1$$

对于单输入单输出系统,上式成为

$$|1 - q'(s)| < 1$$

这表明 $q'(s)$ 的 Nyquist 轨迹必须位于 s 平面以 $(1,0)$ 为圆心的单位圆内(如图 5.17(b)所示),方能保证其学习控制律的 L_2 稳定。

(2) 增加直接通道的情形

显然上述稳定性条件过于保守,其原因实质上是此类“连续”学习方法不再满足对于 $\forall k=1,2,\dots, e_k(0)=0$ 的假设,即不再具有每一学习周期初始“重定位”而造成的。为此可设想增加时域法中的“D”型异步自学习,即通过引入以 $E(s)$ 到 $U(s)$ 的直接通道来扩大稳定域范围。这时式(5.19)的异步自学习控制律成为

$$U_{k+1}(s) = U_k(s) + L(s)E_k(s) + E_{k+1}(s)$$

相应的系统框图如图 5.18 所示。

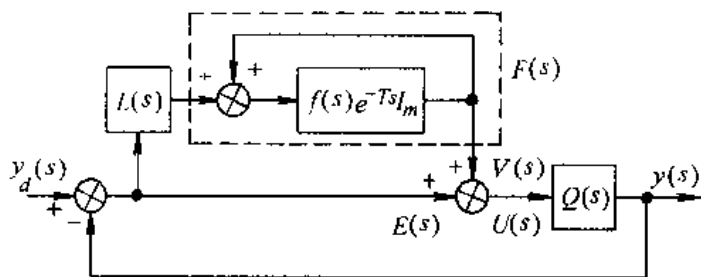


图 5.18 异步自学习控制频域法(I)

定理 5.3.9 考虑增加直接通道的异步自学习控制系统(图 5.18),其中暂令 $f(s)=1$,如果 (I) $[I_m + Q(s)]^{-1}Q(s) \in R_-$; (II) $\|[I_m + Q(s)]^{-1}[I_m + Q(s)(I_m - L(s))]\|_{\infty} < 1$, 则对有界连续周期信号 $y_d(t)$, 有

$$e(t) = L^{-1}[E(s)] \in L_2$$

这里, $Q(s) = [I_m + G(s)]^{-1}G(s) \in R_-$ 且 $L(s) \in R_-$

证明:由图 5.18 易知

$$E(s) = Y_d(s) - Y(s),$$

$$Y(s) = Q(s)U(s),$$

$$U(s) = E(s) + V(s),$$

$$V(s) = e^{-Ts}[V(s) + L(s)E(s)]$$

从而可得

$$E(s) = e^{-Ts}[I_m + Q(s)]^{-1}[I_m + Q(s)(I_m - L(s))]E(s) + [I_m + Q(s)]^{-1}Y_{d0}(s)$$

其中 $Y_{d0}(s)$ 的表示与式(5.23)相同,因而可作出其等价系统如图 5.19(a)所示。

由于

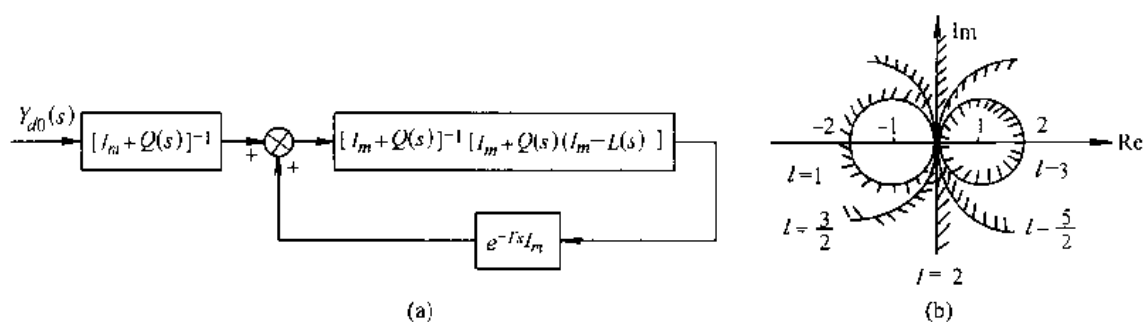


图 5.19 与图 5.18 等价的系统和稳定域

$$\begin{aligned} [I_m + Q(s)]^{-1} &= [I_m + Q(s)]^{-1} [I_m + Q(s) - Q(s)] \\ &= I_m - [I_m + Q(s)]^{-1} Q(s) \end{aligned}$$

由定理的条件(I)可知 $[I_m + Q(s)]^{-1} \in R_-$, 又 $L(s) \in R_-$, $Q(s) \in R_-$, 从而

$$[I_m + Q(s)]^{-1} [I_m + Q(s)(I_m - L(s))] \in R_-$$

这样由小增益定理, 对于 $y_d(t) \in L_2$, 必有 $e(t) \in L_2$. (证毕)

从这里可以看出, 上述定理中的条件(I)相当于要求由直接通道与被控系统 $Q(s)$ 组成的闭环系统稳定, 而条件(II)对单输入单输出系统也有如下的几何解释.

若令 $q^*(j\omega)q(j\omega) = \|q(j\omega)\|^2$, 则可推得 $q(j\omega) + q^*(j\omega) + [2 - l(s)]q^*(j\omega)q(j\omega) > 0$, 其中 $0 \leq \omega \leq \infty$, 相应的稳定域如图 5.19(b) 所示. 这时 $q(j\omega)$ 的 Nyquist 轨迹必须位于图中的阴影部分, 方能保证其异步自学习控制的 L_2 稳定. 特别地, 当 $l(s)$ 为实常数且 $l = 1$ 时, 上述稳定性条件实际上相当于最优调节或 kalman 滤波的最优条件.

(3) 增加直接通道和低通滤波器的情形

应该指出上述定理也有不足之处: 一是它只能保证 L_2 有界输入有界输出稳定, 而实际中更需知道 $e(t)$ 是否趋近于 0; 二是定理中条件(II)只有对直接通道, 即相对阶为 0 的系统才能严格满足, 否则当 $\omega \rightarrow \infty$ 时, 由于 $Q(j\omega) \rightarrow 0$, 条件(II)将不再成立.

上述问题的原因主要是因为要求异步自学习控制系统跟踪任意周期信号的想法不切实际. 事实上, 对任意高频分量的精确跟踪是可能的. 为此, 可以通过对 e^{-Ts} 引入一个低通滤波器, 即减少记忆系统的高频分量来放宽其稳定条件.

定理 5.3.10 考虑图 5.18 中增加低通滤波器 $f(s)$ 的异步自学习控制系统, 如果 (I) $[I_m + Q(s)]^{-1}G(s) \in R_-$; (II) $\|f(s)[I_m + Q(s)]^{-1}[I_m + Q(s)(I_m - L(s))]\|_\infty < 1$, 其中 $f(s) \in R^{1 \times 1}$ (对 $\forall \omega, f(j\omega) \neq 0$), $Q(s) = [I_m + Q(s)]^{-1}G(s) \in R_-$ 且 $L(s) \in R_-$, 则相应具有最小实现的系统为学习指数稳定, 并且 $e(t)$ 对任意周期性期望输出 $y_d(t)$ 有界. 特别地, 当 $L(s) = I_m$ 时, 必有 $\lim_{t \rightarrow \infty} e(t) = 0$.

推论 5.3.2 图 5.18 中 $L(s) = I_m$ 的异步自学习控制系统渐近稳定的充分条件为 (I) $[I_m + 2G(s)]^{-1}G(s) \in R_-$; (II) $\|f(s)[I_m + 2G(s)]^{-1}[I_m + G(s)]\|_\infty < 1$.

定理 5.3.10 的证明与定理 5.3.9 基本类似, 推论的证明也较容易, 此处从略.

2. 其他研究结论

考虑图 5.10 中 $L(s) \neq I_m$ 时的异步自学习控制系统,前已指出,它可理解为在一般的 $m \times m$ 阶闭环控制系统之上增加如图中虚线所示的异步自学习控制器构成的。这时闭环系统的传递函数矩阵为 $H(s) = [I_m + Q(s)]^{-1}Q(s)$,而由定理 5.3.10 可得如下推论:在与定理 5.3.10 相同的条件下,条件(I)可重写为 $\|f(s)[I_m - H(s)L(s)]\|_\infty < 1$ 。但这一条件一般只能得出 L_2 稳定的结论,为了研究渐近稳定,可以 DNA 法(直接乃氏阵列法)的角度考虑(H. H. Rosenbrock, 1974),导出一种使稳定域变窄的渐近稳定条件。

前面介绍的频域方法都有一定的局限性,一是仅适于周期性有界连续期望输出;二是稳定性条件比较保守(仅限于线性定常系统)。其中的原因实际上是在于“连续”学习的局限性,特别是这种“连续”学习,由于缺乏每一学习周期中的初始重定位所造成的,为此,可将记忆环节 $f(s)e^{-Ts}I_m$ 代之以存储式渐消记忆,从而使相应得到的“间断”学习频域方法既具有“连续”学习中带低通滤波器的渐消记忆,又具有“间断”学习中的递阶离线计算,或者说,作为前述方法的一个有效综合,它不但可使稳定域较宽,在线计算量小,而且具有物理意义明确的优点。

最后,作为一种统一理论的频域方法,本节给出的结果参考了重复自学习控制方法的研究,又从迭代自学习控制理论的多步迭代,“D”型学习和渐消记忆等方法中得到了启示。可以看出,由于异步自学习控制的理论方法使时域与频域中的学习概念趋于一致,两类方法相互渗透,这才使频域法的研究得到了新的结果。进一步研究的问题还应包括:多步异步学习(涉及高阶学习动态模型的建立)、学习动态特性(稳定条件和速度的进一步改善)、线性周期性时变系统的研究(利用广义传递函数矩阵的概念)等。

5.4 联结主义学习控制

所谓控制设计问题,实质上就是为开环系统选择一个控制律函数,使系统达到某个性能目标。为此,自然地要牵涉到从受控对象的实际输出和期望输出到控制作用之间的映射关系,以及其他有关的映射关系。因而,从根本的意义上看,控制设计问题也就是确定合适的函数映射的问题。而学习系统的功能在于它可用来在线地综合所涉及的各种函数映射,使系统表现出一定的智能。联结主义的机制是实现学习功能、形成学习控制系统总体结构的一种有效方法。联结主义学习控制的研究主要基于人工神经网络等技术,它代表了学习控制问题研究实践中一类重要的理论观点。

5.4.1 基本思想

1. 控制设计问题中的函数映射

从联结主义的机制看,“学习”是在某种输入刺激与期望的输出作用之间自动地生成一种“联结(association)”关系。如果用严密的数学方式来解释这种联结关系,“学习”就可以看作是一种自动综合多变量函数映射的过程,这一过程基于某种优化原则以及在时间上逐步获得的经验信息。

在一个控制系统中,表现为“输入刺激”与“期望输出作用”之间“联结”关系的有如下

几种函数映射。

(1) 控制器映射(图 5.20(a)),即从受控对象的实际输出 y_m 和期望输出 y_d 到一个合适的控制作用集 u 的映射

$$u = f(y_m, y_d, t)$$

(2) 控制参数映射(图 5.20(b)),即从受控对象的实际输出 y_m 到控制器某些参数 k (例如增益等)的映射

$$k = f(y_m, t)$$

(3) 模型状态(估计器)映射(图 5.20(c)),即从受控对象的实际输出 y_m 和控制作用 u 到系统状态的估计 x 的映射

$$x = f_x(y_m, u, t)$$

(4) 模型参数映射(图 5.20(d)),即从包括受控对象实际输出 y_m 和控制作用 u 的系统运行条件到精确的模型参数集 p 的映射

$$p = f_p(y_m, u, t)$$

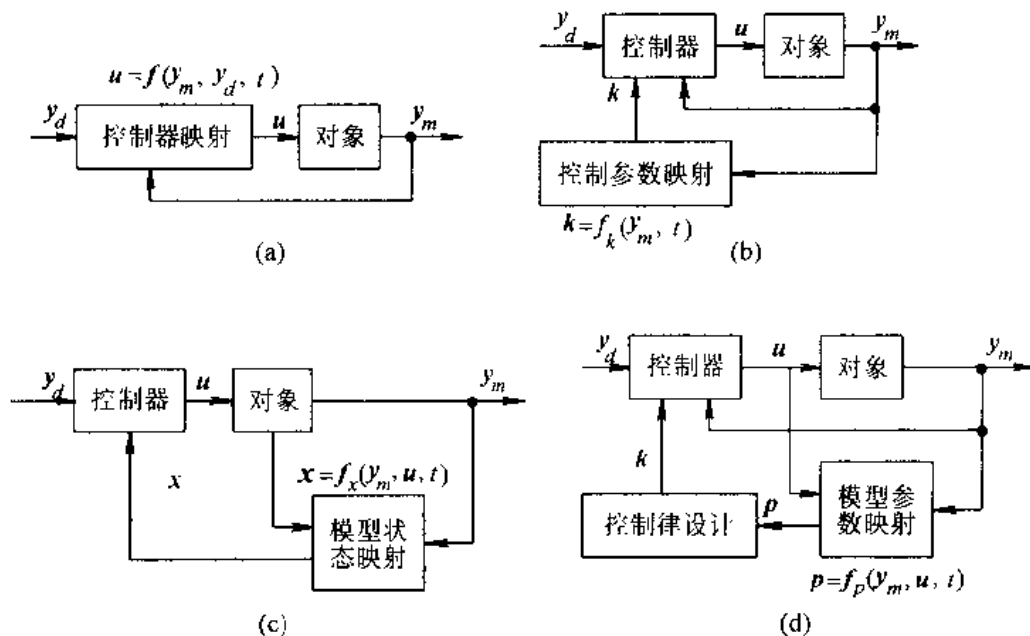


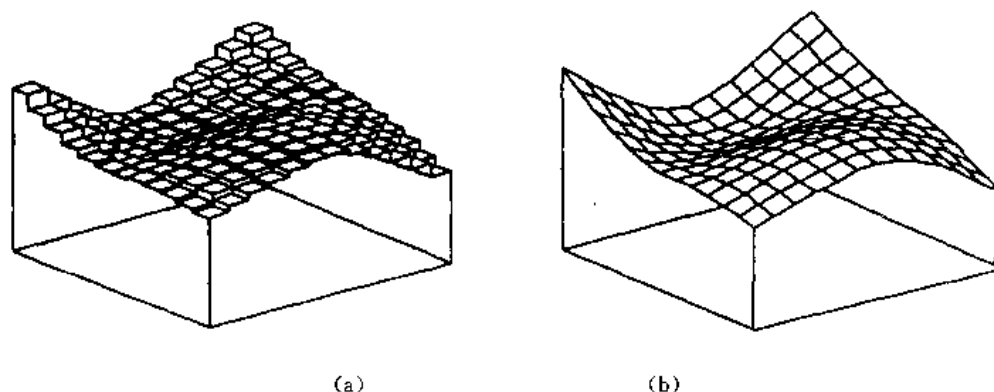
图 5.20 控制设计问题中的函数映射

上述映射关系一般应表示为动态函数(涉及到瞬时的微分或积分)。当这些映射关系由于先验不确定性(例如建模误差等)的存在而不能预先完全确定时,就需要学习的作用。在典型的学习控制应用中,所期望的映射关系是静态的,亦即并不显式地依赖于时间,因而可以隐含地表示为一种目标函数,它既涉及到受控象的输出,又涉及到学习系统的输出。这种目标函数为学习系统提供了性能反馈,学习系统通过性能反馈来指定映射关系中的可调整元素。系统中的各种映射关系是存放在存储单元中的,经过逐步修正和积累而形成改进系统性能的“经验”。

2. 学习系统的表示结构

一个学习系统必须能够积累和操作经验信息,存储和检索经过编辑的知识,并且修正所储存的知识以便提供新的经验。因此,学习系统需要一种有效的表示结构来保持经验知识。而且,学习系统的组织结构和运行属性需要得用定量和定性的先验设计知识来确定,其中包括可以在线度量的经验信息的预期特性等。

一种简单的方法是利用离散输入-模拟输出的函数映射作为学习系统的表示结构。如图 5.21(a)所示,将系统的输入空间分割成许多互不相交的区域。这样,当前输出的确定问题就是“查找”与当前输入区域关联的模拟输出。尽管这种表示结构的输出是连续的模拟量,但总的映射关系是不连续的。许多早期的学习控制系统实际上就是这种结构(例如基于模式识别的学习控制)。假定这种学习系统的输出直接用于控制作用,那么这样得到的就是一种非线性的控制律,因为通过对每一输入区域“学习”其合适输出,其结果只能是对真实的期望控制律的一种“阶梯式”近似。这种方法的缺点是,当输入空间的维数增加或每个维度的区域分割数增加时,学习系统所需要的区域数就会出现“组合爆炸”问题。



(a) 基于离散输入-模拟输出映射的“阶梯式”近似
(b) 基于连续映射结构的平滑近似

图 5.21

较为完善、精细的学习系统可以借助于一种连续函数簇的数学表示结构来加以实现,如图 5.12(b)。这种结构可以是固定的,也可以是可变的,它还能包含大量的自由参数。在人工神经网络等新型的学习系统中一般都采用这种结构。基于这种结构的学习过程就是自动地调整连续函数中的参数或函数本身的形式,从而达到期望的输入-输出映射。比较前述那种“查找”表的方法,连续函数簇的结构具有很重要的优点:首先,它可以有效地利用自由参数的个数来近似某种平滑的映射;而且,它能自动地提供学习经验的局部泛化作用。

5.4.2 联结主义学习系统的实现原理

1. 函数映射的综合

(1) 学习过程的基本步骤

定义 M 为学习系统的记忆器,同时又表示一种函数映射的法则;定义 D 为记忆器 M 的工作辖域。设有“局势” $x \in D$,则表达式 $u = M(x)$ 表示从记忆器中“调用”一个“响应” u 。

特别地,通过学习在记忆器 M 中存放的期望映射可以表示为 $M^*(x)$ 。

在学习控制系统中, x 实际上可表示受控对象的状态或输出,甚至更为一般的控制局势; u 可以直接表示控制作用,也可表示控制系统或受控对象模型的参数。不失一般性,在以下的讨论中,假设 x 就表示受控对象的状态(即状态空间中的一个点),或者受控对象状态的一个小集合(即状态空间中的一个小闭域);而假设 u 就相应于控制作用。

如果期望映射是显式已知的,那么学习过程的基本步骤为:

(I) 给定 x , 利用当前的映射 M 产生 $u = M(x)$;

(II) 比较 u 与 u^* 的差异, 其中 u^* 为期望控制作用,它可通过期望映射得到,即 $u^* = M^*(x)$;

(III) 修正映射法则 M , 以便减小 u 与 u^* 的差异。

如果期望映射 M^* 未知,那么上述步骤(I)可近似实现,即通过对映射 M 的期望值的估计直接得到 u^* ,或者通过对映射 M 的目标函数的梯度的估计来间接地求取 u^* 。

对于一类广泛而重要的学习控制系统来说,期望映射 M^* 预先就已知或者可假定为连续函数。因此,只要能把 M 近似为 M^* ,就也能把 M 同样表示为一种连续函数。于是通过参数化,映射 M 就可进一步表示为 $M = M(x; p)$, 式中 p 为参数向量。利用参数化方法,上述学习过程的步骤(III)的具体描述即为(III)修正映射法则 M , 形成 $\bar{u} = M(x; \bar{p})$, 使得 \bar{u} 比 u 与期望响应 u^* 的差异更小。其中参数向量 $\bar{p} = p + \Delta p$, 而 Δp 为适当调整参数向量 p 的待定变量。这样随着新的学习经验($\bar{u} = M(x; \bar{p})$)的取得,映射法则 M 就能逐步改善。

(2) 泛化

用参数化的方法来实现函数映射的综合,随着时间进展所获得的知识是分布式地存储在记忆器的参数空间中的。当原先在类似条件下得到的学习经验可以组合起来为当前局势提供合适的响应时,这种分布式的学习就显得非常有效。学习经验的组合运用过程扩展了每一个学习经验的范围和影响,称之为“泛化”。

泛化过程具有以下特点。首先它消除了记忆器中的“空白点”(未发生学习的特殊点);其次它限制了可能的输入-输出映射的集合,因为在大多数情况下相邻的输入局势将产生类似的输出(映射可成为平滑的或分段平滑的函数);最后,泛化造成了学习过程的复杂化,因为根据学习经验来调整映射关系不再是一种独立的、逐步进行的过程。

泛化是基于参数化连续映射函数综合方法的内在特征。在有限参数可调的情况下,每个可调参数都将影响非零量测区域上的映射。即如参数 $p_i \in p, p = (p_1, p_2, \dots, p_m)^T$ 得到调整,那么至少有一个 $M_i \in M, M = (M_1, M_2, \dots, M_n)^T$, 在参数 p_i 的整个作用区域上受到影响。这个作用区域可以用偏微分 $\partial M_i / \partial p_i$ (输入局势 x 的函数)来确定。这样,一条学习经验的效果将被自动地泛化,扩展到映射法则中任何 $\partial M_i / \partial p_i$ 为非零的部分。 $|\partial M_i / \partial p_i|$ 最大,则泛化效果最大,而 $|\partial M_i / \partial p_i|$ 很小或为零,则泛化效果很小或无效。 $\partial M_i / \partial p_i$ 被称为“敏感函数”。

(3) 梯度学习算法

对于参数化函数综合方法,还需要研究合适确定待定参数变量 Δp 的算法。如果映射法则是连续可微的,那么可以设定一个目标函数(代价函数),通过 J 的极小化来构造 Δp 。

这里可采用一种梯度学习算法

$$\Delta p = -W \cdot \frac{\partial J}{\partial p} \quad (5.25)$$

其中 $\partial J / \partial p$ 为目标函数 J 关于可调参数 p 的梯度向量(行向量), W 为加权矩阵(正定), 它确定了学习的速率。设用二阶 Taylor 级数在当前参数 p 处展开 J , 那么与二次目标函数 J 极小化相应的“最优学习速率”即为

$$W^* = H^{-1} = \left[\frac{\partial^2 J}{\partial p \partial p^T} \right]^{-1}$$

式中的 H 表示 J 的 Hessian 矩阵。上式成立的条件是矩阵 H 为正定。在线求逆 H 是困难的, 通常选取 W 为一速率系数 α , 即 $W = \alpha I$ 。

梯度学习算法还可以进一步表示。根据梯度行向量 $\partial u / \partial p$ 定义一个 Jacobian 矩阵 $\partial u^T / \partial p$, 从而有 $\partial J / \partial p = (\partial u^T / \partial p) \cdot x \bar{\omega} \partial J / \partial p$, 于是式(5.25)就变为

$$\Delta p = -W \frac{\partial u^T}{\partial p} \cdot \frac{\partial J}{\partial u} \quad (5.26)$$

式中目标函数 J 对于映射输出 u 的梯度 $\partial J / \partial u$ 取决于目标函数 J 的具体设定, 及映射输出 u 影响目标函数的方式(进而取决于控制系统结构中运用学习系统的方式); 而映射输出 u 对可调参数 p 的 Jacobian 矩阵 $\partial u^T / \partial p$ 则完全取决于映射 M 的近似结构, 因而是输入局势 x 的先验函数。应该指出, 输出梯度 $\partial J / \partial u$ 实际上就是提供给学习系统的性能反馈信息。 $\partial J / \partial p$ 所包含的信息量远大于目标函数 J 本身, 它既表明了 Δp 的方向, 又表明了 Δp 的大小(由于 $\partial u^T / \partial p$ 为先验已知)。

(4) 举例

设有二次目标函数

$$J = \frac{1}{2} \sum_E e_i^T e_i \quad (5.27)$$

式中 J 表示在一个有限求值点集 $x_i \in E = \{x_1, x_2, \dots, x_r\}$ 上极小化的代价, 而输出误差 $e_i = u_i^* - u_i = M^*(x_i) - M(x_i)$ 假定为已知。若 $W = \alpha I$, 则由梯度学习算法式(5.26)可得到规则

$$\Delta p = \alpha \sum_E \frac{\partial u_i^T}{\partial p} \cdot e_i$$

如果 J 是 p 的一个严格凸函数, 那么上述规则就能确定使 J 极小化的最优参数向量 p^* 。但在实际问题中, 期望输出 u^* 往往不是显式已知的, 这等价于输出误差 e 不能量测; 另外, J 可能是动态函数而非简单的静态函数; 再有, 定义在有限求值点集上的目标函数(式(5.27))通常不能直接用于在线学习控制。总之目标函数 J 实际上非常复杂。

如果 J 不是 p 的单峰函数, 那与所有基于梯度的优化技术类似, 对于上述目标函数也有可能收敛于一个局部极小值, 但必须解决两个问题: 首先, 学习控制系统的结构应该能够确定或者精确估计梯度 $\partial J / \partial u$; 其次, 目标函数应该是可调参数 p 的凸函数。

2. 联结主义学习系统

典型的联结主义学习系统具有网络结构的形式(例如人工神经网络), 它由节点以及节点间的联结弧组成。每个节点可以看作一个简单的处理单元, 其中包含若干可调参数

(对于节点的输入-输出关系不一定是线性的)。在这类联结主义学习系统中,常用的有“多层S形网络”和“径向基准函数网络”等。这些网络型式简单,适合于梯度学习方法,并能用并行计算的硬件来实现。例如在多层S形网络中,所谓“误差反向传播”就能有效地实现梯度算法,根据网络输出的误差平方来修改可调整网络参数。

在很多联结主义学习系统中都具有一种“通用近似性质”,即只要网络足够大,就能按给定精度近似任何连续函数。这种性质非常重要,但是并不能用来作为区分各种不同的近似结构的依据。从学习控制的目的看,更重要是学习发生于其中的环境。例如,为了选用合适的学习方法,必须认真考察学习系统运行过程中所能得到的信息内容以及信息的量和质,因为这种环境因素对于系统的性能具有重要的影响。

学习系统中可采用两种不同的学习策略:被动学习策略——学习的发生是机遇式的,在闭环系统正常运行过程中遇到什么信息就利用什么信息;主动学习策略——学习系统不仅试图沿着期望轨迹来推进受控对象的输出,而且要显式地设法改进学习系统所维持的函数映射的精度。为了实现这种策略,可以引入一种“探测”信号,它能把受控对象导向状态空间中那些学习不够充分的区域。对于被动学习策略,学习系统必须能够在闭环系统正常运行过程中进行在线量测和性能反馈。

3. 递增式学习

考虑受控对象的离散时间动态模型

$$\begin{cases} x_{k+1} = f(x_k, u_k) \\ y_k = h(x_k, u_k) \end{cases}$$

其中 $f(\cdot, \cdot)$ 和 $h(\cdot, \cdot)$ 为连续函数。对于这类系统,如果要求在线学习,那么式(5.27)给出的目标函数是不能直接利用的。主要问题在于学习系统所维持的映射的可能输入集不再是离散点的有限集,因此不容易选择一个典型求值点 z_i 的有限集 $E, z_i \in E$, 而且也不可能保证任何 z_i 都可访问。一般,学习系统的输入 z 是由连续集 $\{x, u, y\}$ 的量测值或估计值组成的。在实际中,可采用其他目标函数来近似式(5.27)。例如,可允许集合 E 在线扩充,以便包括所有遇到的 z_i , 即

$$E_k = \{z_1, z_2, \dots, z_k\} \quad (5.28)$$

如果可调参数 p 是线性地出现于 $\partial J / \partial p$, 其中 J 由式(5.27)给定, E 由式(5.28)给定,那么利用递归线性估计技术(例如RLS)就可求得最优参数向量 p^* (相应于特定集合 E)。但是在大多数联结主义网络中,总有某些(甚至全部)可调参数在 $\partial J / \partial p$ 中是非线性的,从而不能利用线性优化方法。而且,形如式(5.28)的求值集也难以用于非线性方法。

为能在线学习,学习控制系统中常用的目标函数是一种“点积函数”

$$J = \frac{1}{2} e^T e \quad (5.29)$$

式(5.29)可看作式(5.27)的特例,即求值集 E 在每一个采样时刻只包含一个点。这种以点积目标函数取代定义在连续集上的目标函数,通过极小化点积目标函数进行学习的算法被称为“递增式梯度学习算法”,它实际上与一类随机梯度算法有关。这种算法的要点是根据式(5.29),利用梯度的瞬时估计来近似式(5.27)中 J 的实际梯度 $\partial J / \partial p$ 。

最小均方算法(LMS)是一种广泛使用的随机梯度算法。LMS的参数调整律可表示

为

$$\Delta p = -\alpha \frac{\partial J}{\partial p}$$

式中 J 由式(5.29)给定。假设梯度 $\partial J/\partial p$ 是线性的、静态的,而且是 Gaussian 分布的随机变量情况,那么可以证明,LMS 算法可以按均值或均方值收敛,即

$$\lim_{k \rightarrow \infty} E(p_k) = p^* \text{ 或 } \lim_{k \rightarrow \infty} E(J_k) = J_{\text{次优}} > J_{\text{极小}} \quad (5.30)$$

这里,收敛的条件是学习速率系数 α (常量)满足有关 z 的相关矩阵特征值的某些要求,例如 α 不能太大等(S. Haykin, 1991)。式(5.30)中第一个极限式表明,当学习经验的个数 $k \rightarrow \infty$ 时,参数向量的期望值接近最优参数向量 p^* ;而第二个极限式表明,当 $k \rightarrow \infty$ 时,代价函数的期望值(均方误差)也接近一个极限,但不是最优的极小值。这时如果学习速率系数 α 能以一个特定比率(例如 $\alpha_k \sim 1/k$)逐步减小,那么参数向量本身(而非其期望值)也能收敛于它的最优值(A. Gelb, 1974),即有

$$\lim_{k \rightarrow \infty} p_k = p^*$$

尽管 LMS 算法的稳定性和收敛性条件只适用于一定假设下的线性网络,但 LMS 算法所体现的基本策略也能形成一种非线性网络的简单学习算法。在这种情况下参数调整律变为

$$\Delta p = \alpha \frac{\partial u^T}{\partial p} e \quad (5.31)$$

式中涉及的 J 由式(5.29)给出, $e = u^* - u$, 而提供给网络的性能反馈信号为 $\partial J/\partial u = -e$ 。式(5.31)表示了在线学习控制的标准的递增式梯度算法,它等价于递增式的“误差反向传播”(D. Rumelhart 等, 1986)。

4. 空间局域化学习

(1) 问题的提出

在闭环系统在线运行过程中进行学习是受限制的,这些限制将影响学习系统的网络结构、学习算法以及训练过程。例如在一个基于被动学习策略的系统中,由于对象状态(及输出)受制于系统的动态特性,因而学习经验(训练样本)并不能自由选取,另外,期望的对象输出也受制于具体的控制设计而并不取决于学习。在这些限制条件下,系统的状态只是限于状态空间中的一个小区域(例如工作点附近),这就意味着递增式学习所用的量测值 z 也只能处于映射的输入辖域的一个小区域内,形成一种“固定”态势。如果基于递增式学习算法的参数调整对于学习系统维持的映射具有一种非局部化的效应,那么上述那种“固定”态势将导致并不期望的副作用。

举例来说,如果反复调整一个参数,以便对输入辖域中某个特定小区域内的映射函数进行改进,而这个参数具有非局部化效应,这就会造成其他区域中的映射的变质,甚至可能“擦除”原先发生过的学习。出现这种副作用的原因在于,由递增式学习算法决定的参数调整是按单一求值点的原则进行的,而没有考虑映射的其他部分。递增式学习算法一般还存在一种共同的问题:对于可调整参数可能形成一些冲突要求,例如,使目标函数 J (式(5.29))在某个输入点 z_i 上极小化的 p_i^* 一般不同于在另外某个 z_j 上使 J 极小化的 p_j^* 。被动学习控制系统中的上述特异性就是所谓空间局域化学习要研究的问题。

(2) 空间局域化学习的性质

空间局域化学习的基本思想是,如果在一个可调整元素子集与空间中某个局域之间能得出清晰的联结关系,那么学习就比较方便,可通过对基于递增式梯度学习算法的学习系统明确一些期望品质来解决特异性问题。具体作法是利用“敏感函数” $\partial M_i/\partial p_j$ (见本节有关泛化的讨论)。在映射的输入辖域中每个点 x 上,希望能具有如下性质:

(1) “有效区”性质:对于每一个 M_i 至少存在一个 p_j ,使得在 x 的邻域中 $|\partial M_i/\partial p_j|$ 相对较大;

(2) “局域化”性质:对于所有的 M_i 和 p_j ,如果 $|\partial M_i/\partial p_j|$ 在 x 的邻域中都相对较大,那么 $|\partial M_i/\partial p_j|$ 必须在其他区域中都相对较小。

明确了上述性质后,递增式梯度学习算法仍然可以在映射的整个输入辖域上得到支持,但是效果将限于每个学习点的邻域这样一个局部区域内。从而在某个局部区域内的学习经验以及后续的学习,对于已经在映射其他部分形成的知识的依赖性是有界性的。而且由此还可减少对于可调参数的冲突要求问题。

有些学习系统一般都具备上述空间局域化学习的性质,例如 BOXES (D. Michie 和 Chambers, 1968), CMAC (J. Albus, 1975), “径向基准函数网络” (T. Poggio 和 F. Girosi, 1990), “局域基准-影响函数网络” (W. Baker 和 I. Farrell, 1990) 等,而像“随遇 S 形(感知器)网络”就不具备这种性质。为解决非局域化学习和冲突参数修正存在的问题,对于各种 S 形网络也可采取一些简单措施,包括局部分批学习,降低学习速率,分布式输入序列,及随机化输入缓冲器等(可见 L. Baird 和 W. Baker, 1990)。

(3) 举例

下面以“线性-Gaussian 网络”(一种“局域基准-影响函数网络”)为例来说明空间局域化学习。这种网络依靠“局域基准函数”和“影响函数”的组合节点单元,在量化学习系统的空间局域化学习性质之间实现一种折衷。完整的网络映射是由“基准函数” $f_i(x)$ 的集合构成的, $f_i(x)$ 仅运用输入空间的空间局部化区域;“影响函数” $\gamma_i(x)$ 与“基准函数” $f_i(x)$ 一对一地相结合,用于描述每一个局域基准函数 $f_i(x)$ 在输入空间的辖域(又称“影响域”)。即相对于输入空间的某个点 x^0 ,每一个影响函数 $\gamma_i(x)$ 都被定义为非负函数,它们在 x^0 处有最大值,而在远离 x^0 的所有点 x 处都趋于零。总的输入-输出关系可表示为

$$y(x) = \sum_{i=1}^n \Gamma_i(x) f_i(x)$$

其中 $\Gamma_i(x)$ 为规范化影响函数,定义为

$$\Gamma_i(x) = \frac{\gamma_i(x)}{\sum_{j=1}^n \gamma_j(x)}, \quad 0 \leq \Gamma_i(x) \leq 1, \text{ 且 } \sum_{i=1}^n \Gamma_i(x) = 1$$

这样,网络中每一个可调参数对于总的映射的影响就仅限于规范化影响函数 $\Gamma_i(x)$ 所规定的输入区域,因而前而提到的“固定”态势问题也就被避免了。但要指出,(局部)泛化是网络的固有属性,标准的递增式梯度学习算法仍然是可用的。

为了进一步说明上述基本概念,假定“局域基准函数”为带有偏置的线性函数,而“影响函数”为 Gaussian 函数。在这种“线性-Gaussian 网络”中,就有

$$f_i(\mathbf{x}) = M_i(\mathbf{x} - \mathbf{x}_i^0) + b_i,$$

$$y_i(\mathbf{x}) = C_i \exp\{-(\mathbf{x} - \mathbf{x}_i^0)^T Q_i (\mathbf{x} - \mathbf{x}_i^0)\}$$

其中,对网络中每个节点对 i , 矩阵 M_i 和 Q_i (正定), \mathbf{x}_i^0 向量和 b_i , 以及标量 C_i 都是可调整 \mathbf{x}_i^0 的。向量表示线性-Gaussian 对可用的局域, 即总的映射在 \mathbf{x}_i^0 的邻域中被近似为 $f_i(\mathbf{x})$ 。由于这种结构上的唯一性, 很容易为每一个参数以及网络总体结构赋予物理含义。因而在这种网络中很容易结合利用先验知识和问题的局部解 (例如相应于 $f_i(\mathbf{x})$ 的线性控制点的设计)。实际上, 之所以选用线性函数作为局域基准单元, 就是因为它比较简单, 而且对惯用的增益调度映射是兼容的 (如果能先验已知期望映射的局域函数结构, 就还可选取更合意的局域基准单元)。另外, 由于这种结构, 网络还允许采用在线的变结构学习方式, 即通过把节点单元加入或移出网络来实现更精确、更有效的学习。

图 5.22 表示了一个简单的“线性-Gaussian 函数网络”。其中有 5 个局域基准影响函数节点对, 图中下部是影响函数, 它与图中上部的线性局部函数是分开画的, 这样在总的输入-输出映射中就能清晰可见。

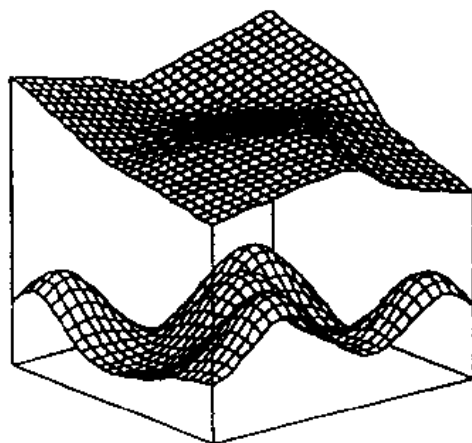


图 5.22 线性-Gaussian 网络映射 ($R^2 \rightarrow R$)

(4) 局域化方式

空间局域化网络的学习算法使用局域化技术的方式及其优点有以下两个方面。其一, 空间局域化意味着在每个时刻仅有节点单元的一个小子集 (因而有可调参数的一个小子集) 对网络的映射产生大的效果。这样, 网络输出的计算以及网络参数的修改都更有效 (其他作用效果不大的节点单元可以忽略)。例如“线性-Gaussian 网络”中可仅用那些规范化影响函数值较大的节点单元, 这些单元的影响函数值等于或超过某个预定的阈值 (例如 0.95)。这是一种“稀疏”计算问题的方法, 它在序贯计算硬件实现网络的情况下可以极大地增加网络的吞吐量。其二, 由于系统状态可以一直限于状态空间中的特定区域, 因而近似误差将不会一致地趋于零, 这正是所希望的。否则, 在学习发生量最大之处误差就最小, 这样将产生对学习速率的冲突要求问题。即在近似误差较小的区域内, 噪声的滤波效果应该比较小, 而在近似误差大的区域内进行快速学习时, 这种滤波效果应该比较大。学习速率的空间局域化方法就是解决这种冲突要求问题的有效途径, 即对每一个 (空间局域

化)的可调参数维持各别的学习速率系统,而根据局部学习条件的不同加以修正。在这种情况下,加权矩阵 W (见式(5.25))将随时间而异。

空间局域化网络对于计算方面的存储需要介于非局部化联结主义网络与离散输入-模拟输出映射结构之间。由于只要求每个参数对总的映射局部生效,因此应该多增加一些参数,使它在精度方面能与非局域化方法相比拟。尽管如此,在学习控制系统中,训练速度和近似精度方面的要求应该优先于存储需要,因为比起不精确、不合适的控制作用来说,存储要求的代价一般是比较低的。

5.4.3 联结主义学习控制系统的结构

兼具自适应性能和学习功能的混合控制结构可形成一种重要的联结主义学习控制系统。在这种方案中,一个自适应系统与一个联结主义学习系统相配合,对新控制局势和缓慢的时变动态特性提供自适应性能,自适应过程与学习过程相结合,调节系统的稳态的或准稳态的状态空间属性(例如无记忆的非线性)。自适应系统的特点是对受控对象的期望行为与实际行为之间的差异作出反应,从而维持所要求的闭环系统性能。出现这些差异可能是由于时变的动态特性、干扰、或者未建模动态特性。实际中,对于时变动态特性和干扰很难有什么预定的办法,在自适应系统中通常利用反馈来解决。相反,某些未建模动态特性的影响(特别是静态非线性)倒可以根据以往的经验来预示,这就是学习系统的任务。开始时,所有的未建模动态特性问题可由自适应系统处理,而最终,学习系统就能对有过经验的,起初未建模的特性采取措施。这样,自适应系统可以专注于新的控制局势(未经学习或学习很少)以及缓慢的时变特性。

后面将概述两种一般的混合结构,它们分别与传统的“直接自适应控制”和“间接自适应控制”策略相对应。这两种方式的自适应控制器都要对于受控对象的动态特性中可预示的状态空间属性连续地作出反应,而混合结构中的学习系统就用于减轻自适应控制器的这种负担。为了保证混合结构方法的成功实现,必须考虑全面的技术问题。例如,要保证闭环系统(包括自适应系统、学习系统以及受控对象)的稳定性和鲁棒性就必须考虑能控性和能观测性的问题;另外还有噪声效应,干扰、模型阶次误差以及其他不确定性的问题;参数收敛性、激励的充分性及非稳态的问题;计算需求、时延、运算精度效应的问题等等。

1. 直接自适应-学习结构

在图 5.23 所示的结构中包括了典型的直接自适应控制方法。产生每一个控制作用 u 的依据是,对象输出的实际量测值 y_m 和期望值 y_d ,控制器的内部状态及适当的控制律参数的估计值 k 。控制律参数的估计值在每一个时间步都要根据对象输出 y_m 与参考系统的输出 y_r 的误差 e 来加以修正。当然所选择的参考系统的性能要保证对象实际上是能达到的。直接自适应控制并不依赖于显式的对象模型,因而可避免进行在线的系统辨识。

图 5.23 中,如果学习系统未被实现,控制器也能产生正常的自适应操作。参考系统表示了控制器及对象的总的期望特性,而自适应机制则用于把参考误差 e 直接转换为当前的控制系统参数的修正值 Δk 。自适应算法可以由多种不同方式来实现,例如梯度法,基于 Lyapunov 的方法等。学习系统可以把所需要的控制系统参数作为对象运行状况的函数而加以存储、记忆。学习系统也可以用来把合适的控制作用作为实际的和期望的对象输出

的函数而加以存储、记忆。图 5.23 所示为前一种情况。

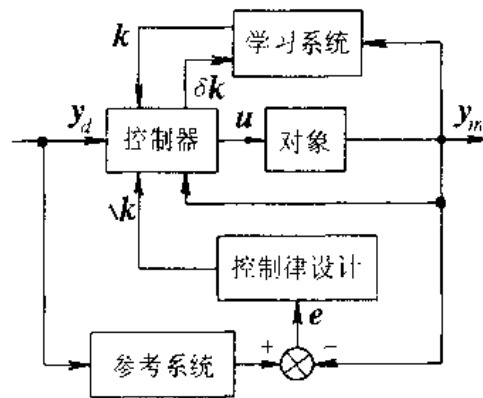


图 5.23 直接自适应-学习结构

当学习系统用于存储控制系统参数时,自适应系统将为学习系统产生的控制参数 k 提供任何需要的扰动 δk ,它与原先出现过的运行状况相关联。这种关联过程即为递增式学习过程,其作用就是把自适应系统产生的估计与针对原先状况已经学习得到的控制参数加以结合。在每一个采样时刻,学习系统都产生一个与当时状况相关联的估计 k ,然后把它传送给控制器。在控制器中,由自适应系统保持的扰动参数估计值与 k 相结合,可用来产生控制作用 u 。如果学习是完全的,且没有噪声、干扰和时变动态特性的影响,那么学习系统总能提供正确的参数值。于是,自适应系统产生的扰动 δk 和纠正量 Δk 将变为零,系统结构则类似于增益调度法的情况。

当学习系统用于存储控制作用时,自适应系统将为学习系统产生的控制作用提供任何所需要的扰动。但要注意,如果期望有一种动态的反馈律,那么就必须由学习系统来综合一种动态的映射。比起前面介绍的那种方法,这里介绍的学习方法可以学习更为一般的控制规律,但是它的缺点是需要增加存储量,学习问题也变得更为困难。

2. 间接自适应-学习结构

图 5.24 所示的结构与典型的间接自适应控制方法相对应。其中每个控制作用 u 的产生依据是对象的实际量测输出 y_m 和期望输出 y_d ,控制器的内部状态,及一个局部对象模型的估计参数 p_e 。 k 为一种局部控制律的参数,是根据观测到的对象特性在线地显式设计出来的。如果对象特性有变化(例如由于非线性),自适应估计器就尽快地自动修改它的对象模型,依据是从输出的量测值(一般有噪声)得到的信息。间接自适应方式的重要优点是具有很强的设计方法(包括最优控制技术),且有可能在线运用这些方法。但要注意间接自适应所需的计算量常常很大,因为模型辨识和控制律设计都是在线完成的。

如图 5.24 中的学习系统未被实现,这种结构就表示了传统的间接自适应控制系统。信号 p_e 是对象模型参数的自适应估计,它用于计算控制律参数 k 。在这种结构中加入学习系统后,就可把对象模型参数作为对象运行状况的函数而进行学习。学习系统产生的模型参数可以使原先经历过的对象特性得到预示。在这种情况下,学习系统的输出 p_i 是一种与当前运行状况相关联的模型参数的先验估计,而估计器所产生(涉及到滤波和后验平

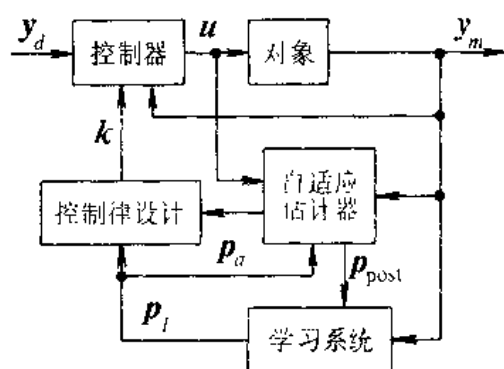


图 5.24 间接自适应-学习结构

滑操作)的后验参数估计 p_{post} 则用于修改学习系统中存储的映射。整个系统利用学习系统和自适应系统产生的估计值来执行控制律设计,并确定合适的控制律参数。如果设计过程比较复杂、费时,那控制律参数也可作为对象运行状况的函数而被存储记忆,但这要借助学习系统中另外的映射。这样控制律的设计可以较低的速率完成,为此要求学习系统维持的控制律参数映射能有足够的精度,否则仍要快速设计才能提供合理的控制作用。

3. 小结

在上述两种混合结构中,学习系统仅包含根据设计模型导出的知识。在闭环运行之初,自适应系统用于解决先验设计知识不充分的问题。其后随着对实际对象的经验逐步积累,学习系统就用来预示合适的控制参数或模型参数,把它们作为当前对象运行状况的函数。而自适应系统则主动地处理新的控制局势,处理学习系统的局限性(精度有限等)。如果无噪声,无干扰,对象没有时变特性,且学习是完全的,那么自适应系统的作用最终将变为“零”;而如果存在噪声和干扰,那么自适应系统的贡献虽然不是“非零”,但是会变得很小。总之,在一般的情况下,混合结构的功能比两个系统单独作用都好。正如前面指出的,自适应与学习是互补的,它们可以按协作的方式同时使用。

5.4.4 研究课题

作为智能控制的重要类型,学习控制的主要特点是,它可以通过与实际对象的在线交互来解决不确定性问题,可通过在线的自优化来改进系统的效率和性能。对于已有成果的力度及其工程实用性,特别在基本理论的发展等方面,学习控制还有待继续研究。以下提出的是对已有学习控制系统进一步研究的课题。

(1) 递增式函数综合。表达能力——需要多大的网络就可以充分表示期望的映射?表达效益——需要什么计算资源就可以实现这种方法?可调参数的稳定性和收敛性能在任何条件下得到保证吗?快速性又如何?

(2) 变结构学习。联结主义网络的表达能力和计算需求在很大程度上是由它的结构决定的。如果结构是先验地确定的而且是固定的,那么实际设计时很容易导致“过度”的保守设计,造成资源的无效使用。于是,就可能分配了过少的资源去近似期望映射的复杂部

分,从而限制了近似精度;也可能分配了过多的资源去近似映射的相对简单的部分,从而出现计算供需的过剩。变结构学习方式确是解除这种困境的一条途径,但是对于何时、何处,如何改变学习结构的问题仍然有待研究一些有效的规则。

(3) 自适应与学习的配合。在混合结构中,或者必须将自适应系统和学习系统得到的控制系统参数加以结合(直接式),或者要求模型参数估计值的结合先于在线的控制律设计。是否存在某种最优方式来完成这种“融合”?如果考虑象 Kalman 滤波这类最优线性估计技术,那么很自然地会想到:自适应估计和学习估计的任何合理混合都取决于对两种估计的“品质”(例如误差的协方差)的量测。那么,如何来表达并维护与学习系统当前状态相关联的这种“品质”呢?

(4) 高级学习。要对整个运行系统都能完成的实际目标函数事先加以详细说明,这是一个难题。因此可以研究的是:在线调整目标函数,以便在可能之处增强系统的性能,而在必要之处则减轻系统的负担。还可以研究把规划技术和探测技术用于实现控制和学习的双重任务,这样就能保证在整个运行期间都能得到充分的训练。而这类高级学习将会牵涉到更为复杂的优化问题。

参 考 文 献

- [1] Fu K S. Learning Control System and Intelligent Control Systems: An Intersection of Artificial Intelligence and Automatic Control, IEEE Trans. Automatic Control, 1971
- [2] Fu K S. Learning Control Systems—Review and Outlook, IEEE Trans. Automatic Control, 1970
- [3] 维纳 N. 控制论. 科学出版社, 1962
- [4] Glorioso R M. Engineering Cybernetics, Prentice-Hall, Inc., 1975
- [5] Tsytkin Y Z. Adaptation, Learning and Self-learning in Automatic Control Systems, Proc. 1966 IFAC Cong., London, England, 1966
- [6] Saridis G N. Self-organizing Control of Stochastic Systems, Marcel Dekber Inc., New York and Basel, 1977
- [7] Baker W L, Farrell J A. An Introduction to Connectionist Learning Control Systems, in Handbook of Intelligent Control. White D A, Sofge D A, Eds. New York: VNR, 1992
- [8] Sklansky J. Learning Systems for Automatic Control. IEEE Trans. Automatic Control, 1966
- [9] 涂序彦, 郭荣江. 智能控制及其应用. 自动化, 1977
- [10] 杨忠祥. 机器学习的发展现状与动向. 信息与控制, 1987
- [11] Narendra K S, Streeter D N. A Self-organizing Control System Based on Correla-

- tion Techniques and Selective Reinforcement, Cruft Lab. , Harvard Univ. , Cambridge, Mass. , Tech. , 1962
- [12] Smith F W. Contact Control by Adaptive Pattern-Recognition Techniques, Stanford, Calif. , Tech. , 1964
 - [13] Smith F B, Jr, *et al.* Trainable Flight Control System Investigation, Wright- Patterson Air Force Base, Dayton, Ohio, Tech. , 1964
 - [14] Butz A R. Learning Bang-Bang Regulators, Proc. 1968 Hawaii Intern. Conf. Sys. Sci. , Hawaii, 1968
 - [15] Mendel J M, Zapalac J J. The Application of Techniques of Artificial Intelligence to Control System Design, in Advance in Control Systems. Lecondes C T. , ed. , New York; Academic Press, 1968;6
 - [16] Waltz M D, Fu K S. A Heuristic Approach to Reinforcement Learning Control Systems, IEEE Trans. Automatic Control, 1965
 - [17] Mendel J M. Applications of Artificial Intelligence Techniques to a Spacecraft Control Problem, Douglas Aircraft, Sanfa Monica, Calif. , Tech. , 1966
 - [18] Fu K S. A Class of Learning Control Systems Using Statistical Decision Functions, Proc. 1965 IFAC Symp. of Theory on Self-Adaptive Control Sys. , Teddington, England, 1965
 - [19] Tsypkin Y Z. Self-learning—What Is It? IEEE Trans. Automatic Control, 1968
 - [20] Riordon J S. An Adaptive Automation Controller for Discrete-time Markov Processes, Proc. 1969 JACC, Boulder, Colo. , 1969
 - [21] Arimoto S, Kawamura S. *et al.* Bettering Operation of Robots by Learning, Robotics Systems, 1984
 - [22] Hara S, Yamamoto Y. Stability of Repetitive Control Systems, Proc. 24th CDC, 1985
 - [23] 邓志东. 自学习控制理论与应用. 哈尔滨工业大学博士学位论文, 1991
 - [24] Wee W G, Fu K S. A Formulation of Fuzzy Automata and its Application as Model of Learning Systems, IEEE Trans. Sys. Sci. and Cyb. , 1969, 25
 - [25] Graham J H, Saridis G N. Linguistic Decision Structures for Hierarchical Systems, IEEE Trans. Syst. , Man, Cybern. , 1982, 12(3)
 - [26] Anderson B D O, Moore J B. Linear Optimal Control. Prentice-Hall, 1971
 - [27] Haykin S. Adaptive Filter Theory. 2nd ed. , Prentice-Hall, 1991
 - [28] Michie D, Chambers. BOXES: An Experiment in Adaptive Control, in Dale E, Michie D, eds. , Machine Intelligence. Oliver and Boyd, 1968, 2
 - [29] Albus J. A New Approach to Manipulator Control; The Cerebellar Model Articulation Controller (CMAC), ASME Journal of Dynamic Systems, Measurement,

and Control, 1975,11

- [30] Poggio T, Girosi F. Networks for Approximation and Learning. Proc. IEI, 1990, 11
- [31] Baker W L, Farrell J A. Connectionist Learning Systems for Control. Proc. SPIEOE, Boston'90, November, 1990
- [32] Baird L, Baker W L. A Connectionist Learning System for Nonlinear Control. Proc. 1990 AIAA Conf. on Guidance, Navigation and Control, 1990

第 6 章 分层递阶智能控制

6.1 一般结构原理

为了实现规划、决策、学习等智能功能,智能控制所实现控制的含义要比常规控制广泛得多。广义的控制可以定义为:驱使系统实现要求功能的过程。为了实现广义的控制功能,智能控制需要将认知系统研究的结果与常规的系统控制方法加以有机的结合。

认知系统传统上是作为人工智能的一部分,它主要实现类似于人的一些行为功能,如声音识别和分析、图像和景物分析、数据库组织、学习和高层决策等。这些功能主要是基于从简单的逻辑操作到高级的推理方法来实现的。这方面已经取得了很大的进展,如模式识别、语言学以及启发式方法等已经作为认知系统的一部分,广泛地用于声音、图像及其它传感信息的分析和分类。系统控制方面也已经建立起许多成熟的理论和方法。它们可用于来进行运动控制和轨迹跟踪、动态规划及优化控制等。

G. N. Saridis 等提出了一种分层递阶智能控制系统理论,它将计算机的高层决策、系统理论中的先进的数学建模和综合方法以及处理不精确和不完全信息的信息学方法结合在一起,形成了一种适合于工程需要的统一的方法。该理论可认为是三个主要学科领域的交叉:人工智能、运筹学和控制理论。本章主要介绍这方面的内容。

分层递阶智能控制系统的结构如图 6.1 所示。它由组织级,协调级和执行级三个层次组成,并按照自上而下精确程度渐增、智能程度递减的原则进行功能的分配。如图 6.2 所示为一个典型的智能机器人的分层递阶结构。

智能控制系统上层的作用主要是模仿人的行为功能,因而主要是基于知识的系统。它所实现的规划、决策、学习、数据的存取、任务的协调等主要是对知识进行处理。智能控制系统下层的作用是执行具体的控制任务。它主要是对数值进行操作和运算。

在组织级,Moed 和 Saridis 提出了一种用 Boltzmann 机神经网络来实现推理、规划和决策的方法。在协调级,Wang 和 Saridis 提出了 Petri 网翻译器方法来实现语言决策的功能。在执行级,Saridis 提出了一种对于系统控制问题用熵进行测试的方法,从而为智能控制系统提供了一种统一的用熵函数来表示的性能测度。因而可以用数学规划的方法来设计和求解智能控制系统的最优操作和控制问题。

分层递阶智能控制的理论可以表述为:对于自上而下按照精度渐增、智能递减的原则所建立的分层递阶结构的系统,智能控制器的设计问题可以认为是这样的数学问题:寻求正确的决策和控制序列,以使整个系统的总熵最小。



图 6.1 典型的分层递阶结构

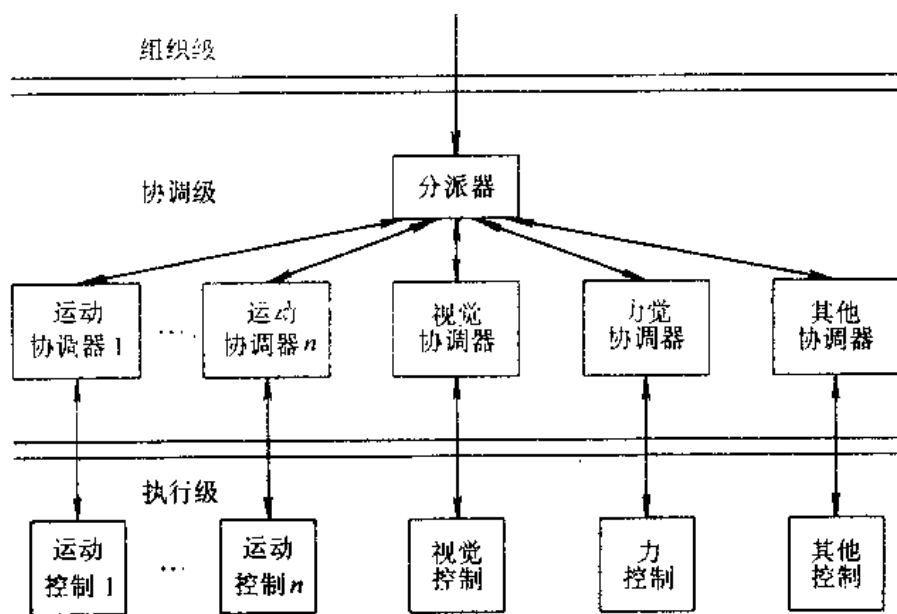


图 6.2 机器人分层递阶智能控制系统

为了对智能控制系统进行分析,有必要对其中常用的名词术语,如知识、智能、精确等作进一步的解释和说明。

机器知识(machine knowledge)定义为用来消除智能机执行任务的不确定性所需的结构性信息。

由于知识在机器内是逐渐增长的累积量,所以不适于用作执行任务的一个变量。而下面要定义的机器知识(传输)率则是一个比较合适的变量。

机器知识率(rate of machine knowledge)定义为通过智能机的知识流量。

一般情况下定义智能为获取和应用知识的能力,而对于机器智能可进行如下修正:

定义机器智能(Machine Intelligence MI)为行动或规则的集合,它们作用于事件数据库(Data Base DB)而产生知识流。

另一方面,关于精确和不精确可作如下的定义:

不精确(imprecision)是智能机执行各种任务的不确定性。而精确(precision)则是不精确的补,它代表了过程的复杂性。

下面进一步分析上述各量的关系。设 $P(K)$ 为知识的概率密度函数,则根据概率的性质(取值 0 到 1 之间,概率之和为 1),可设

$$P(K) = \frac{e^{-K}}{\int_x e^{-K} dx}$$

令

$$\alpha = \ln \int_x e^{-K} dx$$

可得

$$P(K) = e^{-\alpha K} \quad K = -\alpha - \ln P(K)$$

对于固定的时间间隔 T , 知识率 R 可定义为

$$R = \frac{\Delta K}{T}$$

对于前面机器智能的定义, 可写成如下的关系:

$$(MI); (DB) \rightarrow R$$

它表示机器智能作用于数据库产生知识率。上式也说明较高的智能作用于较小的数据库, 或者较低的智能作用于较大的数据库可产生相同的知识率。它也进一步说明了精度渐增, 智能逐减的原理。

6.2 组 织 级

组织级是分层递阶智能控制系统的最上面一层。它的作用是针对给定的外部命令和任务, 设法找到能够完成该任务的子任务(或动作)组合。再将这些子任务要求送到协调级, 通过协调处理, 最后将具体的动作要求送至执行级完成所要求的任务。最后对任务执行的结果进行性能评价, 并将评价结果逐级向上反馈, 同时对以前存储的知识信息加以修改, 从而起到学习的作用。

由此可见, 组织级的作用主要是进行任务规划, 它是典型的人工智能中的问题求解, 已有很多 AI 专家在这方面做了大量工作。这里介绍一种由 Moed 和 Saridis 所提出的 Boltzmann 机神经网络(以下简称 BM 网络或 BM)来实现组织级功能的方法。

为了便于对问题进行描述, 需要定义一组基元事件的集合 $E = \{e_1, e_2, \dots, e_n\}$, e_i 可以表示基本动作、动作对象、动作结果等, 它们是最基本的事件。这些基元的组合既可以表示外部的任务输入要求, 也可表示子任务的组合。在 BM 中, e_i 表示了神经网络的结点。BM 网络由如下 3 部分结点所组成:

(1) 输入结点。它用来表示要求的目标或子目标。在这里外部输入命令即是要求的目标。

(2) 输出结点。它由基元事件组成。这些基元事件的适当组合可实现要求的目标。

(3) 隐结点。它主要用来实现输入和输出结点之间复杂的连接关系。

对于每个结点都用一个二进制随机变量 $x_i = \{0, 1\}$ 来表示, 并令 $P(x_i = 1) = P_i$, $P(x_i = 0) = 1 - P_i$ 。其中 1 表示神经元结点处于激发状态, 0 表示结点处于闲置状态。网络的状态向量 $\mathbf{x} = (x_1, x_2, \dots, x_i, \dots, x_n)$ 表示了一组 0 和 1 的有序组合, 它描述了 BM 网络的状态。对于给定的输入, 当 BM 网络到达稳定状态时, 抽取相应输出结点的状态便可获得最优的执行特定任务的基元事件的有序组合。

标准的 BM 网络应用能量作为代价函数, 通过使其极小来找到最优的状态。如果将能量与知识联系起来, 那么这里能量的含义是表示缺乏知识的程度, 即能量的减少表示知识的增加。换一种说法, 这里能量是与不确定性的程度相对应的。即能量减少, 不确定性的程度也减少, 并可由它表示在给定任务要求下所得到的基元事件组合的概率, 并可进一步计算出熵函数。

Boltzmann 机事先必须进行学习和训练。学习时必须给出一组样本,每一个样本包括以下 3 部分内容:(1) 输入的任务要求,它由输入结点的状态来表示。(2) 输出的基元事件组合,它由输出结点的状态表示。(3) 该输入输出对的概率,它实际上反映了在这组约束条件下 BM 网络的能量。

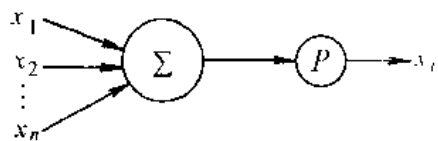


图 6.3 BM 中的一个神经元

BM 中的各个神经元之间互相连接,其中单个神经元的特性可用图 6.3 来描述。

对于第 i 神经元,其输入总和为

$$s_i = \sum_j w_{ij} x_j$$

神经元的输出为 x_i , x_i 只能取 1 或 0, x_i 取 1 的概率由下式决定

$$P_i = \frac{1}{1 + e^{-s_i/T}}$$

定义 BM 的能量函数为

$$K(\mathbf{x}) = \frac{1}{2} \sum_i \sum_j w_{ij} x_i x_j$$

输出基元事件组合是正确的概率为

$$P[K(\mathbf{x})] = e^{-\alpha - K(\mathbf{x})}$$

其中 w_{ij} 是结点 i 和 j 之间的连接权系数,且 $w_{ij} = w_{ji}$, $w_{ii} = 0$, α 是归一化因子。

对于已知的样本,其输入和输出结点受到约束,设其状态为 \mathbf{x}_i ,相应的概率 $P_i(\mathbf{x}_i)$ 也是已知的,那么要求的能量应为

$$K_s(\mathbf{x}_i) = -\alpha - \ln P_i(\mathbf{x}_i)$$

取代价函数为

$$J = \sum_i [K(\mathbf{x}_i) - K_s(\mathbf{x}_i)]^2 + a \sum_i \sum_j w_{ij}^2$$

其中 $K(\mathbf{x}_i)$ 是 BM 网络的能量函数, s 是样本数。 J 中第二项主要用来限制 w_{ij} 不要太大。

可以证明, J 只有一个全局极小值,且取得极小值时必有 $K(\mathbf{x}_i) = K_s(\mathbf{x}_i)$,这正是我们所希望的。这里可以采用简单的一阶梯度寻优算法来寻求最优的连接权系数 w_{ij} 以使 J 极小。

$$w_{ij}(k+1) = w_{ij}(k) - \epsilon \frac{\partial J}{\partial w_{ij}}$$

可以求得

$$\frac{\partial J}{\partial w_{ij}} = 2 \sum_i [K(\mathbf{x}_i) - K_s(\mathbf{x}_i)] x_i^i x_j^i + 2a w_{ij}$$

在 BM 网学习好后,便可用来进行任务规划。具体是这样实现的:首先将要求的任务转换为一定的基元组合,将它们作为 BM 网络的输入约束向量,然后对 BM 网络进行搜索计算,以找出能量函数的最小点,设这时网络状态为 \mathbf{X}^* ,这时的输出结点状态即为要求的子任务输出。根据这时的能量函数 $K(\mathbf{X}^*)$,可进一步算出该输入输出对的概率 $P[K(\mathbf{X}^*)]$,显然这是最大概率。也就是说,搜索的结果求得了一组最大可能完成的子任务。

务组合。从熵的观点,这时的信息熵

$$H[K(\mathbf{x}^*)] = -P[K(\mathbf{x}^*)]\ln\{P[K(\mathbf{x}^*)]\} - \{1 - P[K(\mathbf{x}^*)]\}\ln\{1 - P[K(\mathbf{x}^*)]\}$$

是最小的,也就是不确定性能程度最小,所以组织级的作用可看成是寻求一组子任务的动作组合,以使得信息传输的熵最小,也就是使所规划的结果的不确定性最低。

在进行任务规划时,寻优计算目标函数仍是能量函数或熵函数。被寻优的参数是网络的状态 \mathbf{x} (其中一部分相应于输入结点的状态是受约束的),而连接权系数是不变的。注意这时所进行的规划寻优计算与学习时的寻优计算的不同,前者改变的是网络的结点状态,后者改变的是网络的连接权系数。对于规划寻优计算,能量函数 $K(\mathbf{x})$ 一般可能有多个极值点,因而采用一般的寻优算法常常陷入局部极值点而不能到达全局的极小点。下面介绍两种可以收敛到全局极值点的寻优算法。

1. 模拟退火算法

本书第3章在介绍基于神经网络的路径规划时曾经讨论过模拟退火算法。这里的模拟退火步骤与前而类似,即开始用较高的“温度” T ,然后逐渐减小 T 。这里冷却的规律采用下式:

$$\frac{T(t)}{T_0} = \frac{1}{\log(10+t)}$$

其中 T_0 为初始“温度”, $T(t)$ 为 t 时刻的“温度”。

具体寻优的方法是:当迭代计算到第 k 步,其状态为 $\mathbf{x}_k = (x_1, x_2, \dots, x_n)$,将其随机变化到 \mathbf{x}'_k , \mathbf{x}'_k 与 \mathbf{x}_k 的海明距离为 1。计算熵的变化 $\Delta H = H(\mathbf{x}'_k) - H(\mathbf{x}_k)$ 。如果 $\Delta H \leq 0$,取 $\mathbf{x}_{k+1} = \mathbf{x}'_k$;若 $\Delta H > 0$,则按如下的概率接受新状态:

$$P(\mathbf{x}_{k+1} = \mathbf{x}'_k) = e^{-\Delta H/K_B T}$$

其中 K_B 是 Boltzmann 常数, T 是“温度”,它由前面的式子确定。另外一种方法是将上式替换为下式:

$$P(\mathbf{x}_{k+1} = \mathbf{x}'_k) = \frac{1}{1 + e^{\Delta H/T}}$$

上而两个式子都提供了跳出局部极小值的可能,加之采用模拟退火的步骤。从而使该算法可以收敛到全局的极小值。

2. 扩展子区间随机搜索

这时采用如下的迭代步骤

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{x}'_k & H(\mathbf{x}'_k) - H(\mathbf{x}_k) \leq 2\mu \\ \mathbf{x}_k & H(\mathbf{x}'_k) - H(\mathbf{x}_k) > 2\mu \end{cases}$$

其中 μ 是预先设定的常数,可以证明,利用该算法有

$$\lim_{k \rightarrow \infty} \text{Prob}[H(\mathbf{x}_k) - H_{\min}) < \delta] = 1$$

其中 H_{\min} 是全局最小的熵函数。

以上介绍了利用了 Boltzmann 机进行任务规划的方法。在一次任务执行完成后,需要对任务执行的结果进行评价。并据此性能评价最终来修改 Boltzmann 机的连接权系数。这是一个机器学习的过程。

开始讲到, BM 网络首先需要根据测试样本进行学习。每个测试样本都赋与一个概

率,如果所执行的任务就是测试样本中的一个,那么现在需要根据实际运行的结果来修改先前的概率。如果所执行的任务不在原来的测试样本中,则可将本次 BM 网络的输入输出对加入到新的样本中,同时根据任务执行的结果来修改先前计算得到的概率,具体修改采用如下随机逼近学习算法:

$$P(k+1) = P(k) + \beta(k+1)[\xi - P(k)]$$

其中 ξ 是性能评价因子, $\xi \in \{0, 1\}$ 。若对本次执行结果的性能满意,则取 $\xi=1$, 否则取 $\xi=0$ 。 $\beta(k)$ 的选取需满足如下的 Dvoretzky 条件:

$$\begin{cases} \lim_{k \rightarrow \infty} \beta(k) = 0 \\ \lim_{k \rightarrow \infty} \sum_{i=1}^k \beta(i) = \infty \\ \lim_{k \rightarrow \infty} \sum_{i=1}^k \beta^2(i) < \infty \end{cases}$$

一般可取 $\beta(k) = 1/k$, 它能满足上述条件。

在一次任务执行完成后,按照上述学习算法修改相应的样本概率,并相应的修改长效记忆存储器中的内容。同时根据新的样本概率重新对 Boltzmann 机进行学习和训练,学习的结果是获得一组修改了的连接权系数 w_{ij} 。当下次接收到外部任务输入命令时, BM 网络将按照新的参数结构进行任务规划。

6.3 协调级

协调级是分层递阶智能控制结构的中间层,它接受从组织级传来的命令,经过实时信息处理,产生一系列可供执行级执行的具体动作的序列。

协调级的具体实现方法有很多种,这里介绍 F. Y. Wang 所提出的一种用 Petri 网翻译器实现的语言决策方法来对协调过程进行分析和建模。利用 Petri 网为工具,可以提供分层次的模块化设计方法,并可提供像活性、死锁、有界、平均执行时间、系统可用性等性能的分析。

6.3.1 协调级的原理结构

协调级可采用如图 6.4 所示的树形结构。其中 D 是根结点,称为分派器(Dispatcher), C 是子结点的有限集合,称为协调器(Coordinator),每个协调器与分派器之间均存在双向联系,而在协调器之间没有直接的联系。

组织级来的命令首先传送到协调级中的分派器,这些命令表示为基元事件的组合。分派器负责对各协调器的控制和通讯。它根据当前工作状态将组织级送来的基元事件序列翻译为面向协调器的控制行动,然后在合适的时候将它们送至相应的协调器。在任务执行完毕后,分派器还负责向组织级传送反馈信息。为了完成上述任务,分派器需具备以下功能:

- (1) 通讯功能。它能够向上层的组织级和下而的协调级发送和接收信息。
- (2) 数据处理功能。它能对组织级来的命令信息和从协调器来的反馈信息进行描述。

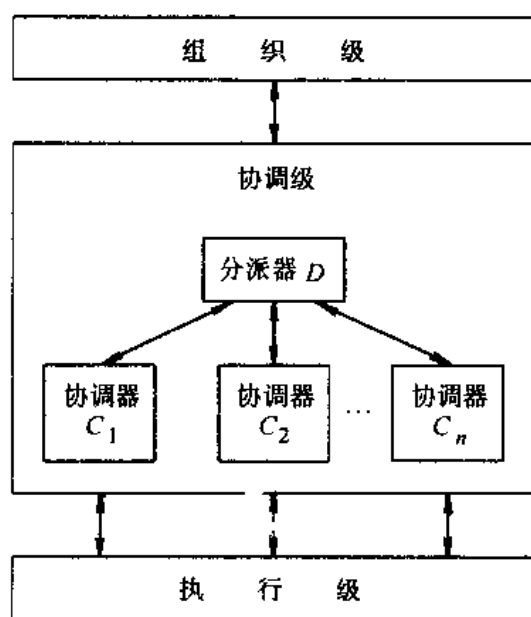


图 6.4 协调级的树形结构

并可为分派器的决策单元提供信息和对它进行修改。

(3) 任务处理功能。它能对要执行的任务进行识别,为相应的协调器选择合适的控制步骤,以及为组织级产生必要的反馈信息。

(4) 学习能力。它能够根据任务不断执行所取得的经验来逐渐减小决策过程的不确定性,以达到不断改进任务执行的能力。

每个协调器均与一定的装置相联系,并将对这些装置进行操作和数据传输。协调器可看成是在特定领域实现具体功能的一个专家。为了完成同一个任务,分派器所发出的指令中可能包含好几种不同的方法,协调器能够根据机器人的工作状态和时间要求从中选择出一种合适的方法。它将面向协调器的控制行动序列翻译为面向执行级的实时操作序列,并连同相关的数据一起送至具体的装置。在任务执行完成后,它还负责向分派器报告执行的结果。协调器与分派器具有完全相同的结构形式,只不过是有一个较低和较具体的水平上实现与分派器相同的功能。

图 6.5 说明任务分派器和协调器之间的具体翻译过程。这些不同层次的任务是用语言来描述的。由于分派器和协调器处在树形结构的不同层次上,因而它们进行这种语言翻译的时间尺度也是不相同的。分派器的一步可以变为协调器的许多步,所有协调器必须在分派器的统一管理下协同工作。

上面已经说到,分派器和协调器具有完全相同的结构形式,只不过它们处于不同的层次以实现各自的功能。图 6.6 表示了它们的统一的结构形式。它们分别由数据处理器,任务处理器以及学习处理器所组成。

数据处理器的功能是提供需要执行的任务的信息以及当前系统的状态。它分为以下三个层次的描述:任务描述、状态描述以及数据描述。在任务描述中,它给出来自上层的需

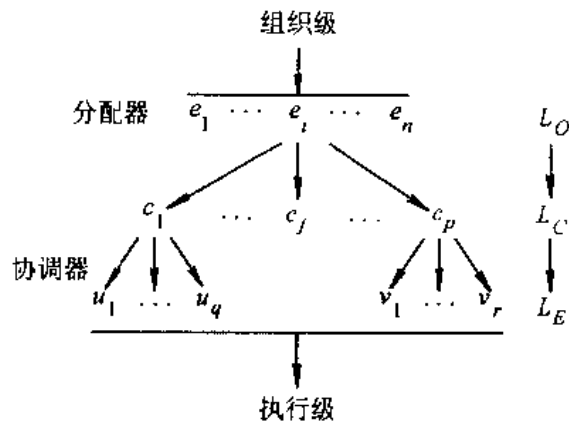


图 6.5 协调级的语言翻译

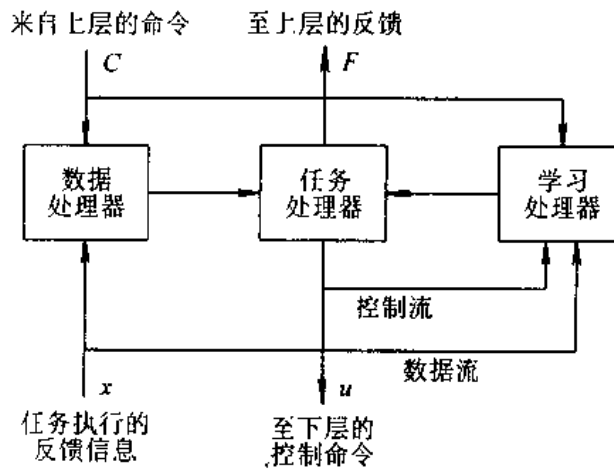


图 6.6 分派器和协调器的统一结构

要执行的任务清单;状态描述按一种比较抽象的形式给出每次任务执行的前件和后件以及系统的状态;数据描述则给出状态描述的具体数值。这样的信息结构形式对于任务处理器中的分层决策是非常有用的。数据处理器中还包括一个监控器,它根据上层来的指令信息和下层的反馈信息对上述三个层次的描述进行维护和修改。该监控器还负责数据处理器与任务处理器之间的连结。

任务处理器的功能是为下层单元提供控制命令的准确描述。它采用分层决策的步骤,具体说来分以下三步:任务调度(scheduling)、任务翻译(translation)和任务的准确描述(formulation)。任务调度是通过检查任务描述及其前件和后件来识别要执行的任务,在这一步并不需要用到状态的实际数值。如果没有满足条件的子任务可以执行,任务调度必须先进行一些内部操作以使得某些任务的前件得以满足。根据当前状态,任务翻译以合适的方式将任务或者内部操作分解为控制作用,最后通过搜索数据库中的数据描述给控制作用赋以实际的数值,从而实现任务的准确描述,并将该完整的控制命令送至低层单元。这样的分层决策方法可以使得任务的处理层次清晰、处理快速。在所有的任务完成后,通过

监控器来组织反馈信息,并以某种特定方式送至上层。该监控器也负责任务处理器与学习处理器之间的连结。

学习处理器的功能是用来改善任务处理器的性能及减小决策和信息处理的不确定性。为了实现这个功能,可以采用各种各样的学习机制。比较常用的是采用线性随机学习算法。

6.3.2 Petri 网翻译器

根据上面的讨论,协调级的基本功能可以看成是将组织级所发出的高级命令语言翻译成低层装置可以执行的操作语言。利用 Petri 网翻译器(transducer)可以实现上述功能。

Petri 网是一个如下的 4 元组

$$N = (P, T, I, O)$$

其中 P 是位置(place)的集合, T 是迁移(transition)的集合, I 是输入函数, O 是输出函数。图 6.7 示出了一个典型的 Petri 网。

其中

$$P = \{p_1, p_2, p_3\}$$

$$T = \{t_1, t_2\}$$

$$I = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$O = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

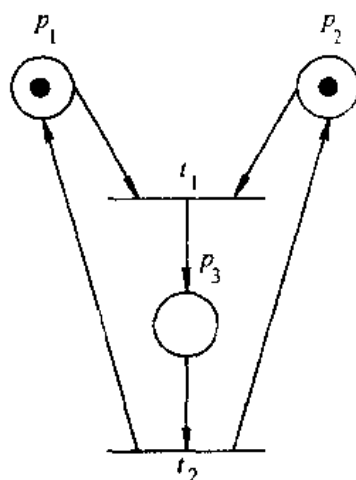


图 6.7 Petri 网举例

位置的集合 P 代表了系统的状态,迁移的集合 T 代表改变系统状态的事件集合。输入函数代表了事件发生的前件,输出函数代表事件发生的结果。一个位置中可以包含非负整数个记号(token),如图中圆圈中的黑点所示,由 Petri 网所建模的系统总状态用它的标记(marking)来表示。标记即为每个位置中的记数,如图 6.7,标记 $m = (1 \ 1 \ 0)^T$,关于 Petri 网理论的详细描述,请参考有关文献。

下面给出几个常用的符号和术语,它们是后面需要用到的。

$\delta(\mu, t)$ 表示在标记为 μ 时激发迁移 t 后产生的新的状态。 $R(N, \mu)$ 或简写为 $R(\mu)$ 表示初始状态为 μ 时的能够达到的标记集。如果 $I(t_1) + I(t_2) \leq \mu$, 则称 t_1 和 t_2 是并发(parallel)的,即 t_1 和 t_2 可以同时激发(fire)。若 $I(t_1) \leq \mu$ 和 $I(t_2) \leq \mu$, 但 $I(t_1) + I(t_2) > \mu$, 则称 t_1 和 t_2 是冲突的(conflict),即激发其中任何一个将使另一个不能激发。如果对于在 $R(\mu)$ 中的任何一个标记, Petri 网中的任何一个迁移总有可能被激发,则称该 Petri 网是活的(Live), Petri 网的活性确保了不会存在死锁现象。如果对于 $R(\mu)$ 中的任何标记,每一个位置中的标记数均不会超过有限数 K , 则称该 Petri 网是有界(bounded)的。当 $K=1$ 时,称该网是安全的(safe)。

Petri 网翻译器(下面有时简称为 PNT)是以 Petri 网为工具的语言翻译器,它定义为如下的 6 元组

$$M = (N, \Sigma, \Delta, \sigma, \mu, F)$$

其中 $N=(P, T, I, O)$ 是具有初始标记为 μ 的 Petri 网, Σ 是有限输入字符集, Δ 是有限输出字符集, σ 是从 $T \times (\Sigma \cup \{\lambda\})$ 到 Δ^* 的有限集的翻译映射, $F \subset R(\mu)$ 是终了标记集合。其中 Δ^* 表示由 Δ 中字符组成的字符串集, λ 表示空字符串。

PNT 由三部分组成:输入带、Petri 网控制器和输出带,如图 6.8 所示。Petri 网翻译器 M 的结构可定义为三元组 (m, x, y) 。其中 $m \in R(\mu)$ 是 N 的当前标记, $x \in \Sigma^*$ 是输入带中剩余部分的字符串, $y \in \Delta^*$ 是已经翻译得到的字符串。用 \Rightarrow 表示 m 的移动(move)关系。例如,若 $m \in R(\mu), t \in T, a \in \Sigma \cup \{\lambda\}, x \in \Sigma^*, y \in \Delta^*, \delta(m, t)$ 有定义, $z \in \Delta^*$ 包含在 $\sigma(t, a)$ 中,则可写为

$$(m, ax, y) \Rightarrow (\delta(m, t), x, yz)$$

多重移动可以用符号 \Rightarrow^* 来表示。PNT 的语言关系并不是一一对应的。即同一个输入语言可以映射为多个输出语言,或者多个输入语言可以映射为同一个输出语言。定义

$\tau(M) = \{(x, y) \mid (\mu, x, \lambda) \Rightarrow^* (m, \lambda, y), m \in F\}$ 为 M 的语言翻译集合, $\alpha(M) = \{x \mid (x, y) \in \tau(M), y \in \Delta^*\}$ 为输入语言的集合, $\omega(M) = \{y \mid (x, y) \in \tau(M), x \in \Sigma^*\}$ 为输出语言的集合。

若两个 Petri 网翻译器 $M_i = (N_i, \Sigma_i, \Delta_i, \sigma_i, \mu_i, F_i) (i=1, 2)$ 组合在一起,可构成一个同步组合结构(synchronous composition)的 PNT,并记为 $M = M_1 \parallel M_2$,其移动关系定义为

$$((m_1, m_2), ax, y) \Rightarrow \begin{cases} ((\delta(m_1, t_1), m_2), x, yz_1) & a \in \Sigma_1 - \Sigma_2 \\ ((m_1, \delta(m_2, t_2)), x, yz_2) & a \in \Sigma_2 - \Sigma_1 \\ ((\delta(m_1, t_1), \delta(m_2, t_2)), x, yz_1 z_2 \text{ or } yz_2 z_1) & a \in \Sigma_1 \cap \Sigma_2 \end{cases}$$

其中 $z_1 \in \sigma_1(t_1, a)$ 和 $z_2 \in \sigma_2(t_2, a)$ 。上式表明,只属于其中一个网的输入符号仍只由那个网单独翻译,对于同时属于两个网的输入符号则由两个网按任意的次序同时翻译。对于输入字符串 x ,如果

$$((\mu_1, \mu_2), x, \lambda) \Rightarrow^* ((m_1, m_2), \lambda, y), m_1 \in F_1, m_2 \in F_2$$

则称 y 为经 $M = M_1 \parallel M_2$ 翻译得到的输出字符串。

6.3.3 协调级的 Petri 网结构

用 Petri 网实现的协调级结构 CS 可以表示为如下的七元组:

$$CS = (D, C, F, R_D, S_D, R_C, S_C)$$

其中 $D = (N_d, \Sigma_d, \Delta_d, \sigma_d, \mu_d, F_d)$ 是分派器的 Petri 网翻译器, $N_d = (P_d, T_d, I_d, O_d)$ 。 $C = (C_1, C_2, \dots, C_n)$ 是协调器的集合, $n \geq 1$ 。每个协调器也是一个 Petri 网翻译器, $C_i = (N_i^c, \Sigma_i^c, \Delta_i^c, \sigma_i^c, \mu_i^c, F_i^c)$, $N_i^c = (P_i^c, T_i^c, I_i^c, O_i^c)$ 。 $T_c = \bigcup_{i=1}^n T_i^c, P_c = \bigcup_{i=1}^n P_i^c, F = \bigcup_{i=1}^n \{f_i^c, f_{si}^c, f_o^c, f_{so}^c\}$ 是连结

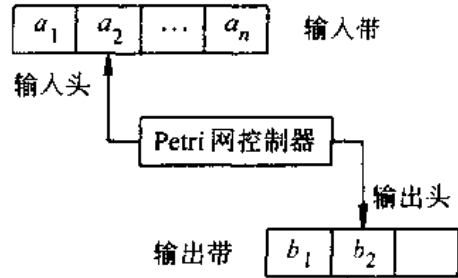


图 6.8 Petri 网翻译器(PNT)

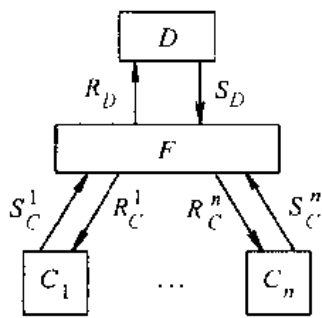


图 6.9 协调级结构框图

点的集合： f_I, f_{SI}, f_O 和 f_{SO} 分别称为 C_i 的输入点、输入号志(semaphore)、输出点和输出号志。 R_D 和 S_D 分别是分派器从 T_d 到 F 的接收映射和发送映射。 R_C 和 S_C 分别是协调器从 T_c 到 F 的接收映射和发送映射。图 6.9 表示了协调级的结构框图，图 6.10 表示了一个简单的例子。

由图看出，每个协调器 C_i 均与分派器 D 有双向联系。只有在相应的输入号志中有记号时，分派器才能向协调器发送任务命令；同样也只有在相应的输出号志中有记号时，协调器才能向分派器报告执行结果。

通过设计不同的接收和发送映射可以在分派器和协调器之间产生各种复杂的连接模式。其中一种最简单的连接模式为：(1) C_i 仅能与它自己的连接点发生联系；(2) 在 C_i 中开始只有一个迁移处于能激发状态，并可从输入点处接收输入；(3) C_i 中只有一个迁移能够向它的输出点发送信息。具有上述连接模式的结构称为简单协调级结构。图 6.10 即为这种简单结构的例子。

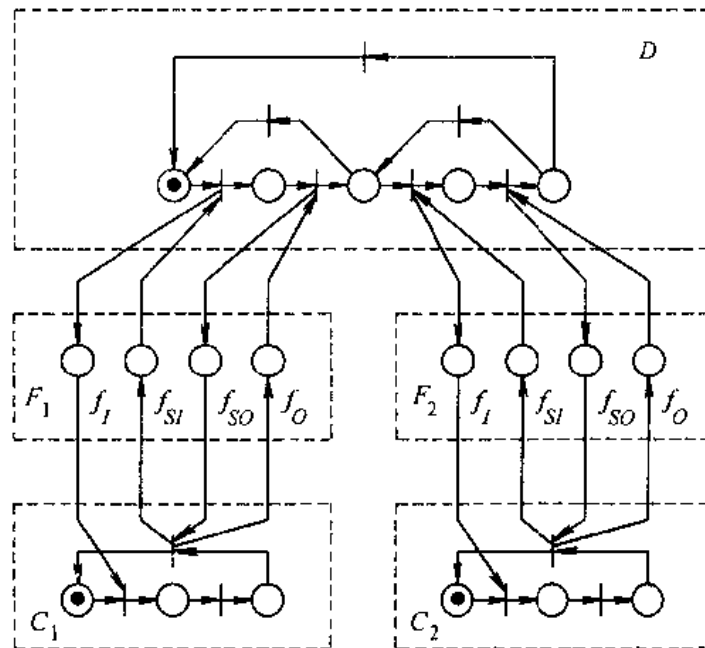


图 6.10 协调级的结构举例

整个协调级结构的工作可以用如下的总的 Petri 网来描述：

$$N = (P, T, I, O)$$

其中

$$P = P_d \cup P_c \cup F \quad T = T_d \cup T_c$$

$$I(t) = \begin{cases} I_d(t) \cup \{f | (t, f) \in R_D\} & t \in T_d \\ I_c(t) \cup \{f | (t, f) \in R_C\} & t \in T_c \end{cases}$$

$$O(t) = \begin{cases} O_d(t) \cup \{f_i(t, f) \in S_d\} & t \in T_d \\ O_c(t) \cup \{f_i(t, f) \in S_c\} & t \in T_c \end{cases}$$

N 的起始标记为

$$\mu(P) = \begin{cases} \mu_d(P) \text{ or } \mu_c(P), P \in P_d \text{ or } P \in P_c \\ 1 & P = f_{SI} \text{ or } f_{SO} \\ 0 & \text{others} \end{cases}$$

开始工作时, CS 从组织级接收一个字符串(即规划的任务), 将该字符串放到分派器 D 的输入带上, 并开始进行翻译, 也即进行任务分派。一旦 D 中的一个迁移 t 激发, Petri 网便执行一条基元事件 a , 设 f_1^i, \dots, f_n^i 是 t 的输出位置, 也就意味着所选择的控制串 $z \in \sigma_d(t, a)$ 送到了协调器 C_{i1}, \dots, C_{in} , 并激发同步组合 $C_{i1} \parallel \dots \parallel C_{in}$, 当协调器 C_j 完成任务后, 它将移送一个记号到 f_b , 即相当于送回一个反馈信号。如果满足激发条件, 分派器取走该反馈信息, 并继续下面的进程, 这时 C_j 将变为空闲状态。一旦分派器到达它的终了标记状态, 各个协调器也到达各自的终了标记状态或有的仍处于初始标记状态(说明该协调器未参与该次任务的工作), 即说明已成功地完成了任务。

显然, 同步结构网络为分派器同步和协调各协调器的工作提供了工具。总的 Petri 网 N 为整个协调级(包括分派器和协调器)工作确定了相互之间的约束关系。组织级所发出的字符串为分派器网 N_d 确定运行路径, 分派器 D 翻译得到的字符串为各协调器网 N_c 确定运行路径。

利用 Petri 网作为工具, 可以帮助研究协调级的许多性质, 如活性、有界性、可逆性、一致性、重复性等。

6.3.4 协调级结构的决策和学习

协调级的决策过程是通过任务调度和任务翻译这两个步骤来实现的。对于要求的作业, 任务调度的作用在于识别出可以执行的合适的任务。当确定好一个任务后, 通过翻译将该任务分解为子任务序列。当赋予这些子任务以实时信息后, 即可将它们送至相应的单元加以执行。若采用 Petri 网翻译器的术语, 任务调度和任务翻译的问题可描述为: 对于给定的任务 a 找到一个可以激发的迁移 t 以使 $\sigma(t, a)$ 有定义, 进而根据 $\sigma(t, a)$ 找到正确的翻译串。

基于 Petri 网的执行规则可以设计出一种简单而统一的调度步骤。例如, 如果设 $M = (N, \Sigma, \Delta, \sigma, \mu, F)$ 表示分派器或一个协调器, 对于任何 $a \in \Sigma$, 定义 $T(a) = \{t | \sigma(t, a) \text{ 有定义}\}$; 定义 $T_\lambda = T(\lambda) = \{t | \sigma(t, \lambda) \text{ 有定义}\}$, 它相当于内部操作的集合。设置 Q_T 和 Q_D 两个队列, Q_T 存储未被执行的任务, Q_D 存储那些暂时条件不具备而需要推迟执行的任务。定义函数 $F(Q)$ 为取出 Q 的第一个元素, $I(Q, a)$ 为插入元素 a 到 Q 的末尾, $U(Q_1, Q_2)$ 为置 Q_2 于 Q_1 的末尾, $N(Q)$ 为使 Q 为空, 令 $v = a_1, a_2, \dots, a_s \in \Delta^*$ 为要执行的任务串, 则 M 的调度步骤如下:

- (1) $Q_T := \{a_1, a_2, \dots, a_s\}, Q_D = \emptyset$ (空集);
- (2) 如果 Q_T 为空, 则退出;
- (3) $u := F(Q_T)$;

(4) 如果存在一个 $t \in T(u)$ 且 t 处于使能激发状态, 那么激发 t , 并转(7);

(5) 如果存在一个内部操作序列 $e \in T_i^*$, 以使得激发 e 后 $t \in T(u)$ 处于使能激发状态, 则激发 et , 转(7);

(6) $I(Q_D, u)$, 如果 Q_T 为空, 则 $Q_T := Q_D$ 且 $N(Q_D)$, 转(2);

(7) 若 Q_D 不为空, 则 $Q_T := u(Q_D, Q_T)$ 并 $N(Q_D)$, 转(2)。

按照上述步骤, 首先对字符串 v 中所代表的任务依次进行检验, 若任务能执行, 则立即执行; 若不能, 则设法找到一个内部操作序列以使得依次轮到的任务能够执行。若这一点也不能实现, 则将此任务移送到等待队列 Q_D 中, 一旦 M 的状态发生了变化, 再将 Q_D 按原次序送返 Q_T 中, 并依次进行检验。这样可做到尽可能按 v 中原来的次序执行任务。对于比较复杂的 Petri 网翻译器(PNT), 寻找内部操作序列是一件很困难的工作, 这时可考虑使用启发式的搜索算法。对于一般的 PNT, 则沿当前标记为起点的可达树, 采用一般的宽度优先的搜索算法。由于假定所发出的任务串是相容且完全的, 因此上述的调度步骤一定能在有限步完成调度任务。

任务翻译可采用主动和被动两种方式。主动方式是基于知识库来进行任务的翻译。该知识库由一组规则和描述相关环境及系统状态信息的数据库所组成; 被动方式则是预先指定了一组翻译, 然后按照当前的局势(situation)来选择其中的一个。下面考虑被动方式, 并采用学习机制来帮助进行这种选择。

设 t 为 PNT $M = (N, \Sigma, \Delta, \sigma, \mu, F)$ 的一个迁移, 其翻译总数设为 $M_t = \sum |\sigma(t, a)|$, x_t 表示系统状态信息及关于 t 的输入位置的限制条件, $u_t \in U_t \triangleq \{a \in \Sigma \cup \{\lambda\} | \sigma(t, a) \text{ 有定义}\}$ 表示要由 t 来翻译的一个任务。 x_t 和 u_t 的组合 (x_t, u_t) 定义为一种局势。设关于 t 的局势总数为 N_t , 则关于 t 的可能的翻译总共为 $M_t \times N_t$ 个。

当给定一种局势 $(x_t, u_t)_j$ 时, 则总共可能有 M_t 种不同的翻译, 设选择翻译输出为 S_i 的概率为 P_{ij} , $i = 1, 2, \dots, M_t$, $j = 1, 2, \dots, N_t$ 。显然 P_{ij} 应满足

$$\sum_{i=1}^{M_t} P_{ij} = 1$$

因此在进行任务翻译时, 采用上述的概率模型进行随机决策, 即当出现局势 $(x_t, u_t)_j$ 时, 按概率 P_{ij} 选择翻译输出 S_i 。

这里 P_{ij} 为主观概率, 它将根据任务的执行情况不断加以修正。修改 P_{ij} 的原则是: 若按这样的选择执行任务后性能满意, 则将此概率 P_{ij} 增加, 否则 P_{ij} 减小。为此, 需要首先对性能进行估计。下面给出一种估计性能的方法。

$$\hat{J}_{ij}(k_{ij} + 1) = \hat{J}_{ij} + \beta(k_{ij} + 1)[J_o(k_{ij} + 1) - \hat{J}_{ij}(k_{ij})]$$

其中 J_o 是观测到的性能值, \hat{J} 是性能估计, k_{ij} 是事件 $[(x_t, u_t)_j, S_i]$ 发生的次数。在有了对性能的估计后, 进一步采用如下的算法对概率 P_{ij} 进行修正

$$P_{ij}(k + 1) = P_{ij}(k) - \gamma(k + 1)[\xi_{ij}(k) - P_{ij}(k)]$$

其中

$$\xi_{ij}(k) = \begin{cases} 1 & \text{when } \hat{J}_{ij} = \min_i \hat{J}_{ij} \\ 0 & \text{others} \end{cases}$$

当 $\beta(k_{ij})$ 和 $\gamma(k)$ 满足 Dvoretzky 条件时, 上述修正算法一定收敛, 且

$$\begin{aligned} \text{Prob}\{\lim_{k \rightarrow \infty} [\hat{J}_{ij}(k) - \bar{J}_{ij}] = 0\} &= 1 \\ \begin{cases} \text{Prob}\{\lim_{k \rightarrow \infty} P_{ij}(k) = 1\} = 1 & \text{when } \bar{J}_{ij} = \min_i \bar{J}_{ij} \\ \text{Prob}\{\lim_{k \rightarrow \infty} P_{ij}(k) = 0\} = 1 & \text{others} \end{cases} \end{aligned}$$

其中 \bar{J}_{ij} 表示 J_{ij} 的期望值。

上述的学习过程可以用熵来测量。因为熵是不确定性程度的度量。所以对于一个 PNT M , 其翻译的不确定性可以用如下的熵函数来表示

$$\begin{aligned} H(M) &= \sum_{i \in T} H(t) = \sum_{i \in T} \{H[(u_i, x_i)] + H[t/(u_i, x_i)]\} \\ &= - \sum_{i \in T} \sum_j P_j \ln P_j - \sum_{i \in T} \sum_j P_j \sum_i P_{ij} \ln P_{ij} \end{aligned}$$

其中 P_j 表示局势 $(u_i, x_i)_j$ 发生的概率。定义

$$\begin{aligned} H(E) &= - \sum_{i \in T} \sum_j P_j \ln P_j \\ H(T/E) &= - \sum_{i \in T} \sum_j P_j \sum_i P_{ij} \ln P_{ij} \end{aligned}$$

则

$$H(M) = H(E) + H(T/E)$$

上式表明, 翻译的不确定性 $H(M)$ 可表示为两部分: 一部分是环境的不确定性 $H(E)$, 另一部分是在给定环境下纯粹的翻译不确定性 $H(T/E)$ 。显然, 只有第二项, 即纯粹的翻译的不确定性, 可以通过学习来减小。

对于整个协调级结构 CS 的不确定性可表示为如下的熵函数

$$H(CS) = H(E_{CS}) + H(T_{CS}/E_{CS})$$

其中

$$\begin{aligned} H(E_{CS}) &= H(E_D) + \sum_{i=1}^n H(E_{C_i}) \\ H(T_{CS}/E_{CS}) &= H(T_D/E_D) + \sum_{i=1}^n H(T_{C_i}/E_{C_i}) \end{aligned}$$

对于协调级, 设计的目标就是要使得 $H(CS)$ 最小, 即分派和协调任务的不确定性程度降至最小。

图 6.11 表示了一个简单的智能机械手系统协调级结构的例子, 该系统由一个六自由度的机械手, 一个视觉系统及一个传感器系统组成。视觉系统用来识别物体并对这些物体定位, 传感器系统提供各种传感信息。该协调级由一个分派器、一个运动协调器、一个视觉协调器和一个传感器协调器组成。图中将分派器和各协调器的连接处集中画在了一起, 仅是为了简化画图。

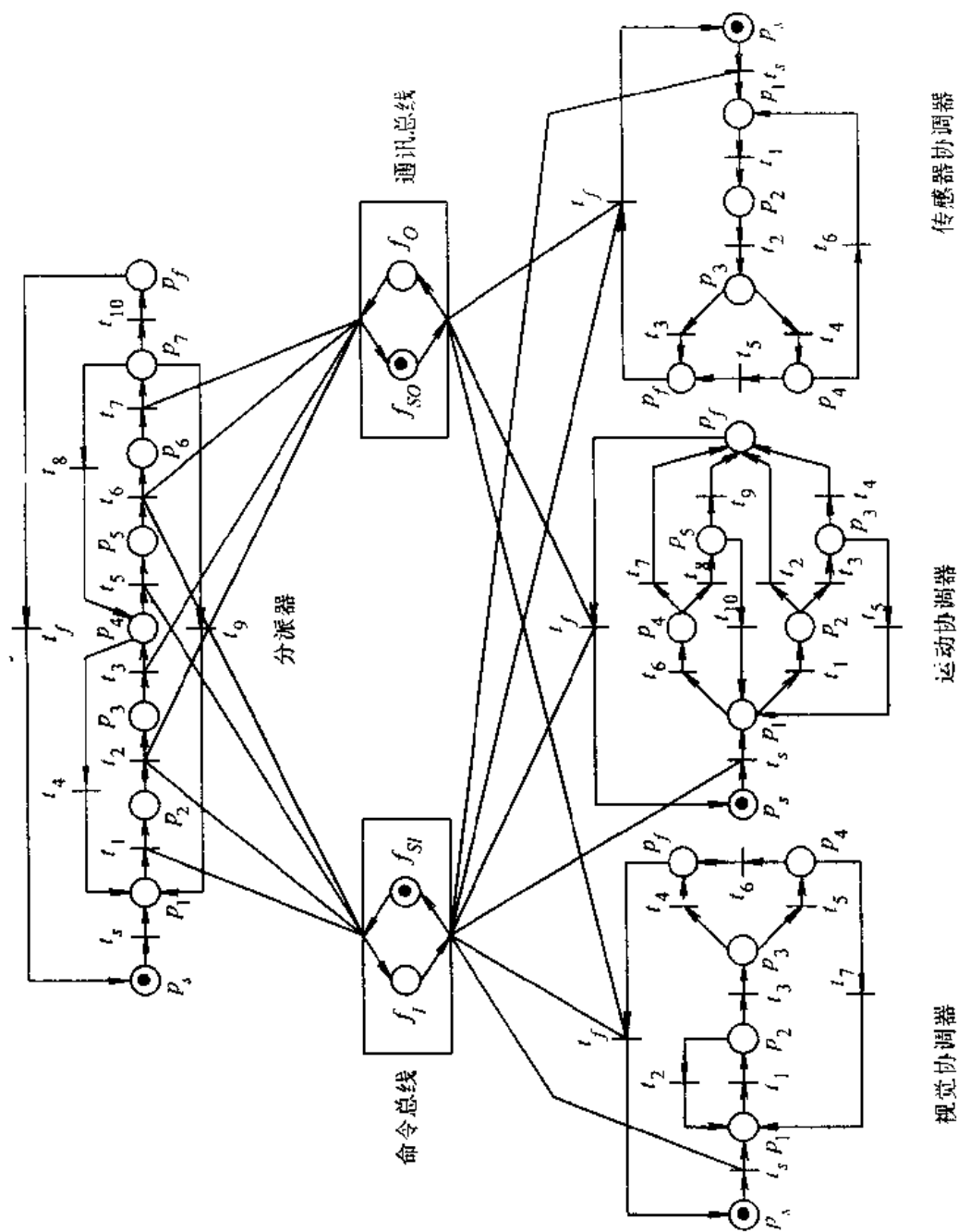


图 9.1.1 智能机械手系统的协调线路结构

6.4 执 行 级

执行级即为常规硬件控制级,对于系统的上层两级,即组织级和协调级。它们主要是基于知识的概念,采用信息理论,通过建立概率模型用信息熵作为系统性能的度量。对于控制级,通常采用系统理论中的指标函数来衡量系统的性能。这样对于智能控制系统便很难有一个统一的性能测试。下面介绍 Saridis 提出的一种方法,它将执行级的控制性能也表示为熵函数的形式,该熵函数可以用来测量执行任务时选择控制作用的不确定性。这样便将控制问题与高层的信息论的分析方法统一起来,从而可将熵作为整个智能控制系统统一的性能测试。

若控制级采用最优控制理论的设计方法。设系统的状态方程和输出方程分别为

$$\begin{aligned}\frac{dx}{dt} &= f(x, u(x, t), w, t) & x(t_0) &= x_0 \\ y &= g(x, v, t) & x(t_f) &\in M_f\end{aligned}$$

其中 x 是状态量, u 是控制量, y 是输出量, w 是随机干扰, v 是随机测量噪声, x_0 是初态, 它也是随机变量, $x(t_f)$ 是末态, M_f 是状态空间 Ω_x 中的一个子集。定义如下的广义能量函数作为系统性能的测量

$$V(x_0, t_0) = E \left\{ \int_{t_0}^{t_f} L(x, t, u(x, t)) dt \right\}$$

其中 $L[x, t, u(x, t)] \geq 0$ 。控制的目标是在容许控制空间 Ω_u 中任意地选择控制量 $u(x, t)$, 以使得 V 极小。

为了用熵函数来表示该控制问题,假定控制量 $u(x, t)$ 在 Ω_u 中按概率密度 $p[u(x, t)]$ 进行分布,且有

$$\int_{\Omega_u} P[u(x, t)] dx = 1$$

相应于该概率分布的熵为

$$H(u) = - \int_{\Omega_u} P[u(x, t)] \ln P[u(x, t)] dx$$

它代表了在所有的容许反馈控制 Ω_u 中选择控制量 $u(x, t)$ 的不确定性。当系统具有最优的性能(即 V 取极小)时相应的控制量 $u(x, t)$ 的概率 $P[u(x, t)]$ 应当取最大。也就是说,最优控制 $u^*(x, t)$ 应当使熵函数 $H(u)$ 取极小。这一点可以通过选取如下的概率密度函数来实现。

$$P[u(x, t)] = e^{-\lambda - \mu V[x_0, t_0, u(x, t)]}$$

上式的选取满足极大熵原理。其中 λ 和 μ 是正则化常数因子,它满足

$$\lambda = \ln \int_{\Omega_u} e^{-\mu V[x_0, t_0, u(x, t)]} dx$$

结合上面各式可得

$$H(u) = \int_{\Omega_u} P[u(x, t)] \{ \lambda + \mu V[x_0, t_0, u(x, t)] \} dx$$

$$\begin{aligned}
&= \lambda + \mu \int_{a_1} P[u(x, t)] V[x_0, t_0, u(x, t)] dx \\
&= \lambda + \mu E\{V[x_0, t_0, u(x, t)]\}
\end{aligned}$$

该式表明,寻求最优控制 $u^*(x, t)$ 使 $V[x_0, t_0, u(x, t)]$ 极小就相当于使熵函数取极小,也相当于使 $P[u(x, t)]$ 取极大。这样就在信息理论与最优控制问题之间建立了等价的测度关系,从而为分层递阶智能控制系统采用熵函数作为统一的性能测度提供了理论基础。

参 考 文 献

- [1] Saridis G N. On the Revise Theory of Intelligent Machines. CIRSSE Report #58. RPI, USA, 1990
- [2] Moed M C, Saridis G N. A Boltzmann Machine for the Organization of Intelligent Machines. IEEE Trans. on SMC, 1990, 20(5)
- [3] Moed M C. A Connectionist/Symbolic Model for Planning Robotic Tasks. CIRSSE Report #78. RPI, USA, 1990
- [4] Wang F Y. A Coordinatory Theory for Intelligent Machines. CIRSSE Report #59. RPI, USA, 1990
- [5] Saridis G N. Entropy Formulation of Optimal and Adaptive Control. IEEE Trans. on AC, 1988, 33(8)

第7章 遗传算法

7.1 概 述

遗传算法(Genetic Algorithm-GA)是一种新发展起来的优化算法,它产生于美国的密执根大学。从60年代起,密执根大学的Hollstien, Bagley 和 Rosenberg 等人的博士论文即与遗传算法密切相关,而 John H. Holland 教授1975年出版的“Adaptation in Natural and Artificial Systems”一书通常认为是遗传算法的经典之作,因为该书给出了遗传算法的基本定理,并给出了大量的数学理论证明。David E. Goldberg 教授1989年出版的“Genetic Algorithms”一书通常认为是对遗传算法的方法、理论及应用的全面系统的总结。从1985年起,国际上开始举行遗传算法的国际会议,以后则更名为进化计算的国际会议,参加的人数及收录文章的数量、广度和深度逐次扩大。遗传算法已经成为人们用来解决高度复杂问题的一个新思路和新方法。目前遗传算法已被广泛应用于许多实际问题,如函数优化、自动控制、图象识别、机器学习、人工神经网络、分子生物学、优化调度等许多领域中的问题。

遗传算法是基于自然选择和基因遗传学原理的搜索算法。它将“适者生存”这一基本的达尔文进化理论引入串结构,并且在串之间进行有组织但又随机的信息交换。伴随着算法的运行,优良的品质被逐渐保留并加以组合,从而不断产生出更佳的个体。这一过程就如生物进化那样,好的特征被不断地继承下来,坏的特性被逐渐淘汰。新一代个体中包含着上一代个体的大量信息,新一代的个体不断地在总体特性上胜过旧的一代,从而使整个群体向前进化发展。对于遗传算法,也就是不断地接近于最优解。研究遗传算法的目的主要有两个:一是通过它的研究来进一步解释自然界的适应过程;二是为了将自然生物系统的重要机理运用到人工系统的设计中。

遗传算法的中心问题是鲁棒性(robustness),所谓鲁棒性是指能在许多不同的环境中通过效率及功能之间的协调平衡以求生存的能力。人工系统很难达到如生物系统那样的鲁棒性。遗传算法正是吸取了自然生物系统“适者生存”的进化原理,从而使它能够提供一个在复杂空间中进行鲁棒搜索的方法。遗传算法具有计算简单及功能强的特点,它对于搜索空间基本上不需要什么限制性的假设(如连续、导数存在及单峰等)。

我们首先来考察常规的寻优方法。按一般的文献介绍,寻优的方法主要有三种类型:解析法、枚举法和随机法。下面分别来讨论它们的鲁棒性能。

解析法寻优是研究得最多的一种,它一般又可分为间接法和直接法。间接法是通过让目标函数的梯度为零,进而求解一组非线性方程来寻求局部极值。直接法是按照梯度信息按最陡的方向逐次运动来寻求局部极值,它即为通常所称的爬山法。上述两种方法的主要缺点是:一是它们只能寻找局部极值而非全局的极值;二是它们要求目标函数是连续光滑的,并且需要导数信息。这两个缺点,使得解析寻优方法的鲁棒性能较差。

枚举法可以克服上述解析法的两个缺点,即它可以寻找到全局的极值,而且也不需要目标函数是连续光滑的。它的最大缺点是计算效率太低,对于一个实际问题,常常由于太大的搜索空间而不可能将所有情况都搜索到。即使很著名的动态规划方法(它本质上属于枚举法)也遇到“指数爆炸”的问题,它对于中等规模和适度复杂性的问题,也常常无能为力。

鉴于上述两种寻优方法有严重缺陷,随机搜索算法受到人们的青睐。随机搜索通过在搜索空间中随机地漫游并随时记录下所取得的最好结果。出于效率的考虑,搜索到一定程度便终止。然而所得结果一般尚不是最优值。本质上随机搜索仍然是一种枚举法。

遗传算法虽然也用到了随机技术,但它不同于上述的随机搜索。它通过对参数空间编码并用随机选择作为工具来引导搜索过程向着更高效的方向发展。目前流行的另外一种称为“模拟退火”的算法也具有类似的特点,它也借助于随机技术来帮助引导搜索过程至能量的极小状态。因此,随机地搜索并不一定意味着是一种无序的搜索。

总的说来,遗传算法比其它寻优算法的优点是鲁棒性能比较好。其主要的本质差别可以归纳为以下几点。

- (1) 遗传算法是对参数的编码进行操作,而不是对参数本身;
- (2) 遗传算法是从许多初始点开始并行操作,而不是从一个点开始。因而可以有效地防止搜索过程收敛于局部最优解,而且有可能求得全部最优解;
- (3) 遗传算法通过目标函数来计算适配值,而不需要其它的推导和附属信息,从而对问题的依赖性较小;
- (4) 遗传算法使用概率的转变规则,而不是确定性的规则;
- (5) 遗传算法在解空间内不是盲目地穷举或完全随机测试,而是一种启发式搜索,其搜索效率往往优于其它方法;
- (6) 遗传算法对于待寻优的函数基本无限制,它既不要求函数连续,更不要求可微;既可是数学解析式所表达的显函数,又可是映射矩阵甚至是神经网络等隐函数,因而应用范围较广;
- (7) 遗传算法具有并行计算的特点,因而可通过大规模并行计算来提高计算速度;
- (8) 遗传算法更适合大规模复杂问题的优化。

在详细讨论遗传算法的机理和功能之前,必须首先明确,当我们说优化一个函数或一个过程时,我们的目标到底是什么。传统的优化定义包括两个方面的含义:一是寻求性能的改进,二是最终寻求到优化点(或极值点)。这两个方面实际上一个是过程,另一个是目的。然而,我们通常只注意一个算法是否能收敛,即能否达到极值,而忽视了中间过程。这种观点实际上是受到微积分中优化概念的影响。然而对于实际问题,往往事先并不知道其优化点,在此情况下,用是否收敛至极值点来判断一个优化算法显然是不可行的。这时只能通过与其它方法的比较来判断某一优化算法的性能。因此,对于更广泛的优化问题来说,定义优化问题应更强调性能改进,也就是能否更快地达到令人满意的性能,而获得最优值比寻求性能改进要次要一些。这一点对于复杂系统更为明显。

7.2 遗传算法的工作原理及操作步骤

本节通过一个简单的例子,详细描述遗传算法的基本操作过程,并给出原理分析。目的在于清晰地展现遗传算法的特点。

7.2.1 遗传算法的基本操作

设需要求解的优化问题为寻找 $f(x)=x^2$ 当自变量 x 在 $0\sim 31$ 之间取整数值时函数的最大值。枚举的方法是将 x 取尽所有可能值,观察是否得到最高的目标函数值。尽管对如此简单的问题该法是可靠的,但这是一种效率很低的方法。下面我们运用遗传算法来求解这个问题。

遗传算法的第一步是将 x 编码为有限长度的串。编码的方法很多,这里仅举一种简单易行的方法。针对本例中自变量的定义域,可以考虑采用二进制数来对其编码,这里恰好可用 5 位数来表示,例如 01010 对应 $x=10$,11111 对应 $x=31$,许多其它的优化方法是从定义域空间的某个单个点出发来求解问题,并且根据某些规则,它相当于按照一定的路线,进行点到点的顺序搜索,这对于多峰值问题的求解很容易陷入局部极值。而遗传算法则是从一个种群(由若干个串组成,每个串对应一个自变量值)开始,不断地产生和测试新一代的种群。这种方法一开始便扩大了搜索的范围,因而可期望较快地完成问题的求解。初始种群的生成往往是随机产生的。对于本例,若设种群大小为 4,即含有 4 个个体,则需按位随机生成 4 个 5 位二进制串。例如我们可以通过掷硬币的方法来生成随机的串。若用计算机,可考虑首先产生 $0\sim 1$ 之间均匀分布的随机数,然后规定产生的随机数在 $0\sim 0.5$ 之间代表 0, $0.5\sim 1$ 之间的随机数代表 1。若用上述方法,随机生成如下 4 个种串

01101
11000
01000
10011

这样便完成了遗传算法的准备工作。下面我们来介绍遗传算法的三个基本操作步骤:

- 复制(reproduction)
- 交叉(crossover)
- 变异(mutation)

1. 复制

复制过程是个体串按照它们的适配值进行复制。本例中目标函数值即可用作适配值。直观地看,可以将目标函数考虑成为利润、功效等的量度。其值越大,越符合我们的需要。按照适配值进行串复制的含义是值越大的串,在下一代中将有更多的机会提供一个或多个子孙。这个操作步骤主要是模仿自然选择现象,将达尔文的适者生存理论运用于串的复制。此时,适配值相当于自然界中的一个生物为了生存所具备的各项能力的大小,它决定了该串是被复制还是被淘汰。

复制操作可以通过随机方法来实现。若用计算机程序来实现,可考虑首先产生 0~1 之间均匀分布的随机数,若某串的复制概率为 40%,则当产生的随机数在 0~0.4 之间时该串被复制,否则该串被淘汰。另外一种直观的方法是使用轮盘赌的转盘。群体中的每个当前串按照其适配值的比例占据盘面上的成比例的一块区域。对应于本例,依照表 7.1 可以绘制出轮盘赌转盘如图 7.1 所示。

表 7.1 种群的初始串及对应的适配值

标号	串	适配值	占整体的百分数
1	01101	169	14.4%
2	11000	576	49.2%
3	01000	64	5.5%
4	10011	361	30.9%
总计(初始种群整体)		1170	100.0%

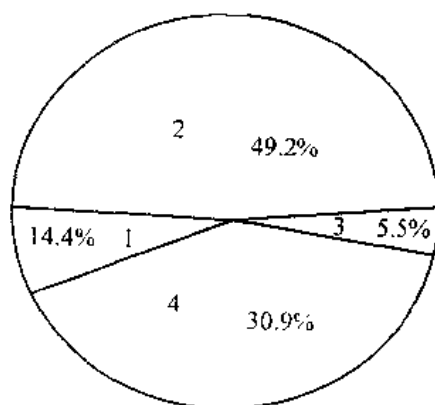


图 7.1 按适配值所占比例划分的轮盘

复制过程即是 4 次旋转这个经划分的轮盘,从而产生 4 个下一代的种群。例如对于该例,串 1 所占轮盘的比例为 14.4%。因此每转动一次轮盘,结果落入串 1 所占区域的概率也就是 0.144。可见对应大的适配值的串在下一代中将有较多的子孙。当一个串被选中进行复制时,此串将被完整地复制,然后将复制串添入匹配池。因此旋转 4 次轮盘即产生出 4 个串。这 4 个串是上一代种群的复制,有的串可能被复制一次或多次,有的可能被淘汰。本例中,经复制后的新的种群为

01101
11000
11000
10011

可见这里串 1 被复制了一次,串 2 被复制了两次,串 3 被淘汰了,串 4 也被复制了一次。

表 7.2 表示了复制操作之前的各项数据

表 7.2 复制操作之前的各项数据

串号	随机生成的 初始种群	x 值	$f(x)=x^2$	选择复制的 概率 $f_i/\sum f_i$	期望的复制 数 f_i/\bar{f}	实际得到的 复制数
1	01101	13	169	0.14	0.58	1
2	11000	24	576	0.49	1.97	2
3	01000	8	64	0.06	0.22	0
4	10011	19	361	0.31	1.23	1
总计			1170	1.00	4.00	4
平均			293	0.25	1.00	1
最大值			576	0.49	1.97	2

2. 交叉

交叉操作可以分为如下两个步骤。第一步是将新复制产生的匹配池中的成员随机两两匹配,第二步是进行交叉繁殖。具体过程如下。

设串的长度为 l , 则串的 l 个数字位之间的空隙标记为 $1, 2, \dots, l-1$ 。随机地从 $[1, l-1]$ 中选取一整数位置 k , 则将两个父母串中从位置 k 到串末尾的子串互相交换, 而形成两个新串。例如本例中初始种群的两个个体

$$A_1 = 0110 : 1$$

$$A_2 = 1100 : 0$$

假定从 1 到 4 间选取随机数, 得到 $k=4$, 那么经过交叉操作之后将得到如下两个新串

$$A'_1 = 01100$$

$$A'_2 = 11001$$

其中新串 A'_2 和 A'_1 是由老串 A_1 和 A_2 将第 5 位进行交换得到的结果。

下面举一个现实中的例子来说明上述复制和交叉的过程如何能获得性能的改进。假设一个工厂为生产某种产品需要经过好几个工序, 厂方向职工征集各道工序的方案。这相当于征求待优化问题的解, 而每一道工序则相当于待优化问题的一个参数。每名职工都提出各自的整体生产方案, 其中包括各道具体工序的设想。这样便形成了种群的初始代。全体职工在商讨会上互相交流, 总体效果好的方案受到较多关注, 效果差的方案可能被当场否定。各个方案之间互相取长补短, 从而使全体职工提出的方案从整体上达到一个更高的水平。遗传算法中的复制过程类似于将好的方案不断推广, 以供更多职工参考借鉴, 同时也淘汰较差的方案。交叉过程则类似于职工们互相取长补短, 以期望找出最佳的方案。

表 7.3 归纳了该例进行复制操作之后的结果。

表 7.3 复制操作之后的各项数据

新串号	复制操作后的 匹配池	匹配对象 (随机选取)	交叉点 (随机选取)	新种群	x 值	$f(x) \cdots x$
1	0110 1	2	4	01100	12	44
2	1100 0	1	4	11001	25	625
3	11000	4	2	11011	27	729
4	10011	3	2	10000	16	256
总计						1754
平均						439
最大值						729

从表 7.3 可以看出交叉操作的具体步骤。首先随机地将匹配池中的个体配对,结果串 1 和串 2 配对,串 3 和串 4 配对。此外,随机选取的交叉点的位置也如上表所示。结果串 1 (01101)和串 2(11000)的交叉点为 4,二者只交换最后一位,从而生成两个新串 01100 和 11001。剩下的两个串在位置 2 交叉,结果生成两个新串 11011 和 10000。

3. 变异

变异是以很小的概率随机地改变一个串位的值。如对于二进制串,即是将随机选取的串位由 1 变为 0 或由 0 变为 1。变异的概率通常是很小的,一般只有千分之几。这个操作相对于复制和交叉操作而言,是处于相对次要的地位,其目的是为了防止丢失一些有用的遗传因子,特别是当种群中的个体,经遗传运算可能使某些串位的值失去多样性,从而可能失去检验有用遗传因子的机会时,变异操作可以起到恢复串位多样性的作用。对于该例,变异概率设取为 0.001,则对于种群的总共 20 个串位,期望的变异串位数为 $20 \times 0.001 = 0.02(\text{位})$,所以本例中无串位值的改变。

从表 7.2 和表 7.3 可以看出,在经过一次复制、交叉和变异操作后,最优的和平均的目标函数值均有所提高。种群的平均适配值从 293 增至 439,最大的适配值从 576 增至 729。可见每经过这样的一次遗传算法步骤,问题的解便朝着最优解方向前进了一步。可见,只要这个过程一直进行下去,它将最终走向全局最优解,而每一步的操作是非常简单的,而且对问题的依赖性很小。

7.2.2 遗传算法的模式理论

前面我们通过一个简单的例子说明了按照遗传算法的操作步骤使得待寻优问题的性能朝着不断改进的方向发展。本节我们将进一步分析遗传算法的工作机理。

在上面的例子中我们发现,样本串的第 1 位的“1”使得适配值比较大,对于本例的函数及 x 的编码方式很容易验证这一点。它说明某些子串模式(schemata)在遗传算法的运行中起着关键的作用。首位为“1”的子串可以表示成这样的模式:1 * * * *,其中 * 是通配符,它既可代表“1”,也可代表“0”。该模式在遗传算法的一代一代地运行过程中不仅保留了下来,而且数量不断增加。正是这种适配值高的模式不断增加,才使得问题的性能不

断改进。

一般地,对于二进制串,在 $\{0,1\}$ 字符串中间加入通配符“*”即可生成所有可能模式。因此用 $\{0,1,*\}$ 可以构造出任意一种模式。我们称一个模式与一个特定的串相匹配是指:该模式中的1与串中的1相匹配,模式中的0与串中的0相匹配,模式中的*可以匹配串中的0或1。例如模式 $00*00$ 匹配两个串: $\{00100,00000\}$,模式 $*11*0$ 匹配四个串: $\{01100,01110,11100,11110\}$ 。可以看出,定义模式的好处是使我们容易描述串的相似性。

对于前面例子中的5位字串,由于模式的每一位可取0、1或*,因此总共有 $3^5=243$ 种模式。对一般的问题,若串的基为 k ,长度为 l ,则总共有 $(k+1)^l$ 种模式。可见模式的数量要大于串的数量 k^l 。一般地,一个串中包含 2^l 种模式。例如串11111是 2^5 个模式的成员,因为它可以与每个串位是1或*的任一模式相匹配。因此,对于大小为 n 的种群则包含有 2^l 到 $n \times 2^l$ 种模式。

为论述方便,首先定义一些名词术语。不失一般性,我们下面只考虑二进制串。设一个7位二进制串可以用如下的符号来表示

$$A=a_1a_2a_3a_4a_5a_6a_7$$

这里每个 a_i 代表一个二值特性(也称 a_i 为基因)。我们研究的对象是在时间 t 或第 t 代种群 $A(t)$ 中的个体串 $A_j, j=1,2,\dots,n$ 。任一模式 H 是由三字母集合 $\{0,1,*\}$ 生成的,其中*是通配符。模式之间仍有一些明显差别。例如,模式 $011*1**$ 比模式 $*****0*$ 包含更加确定的相似特性,模式 $1*****1*$ 比模式 $1*1*****$ 跨越的长度要长。为此,我们引入两个模式的属性定义:模式次数和定义长度。一个模式 H 的次数由 $O(H)$ 表示,它等于模式中确定位置(对于二进制,即0或1所在的位置)的个数。如模式 $H=011*1**$,其次数为4,记为 $O(H)=4$,若 $H=*****1***$,则 $O(H)=1$ 。模式 H 的长度定义为第一个和最后一个确定位置之间的距离,它用符号 $\delta(H)$ 表示。例如模式 $H=011*1**$,其第一个确定位置是1,最后一个确定位置是5,所以 $\delta(H)=5-1=4$ 。若模式 $H=*****0$,则 $\delta(H)=0$ 。

下面我们来分析遗传算法的几个重要操作对模式的影响。

1. 复制对模式的影响

设在给定的时间 t ,种群 $A(t)$ 包含有 m 个特定模式 H ,记为

$$m=m(H,t)$$

在复制过程中, $A(t)$ 中的任何一个串 A_j 以概率 $f_i/\sum f_i$ 被选中进行复制。因此我们可以期望在复制完成后,在 $t+1$ 时刻,特定模式 H 的数量将变为

$$m(H,t+1)=m(H,t)nf(H)/\sum f_i=m(H,t)f(H)/\bar{f}$$

或写成

$$\frac{m(H,t+1)}{m(H,t)}=\frac{f(H)}{\bar{f}}$$

其中 $f(H)$ 表示在时刻 t 时对应于模式 H 的串的平均适配值。 $\bar{f}=\sum f_i/n$ 是整个种群的平均适配值。

可见,经过复制操作后,特定模式的数量将按照该模式的平均适配值与整个种群平均

适配值的比值成比例地改变。换言之,适配值高于种群平均适配值的模式在下一代中的数量将增加,而低于平均适配值的模式在下一代中的数量将减少。另外,种群 A 的所有模式 H 的处理是并行进行的,即所有模式经复制操作后,均同时按照其平均适配值占总体平均适配值的比例进行增减。所以可以概括地说,复制操作对模式的影响是使得高于平均适配值的模式数量增加,低于平均值的模式数量减少。

为了进一步分析高于平均适配值的模式数量增长,设

$$f(H) = (1+c)\bar{f} \quad c > 0$$

则上面的方程可改写为如下的差分方程

$$m(H, t+1) = m(H, t)(1+c)$$

假定 c 为常数时可得

$$m(H, t) = m(H, 0)(1+c)^t$$

可见,对于高于平均适配值的模式数量将呈指数形式增长。

从对复制过程的分析可以看到,虽然复制过程成功地以并行方式控制着模式数量以指数形式增减,但由于复制只是将某些高适配值个体全盘复制,或是丢弃某些低适配值个体,而决不会产生新的模式结构,因而性能的改进是有限的。

2. 交叉对模式的影响

交叉过程是串之间的有组织的而又是随机的信息交换,它在创建新结构的同时,最低限度地破坏复制过程所选择的高适配值模式。为了观察交叉对模式的影响,下面考察一个 $l=7$ 的串以及此串所包含的两个代表模式。

$$A = 0111000$$

$$H_1 = * 1 * * * * 0$$

$$H_2 = * * * 1 0 * *$$

首先回顾一下简单的交叉过程,先随机地选择一个匹配伙伴,再随机选取一个交叉点,然后互换相对应的片断。假定对上面给定的串,随机选取的交叉点为 3,则很容易看出它对两个模式 H_1 和 H_2 的影响。下面用分隔符“|”标记交叉点。

$$A = 011|1000$$

$$H_1 = * 1 * | * * * 0$$

$$H_2 = * * * | 1 0 * *$$

除非串 A 的匹配伙伴在模式的固定位置与 A 相同(我们忽略这种可能性),模式 H_1 将被破坏,因为在位置 2 的“1”和在位置 7 的“0”将被分配至不同的后代个体中(这两个固定位置被代表交叉点的分隔符分在两边)。同样可以明显地看出,模式 H_2 将继续存在,因为位置 4 的“1”和位置 5 的“0”原封不动地进入到下一代的个体。虽然该例中的交叉点是随机选取的,但不难看出,模式 H_1 比模式 H_2 更易被破坏。因为平均看来,交叉点更容易落在两个头尾确定点之间。若定量的分析,模式 H_1 的定义长度为 5,如果交叉点始终是随机地从 $l-1=7-1=6$ 个可能的位置选取,那么很显然模式 H_1 被破坏的概率为

$$p_d = \delta(H_1)/(l-1) = 5/6$$

它存活概率为

$$p_s = 1 - p_d = 1/6$$

类似地,模式 H_2 的定义长度为 $\delta(H_2)=1$,它被破坏的概率为 $p_d=1/6$,存活的概率为 $p_s=1-p_d=5/6$ 。推广到一般情况,可以计算出任何模式的交叉存活概率的下限为

$$p_c \geq 1 - \frac{\delta(H)}{l-1}$$

其中大于号表示当交叉点落入定义长度内时也存在模式不被破坏的可能性。

在前面的讨论中我们均假设交叉的概率为 1, 一般情况若设交叉的概率为 p_c , 则上式变为

$$p_c \geq 1 - p_c \frac{\delta(H)}{l-1}$$

若综合考虑复制和交叉的影响, 特定模式 H 在下一代中的数量可用下式来估计

$$m(H, t+1) \geq m(H, t) \frac{f(H)}{\bar{f}} \left[1 - p_c \frac{\delta(H)}{l-1} \right]$$

可见, 对于那些高于平均适配值且具有短的定义长度的模式将更多地出现在下一代中。

3. 变异对模式的影响

变异是对串中的单个位置以概率 p_m 进行随机替换, 因而它可能破坏特定的模式。一个模式 H 要存活意味着它所有的确定位置都存活。因此, 由于单个位置的基因值存活的概率为 $(1-p_m)$, 而且由于每个变异的发生是统计独立的, 所以一个特定模式仅当它的 $O(H)$ 个确定位置都存活时才存活, 从而得到经变异后, 特定模式的存活率为

$$(1-p_m)^{O(H)}$$

由于 $p_m \ll 1$, 所以上式也可近似表示为

$$(1-p_m)^{O(H)} \approx 1 - O(H)p_m$$

综合考虑上述复制、交叉及变异操作, 可得特定模式 H 的数量改变为

$$m(H, t+1) \geq m(H, t) \frac{f(H)}{\bar{f}} \left[1 - p_c \frac{\delta(H)}{l-1} \right] (1 - O(H)p_m)$$

上式也可近似表示为

$$m(H, t+1) \geq m(H, t) \frac{f(H)}{\bar{f}} \left[1 - p_c \frac{\delta(H)}{l-1} - O(H)p_m \right]$$

其中忽略了一项较小的交叉相乘项。

变异的加入需对前面的分析结论略加改进。从而完整的结论为: 对于那些短定义长度、低次数、高于平均适配值的模式将在后代中呈指数级地增长。这个结论十分重要, 通常称它为遗传算法的模式理论。

根据模式理论, 随着遗传算法的一代一代地进行, 那些短的、低次数、高适配值的模式将越来越多, 最后得到的串即这些模式的组合, 因而可期望性能越来越得到改善, 并最终趋向全局的最优点。

7.3 遗传算法的实现及改进

7.3.1 遗传算法的实现

1. 问题的表示

对于一个实际的待优化的问题, 首先需要将其表示为适于遗传算法进行操作的二进制字符串。它一般包括以下几个步骤。

(1) 根据具体问题确定待寻优的参数;

(2) 对每一个参数确定它的变化范围,并用一个二进制数来表示。例如若参数 a 的变化范围为 $[a_{\min}, a_{\max}]$,用 m 位二进制数 b 来表示,则二者之间满足

$$a = a_{\min} + \frac{b}{2^m - 1} (a_{\max} - a_{\min})$$

这时参数范围的确定应覆盖全部的寻优空间,字长 m 的确定应在满足精度要求的情况下,尽量取小的 m ,以尽量减小遗传算法计算的复杂性。

(3) 将所有表示参数的二进制数串接起来组成一个长的二进制字串。该字串的每一位只有 0 或 1 两种取值。该字串即为遗传算法可以操作的对象。

上面介绍的是二进制编码,为最常用的编码方式。实际上也可采用其它编码方式。

2. 初始种群的产生

产生初始种群的方法通常有两种。一种是完全随机的方法产生。例如可用掷硬币或用随机数发生器来产生。设要操作的二进制字串总共 p 位,则最多可以有 2^p 种选择。设初始种群取 n 个样本 ($n \ll 2^p$)。若用掷硬币的方法可这样进行:连续掷 p 次硬币,若出现正面表示 1,出现背面表示 0,则得到一个 p 位的二进制字串,也即得到一个样本。如此重复 n 次即得到 n 个样本。若用随机数发生器来产生,可在 $0 \sim 2^p$ 之间随机地产生 n 个整数,则该 n 个整数所对应的二进制表示即为要求的 n 个初始样本。

上述随机产生样本的方法适于对问题的解无任何先验知识的情况。对于具有某些先验知识的情况,可首先将这些先验知识转变为必须满足的一组要求,然后在满足这些要求的解中再随机地选取样本。这样选择初始种群可使遗传算法更快地到达最优解。

3. 遗传算法的操作

图 7.2 表示出了标准遗传算法的操作流程图。

计算适配值可以看成是遗传算法与优化问题之间的一个接口。遗传算法评价一个解的好坏,不是取决于它的解的结构,而是取决于相应于该解的适配值。适配值的计算可能很复杂也可能很简单,它完全取决于实际问题本身。对于有些问题,适配值可以通过一个数学解析公式计算出来;而对于有些问题,则可能不存在这样的数学解析式子,它可能要通过一系列基于规则的步骤才能求得,或者在某些情况是上述两种方法的结合。当某些限制条件非常重要时,可在设计问题表示时预先排除这些情况,也可以在适配值中对它们赋与特定的罚函数。

复制操作的目的是产生更多的高适配值的个体,它对尽快收敛到优化解具有很大的影响。但是为了到达全局的最优解,必须防止过早的收敛。因此在复制过程中也要尽量保持样本的多样性。前面所介绍的转轮盘的复制方法是选择复制概率正比于目标

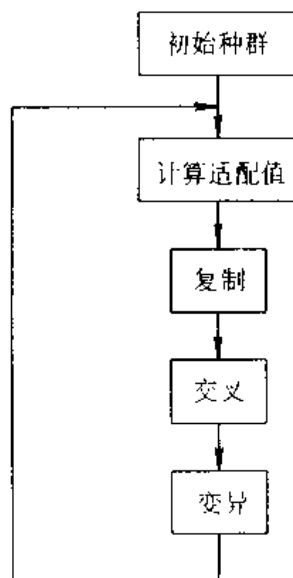


图 7.2 标准遗传算法的操作流程图

函数值(这时目标函数值等于适配值),因此也称之为比例选择法或随机选择法,这种方法可使收敛得比较快;但当个体适配值相差很大时,有可能损失样本的多样性而出现过早收敛的问题。针对此问题,提出了另外一种方法,该方法按目标函数值的大小排序,重新计算适配值,再按适配值的大小比例选择复制概率,因此它又称基于排序的选择法。例如若算得 n 个样本的目标函数值 J_i ,并将它们按大小排序: $J_1 < J_2 < \dots < J_n$,然后按下式计算适配值

$$f_i = kr_i/n \quad i=1,2,\dots,n$$

其中 r_i 是次序号, k 是用来控制适配值之间差别的常数。若 J_i 表示代价函数,即 J_i 越小性能越好,则可取适配值

$$f_i = k(n-r_i)/n \quad i=1,2,\dots,n$$

以上的选取是使得适配值按序号数线性变化,若要求按某种非线性关系变化,也可取某适配值为序号数的某种非线性关系,如 $f_i = \exp(kr_i/n)$ 或 $f_i = \exp[k(n-r_i)/n]$ 。可见,这第二种选择复制方法是基于目标函数的排序而不是目标函数本身的大小,因而避免了适配值差别太大而导致样本的多样性损失太多。

对于交叉操作,前面介绍了最简单的一种方法即单点交叉,交叉点是随机选取的。此外,也还有其它一些交叉的方法。下面介绍一种掩码交叉的方法。这里掩码是指长度与被操作的个体串相等的二进制位串,其每一位的 0 或 1 代表着特殊的含义。若某位为 0,则进行交叉的父母串的对应位的值不变,即不进行交换。而当某位为 1 时,则父母串的对应位进行交换。如下面的例子:

父母 1:001111

父母 2:111100

掩码: 010101

子女 1:011110

子女 2:101101

不难看出,对于前面描述过的单点交叉操作,相当于掩码为 $0\dots01\dots1$;类似地,我们很容易定义两点交叉,其对应的掩码为 $0\dots01\dots10\dots0$ 。

变异是作用于单个串,它以很小的概率随机地改变一个串位的值,其目的是为了防止丢失一些有用的遗传模式,增加样本的多样性。

标准的遗传算法通常包含上述三个基本操作:复制、交叉和变异。但对于某些优化问题,如布局问题、旅行商问题等,有时还引入附加的反转(inversion)操作。它也作用于单个串,在串中随机地选择两个点,然后在将这两个点之间子串加以反转,如下例:

老串:10:1100:11101

新串:10:0011:11101

4. 遗传算法中的参数选择

在具体实现遗传算法的过程中,尚有一些参数需要事先选择,它们包括初始种群的大小 n 、交叉概率 p_c 、变异概率 p_m ,也时还包括反转概率 p_i 。这些参数对遗传算法的性能都有很重要的影响。一般说来,选择较大数目的初始种群可以同时处理更多的解,因而容易找到全局的最优解,其缺点是增加了每次迭代所需要的时间。

交叉概率的选择决定了交叉操作的频率。频率越高,可以更快地收敛到最有希望的最优解区域;但是太高的频率也可能导致收敛于一个解。

变异概率通常只取较小的数值,一般为 $0.001\sim 0.1$ 。若选取高的变异率,一方面可以增加样本模式的多样性,另一方面可能引起不稳定。但是若选取太小的变异概率,则可能难于找到全局的最优解。

自从遗传算法产生以来,研究人员从未停止过对遗传算法进行改进的探索,下面除介绍一些典型的改进思路外,重点介绍一种改进的遗传算法。

7.3.2 遗传算法的改进

1. 自适应变异

如果双亲的基因非常相近,那么所产生的后代相对于双亲也必然比较接近。这样所期待的性能改善也必然较小。这种现象类似于“近亲繁殖”。所以,群体基因模式的单一性不仅减慢进化历程,而且可能导致进化停滞,过早地收敛于局部的极值解。Darrel Whitley提出了一种如下的自适应变异的方法,在交叉之前,以海明(Hamming)距离测定双亲基因码的差异,根据测定值决定后代的变异概率 p_m 。若双亲的差异较小,则选取较大的变异概率 p_m 。通过这种方法,当群体中的个体过于趋于一致时,可以通过变异的增加来提高群体的多样性,也即增强了算法维持全局搜索的能力;反之,当群体已具备较强的多样性时,则减小变异率,从而不致破坏优良的个体。

2. 部分替换法

设 P_G 为上一代进化到下一代时被替换的个体的比例,则按此比例,部分个体被新的个体所取代,而其余部分的个体则直接进入下一代。 P_G 越大,进化得越快,但算法的稳定性和收敛性将受到影响;而 P_G 越小,算法的稳定性较好,但进化速度将变慢。可见,应该寻求运行速度与稳定性、收敛性之间的协调平衡。

3. 优秀个体保护法

这种方法是对于每代中一定数量的最优个体,使之直接进入下一代。这样可以防止优秀个体由于复制、交叉或变异中的偶然因素而被破坏掉。这是增强算法稳定性和收敛性的有效方法。但同时也可能使遗传算法陷入局部的极值范围。

4. 移民法

移民算法是为了加速淘汰差的个体以及引入个体多样性的目的而提出的。所需的其它步骤是用交叉产生出的个体替换上一代中适配值低的个体,继而按移民的比例,引入新的外来个体来替换新一代中适配值低的个体。这种方法的主要特点是不断地促进每一代的平均适配值的提高。但由于低适配值的个体很难被保存至下一代,而这些低适配值的个体中也可能包含着一些重要的基因模式块,所以这种方法在引入移民增加个体多样性的同时,由于抛弃低适配值的个体又减少了个体的多样性。所以,这里也需要适当的协调平衡。

5. 分布式遗传算法

该方法将一个总的群体分成若干子群,各子群将具有略微不同的基因模式,它们各自的遗传过程具有相对的独立性和封闭性,因而进化的方向也略有差异,从而保证了搜索的

充分性及收敛结果的全局最优性,另一方面,在各子群之间又以一定的比率定期地进行优良个体的迁移,即每个子群将其中最优的几个个体轮流送到其它子群中,这样做的目的是期望使各子群能共享优良的基因模式以防止某些子群向局部最优方向收敛。

分布式遗传算法模拟了生物进化过程中的基因隔离和基因迁移,即各子群之间既有相关的封闭性,又有必要的交流和沟通。研究表明,在总的种群个数相同的情况下,分布式遗传算法可以得到比单一种群遗传算法更好的效果。不难看出,这里的分布式遗传算法与前面的移民法具有类似的特点。

7.3.3 改进的遗传算法举例

前面介绍了许多关于遗传的算法的改进思路,下面具体介绍一个我们提出的改进的遗传算法。它是在两个高低不同的层次上都使用了遗传算法。

1. 生物模型

首先我们来考察该改进的遗传算法的生物模型。遗传算法是对一个群体进行操作,该群体相当于自然界中的一群人。第一步的复制是以现实世界中的优胜劣汰现象为背景的。第二步的交叉则相当于人类的结婚和生育。第三步的变异则与自然界中偶然发生的变异是一致的,人类偶尔出现的返祖现象便是一种变异。由于包含着对模式的操作,遗传算法不断地产生出更加优良的个体,正如人类不断向前进化一样。

上面我们分析了标准遗传算法中的几个典型操作均可与生物(尤其是人类)的进化过程相对应。如果我们再仔细研究遗传算法的操作对象(种群),我们发现它实际上对应的是-一群人,而不是整个人类。一群人随着时间的推移而不断地进化,并具备越来越多的优良品质。然而由于他们的生长、演变、环境和原始祖先的局限性,经过相当一段时间后,他们将逐渐进化到某些特征相对优势的状态(例如中国人都是黄皮肤、黑眼睛以及特有的文化和社会传统习惯等),我们定义这种状态为平衡态。当一个种群进化到这种状态,这个种群的特性便不再有很大的变化。一个标准的遗传算法,从某一初始代开始,并且各项参数都设定(如采用什么样的复制和交叉操作,以及采用多大的交叉概率和变异概率等),也会达到平衡态。此时,结果群体中的优良个体仅包含某些类的优良模式,因为该遗传算法的设置特性(它包括初始种群的特性及遗传参数)使得这些优良模式的各个单位未能得到平等的竞争机会。

现实世界中许多民族,每个民族都有各自的优缺点。为了能产生出各方面都十分杰出的人,应该使各民族之间定期地大量移民和通婚,这样就可以打破各个民族的平衡态而推动他们达到更高层的平衡态,也即使整个人类向前进化。现实生活中的例子可在生物学家的实验室中找到,他们为了改良动植物的品种,常常采用杂交、嫁接等措施,即是为了这个目的。而在我们人类中这样的现象却是不多见的。然而,人类历史上连绵不断的种族战争,可以将其看成为改进的遗传算法的一部分现实模型。战争猛烈地打破了民族的平衡态,其结果是征服者赶走了失败者并通过移民占领了原属于被征服者的领土,实现了民族之间的移民和通婚。虽然战争是令人憎恶的,但它却在客观上促进了整个人类的进化。当然,决不能因此来美化战争,人类完全可以有目的地通过和平的方式来进行各民族的移民和通婚,从而达到人类进化的目的。

2. 改进的遗传算法

仿照上述的生物模型,我们可构造如下的改进的遗传算法。对于一个问题,我们首先随机地生成 $N \times n$ 个样本 ($N \geq 2, n \geq 2$)。然后将它们分成 N 个子种群,每个子种群包含 n 个样本。对每个子种群独立运行各自的遗传算法,记它们为 $GA_i (i=1, 2, \dots, N)$ 。这 N 个遗传算法最好在设置特性方面有较大的差异,这样可以为将来的高层遗传算法产生出更多种类的优良模式。

在每个子群的遗传算法运行到一定次数后,将 N 个遗传算法的结果种群记录到二维数组 $R[1 \dots N, 1 \dots n]$ 中,则 $R[i, j] (i \in 1 \dots N, j \in 1 \dots n)$ 表示 GA_i 的结果种群的第 j 个个体。同时,将 N 个结果种群的平均适配值记录到数组 $A[1 \dots N]$ 中,则 $A[i] (i \in 1 \dots N)$ 表示 GA_i 的结果种群的平均适配值。

高层遗传算法与普通的遗传算法操作相类似,也分成如下三个步骤。

(1) 复制

基于数组 $A[1 \dots N]$,即个 N 遗传算法的平均适配值,对数组 R 进行复制操作,结果一些 $R[p, 1 \dots n] (1 \leq p \leq N)$ 被复制,而一些 $R[q, 1 \dots n] (1 \leq q \leq N)$ 被淘汰。也就是说,一些结果种群(GA_p)由于它们的种群平均适配值高而被复制,甚至复制多次;另一些结果种群(GA_q)则可能由于其平均适配值低而被淘汰。

(2) 交叉

如果 $R[\theta, 1 \dots n]$ 和 $R[\phi, 1 \dots n]$ 被随机地匹配到一起,而且从位置 x 进行交叉 ($1 \leq \theta, \phi \leq N, 1 \leq x \leq n-1$),则 $R[\theta, x+1 \dots n]$ 和 $R[\phi, x+1 \dots n]$ 互相交换相应的部分。这一步骤相当于交换 GA_θ 和 GA_ϕ 中的结果种群的 $n-x$ 个个体。

(3) 变异

以很小的概率将少量的随机生成的新个体替换 $R[1 \dots N, 1 \dots n]$ 中随机抽取的个体。

至此,高层遗传算法的第一轮运行结束。 N 个遗传算法 $GA_i (i=1, 2, \dots, N)$ 现在可以从相应于新的 $R[1 \dots N, 1 \dots n]$ 的更新后的种群继续各自的操作。

在 N 个 GA_i 再次各自运算到一定次数后,再次更新数组 $R[1 \dots N, 1 \dots n]$ 和 $A[1 \dots N]$,并开始高层遗传算法的第二轮运行。如此继续循环操作,直至得到满意的结果。图7.3示出了该改进的遗传算法的操作流程图。

根据本章前面所述模式理论,模式在遗传算法中起着十分关键的作用。在改进的遗传算法中, N 个遗传算法中的每一个在经过一段时间后均可以获得位于个体串上一些特定位置的优良模式。通过高层遗传算法的操作, $GA_i (i=1, 2, \dots, N)$ 可以获得包含不同种类的优良模式的新个体,从而为它们提供了更加平等的竞争机会。该改进的遗传算法与并行或分布式遗传算法相比,在上一层上的个体交换是一个突破,它不需要人为地控制应交流什么样的个体,也不需要人为地指定处理器将传送出的个体送往哪一个处理器,或者从哪一个处理器接收个体。这样,改进的遗传算法不但在每个处理器上运行着遗传算法,同时对各处理器不断生成的新种群进行着高一层的遗传算法的运算和控制。

3. 试验举例

下面我们通过一个具体的例子来将该改进的遗传算法与标准的遗传算法进行比较。由于遗传算法对问题的依赖性很小,所以为使描述简便起见,这里只给出一个函数优化的

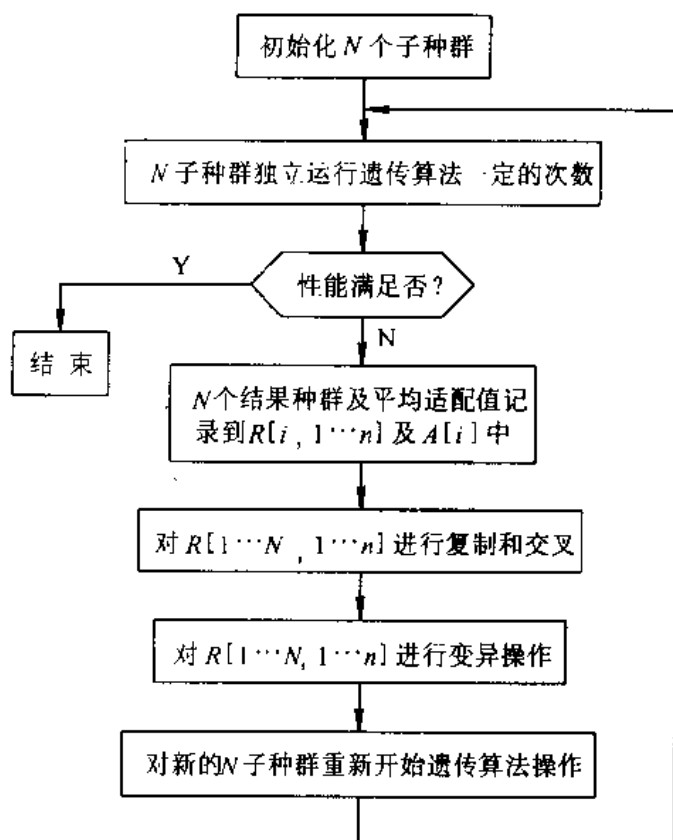


图 7.3 改进的遗传算法的操作流程图

简单例子。

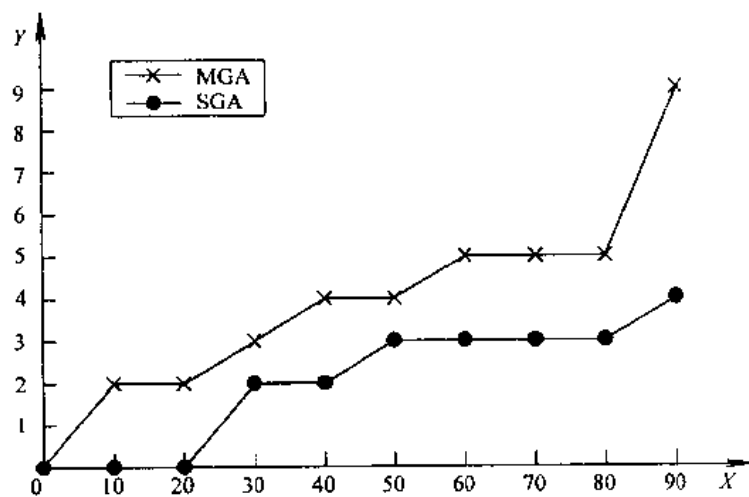
设待求解的问题是找到函数 $y=x^{30}$ ($x \in [0,1]$) 的最大值。自变量 x 的作用域被转化成 28 位的二进制串。 $x=0$ 表示为“0000000000000000000000000000”, $x=1$ 表示为“11111 1111111111111111111111111111”。被比较的标准遗传算法的初始种群包含 100 个随机生成的个体,取交叉概率 $p_c=0.3$,变异概率 $p_m=0.05$ 。

我们将 100 个个体分为 10 组,每组各自独立地运行遗传算法。它们除了种群大小与被比较的标准遗传算法不同外, p_c 和 p_m 的设置也略有不同。表 7.4 给出了具体的设置参数,其中 SGA 表示操作对象为 100 个个体的标准的遗传算法, GA_i ($i=1,2,\dots,10$) 表示操作对象均为 10 个个体的遗传算法。

表 7.4 举例问题的参数设置

	SGA	GA_1	GA_2	GA_3	GA_4	GA_5	GA_6	GA_7	GA_8	GA_9	GA_{10}
p_c	0.3	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.3	0.9
p_m	0.005	0.005	0.002	0.005	0.004	0.005	0.006	0.005	0.008	0.005	0.01

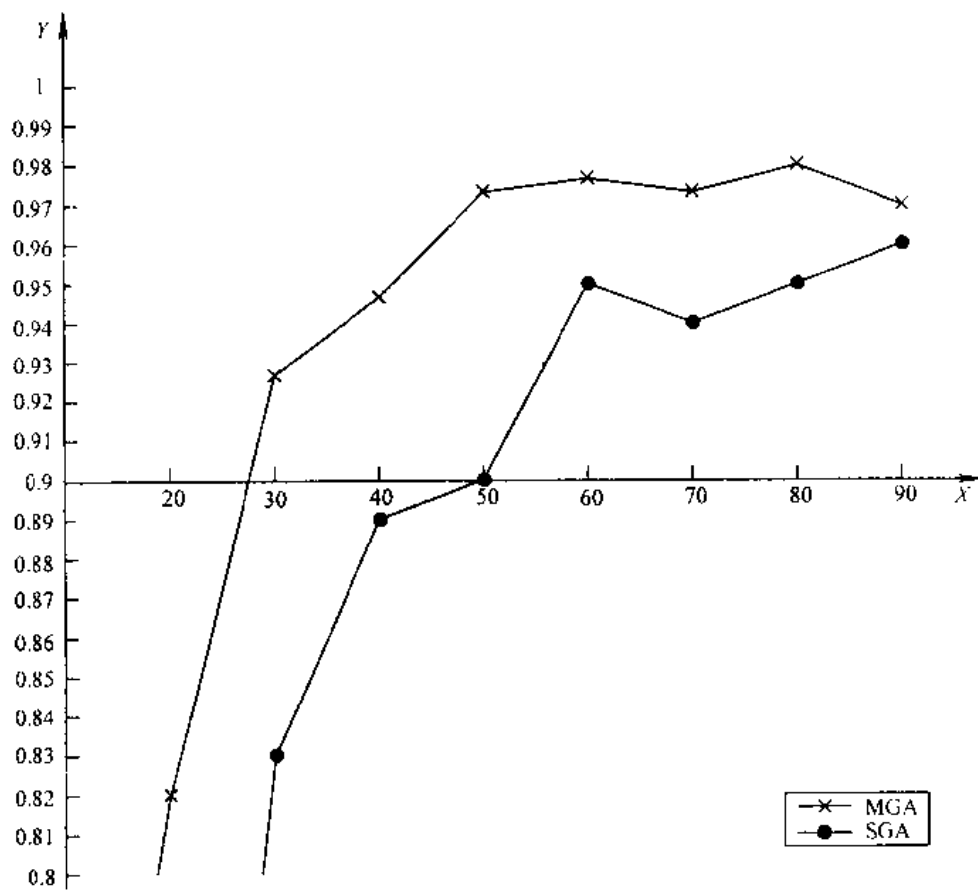
10 个子种群的遗传算法每运行 10 次,进行一次高层遗传算法的操作,结果,高层遗



X——标准遗传算法运行的代数；

Y——最优个体适配值在小数点后9的个数。

图 7.4 改进的遗传算法与标准的遗传算法最优个体适配值的比较



X——标准遗传算法运行的代数；

Y——整个种群的平均适配值。

图 7.5 改进的遗传算法与标准的遗传算法平均适配值的比较

传算法仅运行 2 或 3 次后,这 10 个遗传算法的结果种群的平均适配值即超出被比较的标准遗传算法的结果平均适配值很多(在相同运算量下的比较)。这种结果一直保持到最后。此改进遗传算法在高层遗传算法的第 8 次操作后的 10 个遗传算法运行 10 次后找到最优解 $x=1$,这相当于 9 000 次计算适配值的计算量。而被比较的标准遗传算法则是在第 250 代才找到最优解 $x=1$,它相当于 25 000 次计算适配值的计算量。此外,在相同的运算量下,无论从整个结果种群的平均适配值,还是从得到的最优个体的适配值来看,改进的遗传算法始终优于被比较的标准遗传算法。图 7.4 和 7.5 形象地给出了两者比较的结果。其中 SGA 表示标准的遗传算法,MGA 表示改进的遗传算法。其中 SGA 表示标准的遗传算法,MGA 表示改进的遗传算法。

7.4 遗传算法应用举例

遗传算法由于其对问题的依赖性较小、可以求得全局最优解等特点,吸引了各方面人士的兴趣,并已在许多领域中获得了应用。本节着重它在与智能控制有关的几个方面的应用举例。

7.4.1 遗传算法在模糊逻辑控制中的应用

前面第 2 章详细介绍了模糊逻辑控制,该控制方法比较类似于人的控制方式,因此可借鉴操作人员或专家的经验来帮助选择控制器的结构和参数。然而,由于一个模糊逻辑控制器所要确定的参数很多,专家的经验只能起到一个指导作用,很难根据它准确地定出各项参数,因而实际上还要靠不断地反复试凑。这实质上是一个寻优的过程。遗传算法可以应用于该寻优过程,较有效地确定出模糊逻辑控制器的结构和参数。

利用遗传算法来解决具体应用问题的关键是以下两点:(1) 问题表示,即如何将实际问题表示为遗传算法所能处理的形式;(2) 确定目标函数,它是计算每个样本适配值的基础。下面通过具体例子来说明这个过程。

1. 液位系统的模糊控制

图 7.6 所示为液罐的示意图。问题是要求控制罐内的液面维持在一定的高度 h ,控制

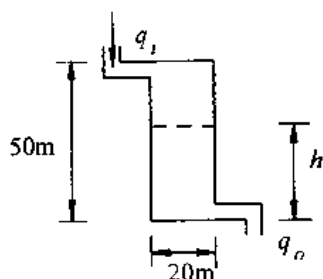


图 7.6 液罐示意图

量有两个:流入的流量 q_i 和流出的流量 q_o ,它的最大容许范围为 $0-200\text{m}^3/\text{s}$,但每次调整只容许改变 $20\text{m}^3/\text{s}$,影响控制量的因素主要有两个:液面高度 h 及高度的变化率 $\dot{h}=dh/dt$ 。对于该问题,若采用模糊控制的方法,需要确定以下几方面的参数。

- (1) 各个变量(h, \dot{h}, q_i, q_o)的模糊分级数;
- (2) 模糊规则的个数;
- (3) 模糊规则的前件和后件;
- (4) 模糊集合的隶属度函数。

对该例设前三项都已事先确定,需调整的只是第(4)项。设 h 的模糊分级数为 4,其余三个量(\dot{h}, q_i, q_o)的模糊分级数为 5,设模糊集合的隶属度函数均采用三角形,如图 7.7

所示。

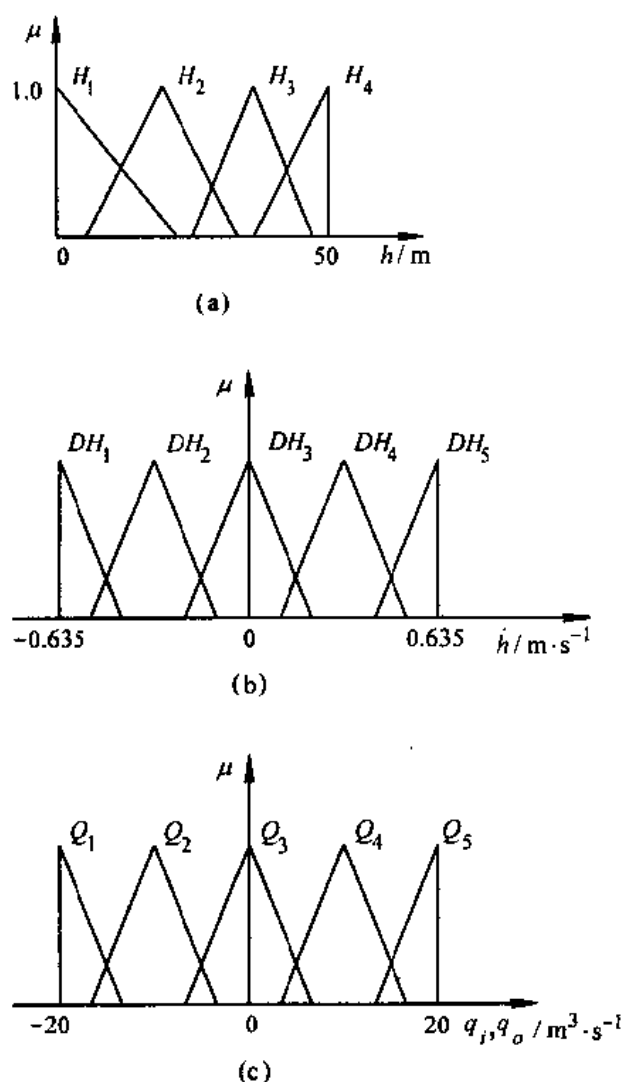


图 7.7 描述 h 、 \dot{h} 、 q 和 q_0 的隶属度函数

设两端的隶属度函数均为直角三角形,每个隶属度函数只有一个参数(斜边与横轴交点)需要调整。中间的隶属度函数均为等腰三角形,每个隶属度函数有两个参数(与横轴的两个交点)需要调整。这样 h 的隶属度函数有 6 个参数, \dot{h} 的隶属度函数有 8 个参数,设 q 和 q_0 的隶属度函数形状完全一样,所以它们的隶属度函数也是 8 个参数。这样总共需要调整的参数为 22 个。若每个参数均用 6 位二进制数表示,则每个样本可用一个 132 位的二进制字符串来表示。

上面描述了问题的表示,即将待寻优的参数表示为一个长的二进制位串。第二个关键问题是如何定义目标函数。在该例中,目标函数定义为

$$J = \sum_{i=1}^4 \sum_{j=0}^{20} (25 - h_{ij})^2$$

其中 $h=25$ 是期望的高度, h_{ij} 表示对于第 i 次仿真情况的第 j T 时刻实际液面高度。这里采样周期 T 取为 1 秒。4 种仿真情况分别为

$$(1) h(0)=0, \dot{h}(0)=0.6366$$

$$(2) h(0)=50, \dot{h}(0)=-0.6366$$

$$(3) h(0)=10, \dot{h}(0)=-0.3183$$

$$(4) h(0)=40, \dot{h}(0)=0.3183$$

仿真计算时,控制对象的计算模型为

$$h(k+1)=h(k)+\frac{q_i-q_o}{S}T$$

其中 S 为液罐的横截面积。对于模糊逻辑控制器,其模糊规则具有如下的形式

R_j :若 h 是 H_j and \dot{h} 是 DH_j , 则 q_i 是 Q_i^j and q_o 是 Q_o^j

$j=1,2,\dots,N$, 对于该例, $N=20$ 。因此对于每一个样本(132 位的字串),就相当于给定了所有的隶属度函数的形状。从而可对相应于该组参数的系统进行仿真计算。

在用遗传算法进行寻优计算时,样本大小取为 500,取交叉概率 $p_c=0.8$,变异概率 $p_m=0.01$,最大计算到第 80 代。经遗传算法求得的参数所组成的模糊控制器取得了满意的控制效果。图 7.8 显示两种初始条件下的仿真结果,从图中看出,用遗传算法设计的模糊控制器的性能优于常规设计的模糊控制器,且所设初始条件并不是计算目标函数时所用到的情况。

2. pH 值控制

上面的例子由于问题比较简单,容易建立模糊控制的规则,一般情况下,也可以利用遗传算法来帮助设计模糊控制规则库,这里以 pH 值控制为例来加以说明。

在该例中,首先设定语言变量值的隶属度函数,然后利用遗传算法来设计模糊规则,最后固定这些模糊规则,再用遗传算法来调整隶属度函数。

该 pH 值控制问题类似于上面的液位控制系统。具体装置仍为如图 7.6 所示的圆柱形罐,这里需要控制的不是其中的液面高度,而是其中溶液的 pH 值。该容器有两个输入口,一个输入基本的溶液,另一个输入酸性溶液。控制的目的是通过调整两个入口阀门,以使得容器中的溶液为中性($\text{pH}=7$)。当 pH 值控制到设定点时将两个入口阀门均关闭,以维持容器内的 pH 值为定值。

该模糊控制器有四个输入量:溶液的 pH 值、pH 值随时间的变化率以及两个入口阀门的当前位置,每个输入变量用三个模糊集合来表示。模糊控制器有两个输出量:两个入口流量阀门的位置,这两个控制变量总共用 7 个模糊集合来表示。

该模糊控制器最多可以有 81 条规则。也就是说,这些规则的前件是控制器输入变量的所有可能的组合,但对设计者来说,要准确地定出所有规则的后件是比较困难的。当然,对于某些极限情况是很容易写出规则后件的,如若 pH 值很低且注入酸性溶液的阀门设置为全开以及注入基本溶液的阀门设置为全关闭,则应停止注入酸性溶液并启动注入基本溶液。但在很多情况下设计者很难确定合适的后件,这时可应用遗传算法来帮助确定。

由于模糊规则的后件共有 7 种可能的选择,因此可用一个 3 位二进制串来表示。如果 81 条规则的后件都要求确定,则可用一个 243 位长的字串来表示任何一个可能的模糊规则库。若排除掉那些可以明显确定的规则外,在该例中实际上只有 16 条规则的后件需要确定,也即总共的字串长度为 48 位。

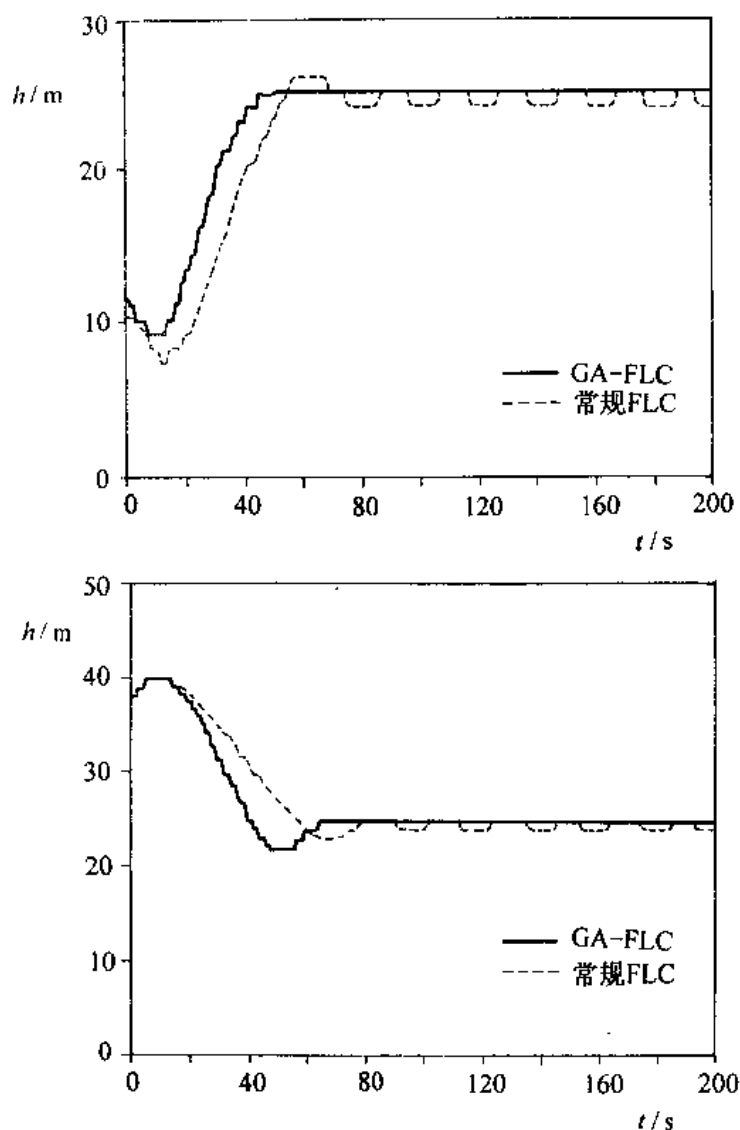


图 7.8 基于遗传算法的模糊控制器与常规模糊控制器性能的比较

在利用遗传算法确定了规则库后,剩下的问题是进一步调整隶属度函数。该例中共有四个变量: pH 值、pH 值变化率及两个入口阀门的位置,每个变量用如图 7.9 所示的三个模糊集合来表示。设隶属度函数为对称分布的三角形,则每个变量的隶属度函数只需 2 个参数来表示,总共需 8 个参数。若每个参数用一个 6 位二进制位串来编码,则总长为 48 位的字串可用来完整地描述隶属度参数。从而可再利用遗传算法来确定出这些参数。

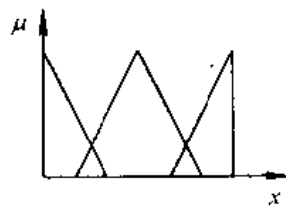


图 7.9 对称的三角形隶属度函数

该例的控制目标是驱动系统到设定点: $\text{pH}=7$, 其余 3 个变量均为 0, 其后维持系统在该设定点。为此代价函数设定为

$$J = \sum_{i=1}^2 \sum_{j=0}^{50} (\text{pH}_{ij} - 7)^2$$

其中 $\text{pH}=7$ 是期望值。 pH_{ij} 表示对于第 i 次情况下第 jT 时刻的实际 pH 值, 这里采样周期 T 取为 1 秒。优化的目标是使得 J 最小。图 7.10 显示了基于遗传算法的模糊控制器与常规模糊控制器的性能的比较, 显然前者具有更好的性能。

7.4.2 遗传算法在神经网络控制中的应用

在神经网络用于系统建模和控制时, 它要利用神经网络的函数估计及分类功能。设计神经网络的关键是如何确定神经网络的结构及连接权系数。它实质上也是一个优化问题,

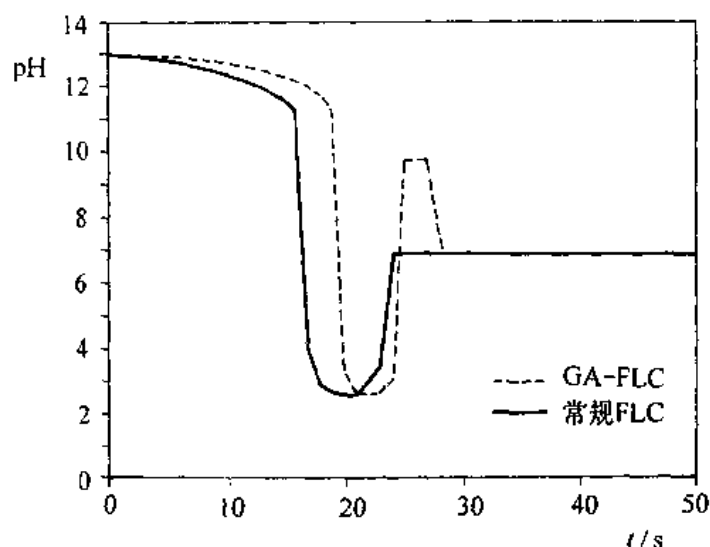


图 7.10 基于遗传算法的模糊控制器与常规模糊控制器性能的比较

其优化的目标是使得所设计的神经网络具有尽可能好的函数估计及分类功能。对于应用最为广泛的 BP 网络, 网络的结构(网络层数和隐层的结点数)主要靠经验和试凑来确定, 而连接权系数则通过 BP 学习算法来确定。BP 学习算法本质上是一种梯度寻优方法, 因而容易陷入局部极值, 它取决于初始权值的选择。这是 BP 学习算法的一个主要缺点。

前面介绍了遗传算法可用于优化计算, 因而它也可用于神经网络的设计, 这里以多层前馈网为例来说明。遗传算法与神经网络的结合可在不同的层次进行。低层的结合是若神经网络的结构已定, 利用遗传算法来确定连接权系数, 高层的结合是利用遗传算法来设计神经网络的结构。完全的应用则是上述两者的结合。下面介绍一个用遗传算法来确定神经网络控制器的例子。在该例中, 遗传算法主要用来确定连接权系数。

1. 改进的遗传算法

前面介绍的最基本的遗传算法虽具有实现简单及鲁棒性好的特点, 但是它也有以下几方面的局限性。第一, 当问题的规模很大时, 遗传算法的性能将变差; 第二, 当样本串中缺乏重要的特征基因时, 遗传算法可能出现过早收敛而不能达到最优解。其原因是遗传算法过分依赖于交叉的步骤, 而变异的概率通常比较小, 不足以跳出局部的搜索空间, 当样

本比较小时更容易出现这样的情况。为了避免这种情况,前面我们给出了几种改进遗传算法的思路。本例采取了适配值修正和变异概率修正的改进措施。

适配值采用如下的方法来进行修正。设 f 为按通常方法算得的适配值, \bar{f} 为平均的适配值, 则修正的适配值取为

$$f' = \begin{cases} k\bar{f} & f \geq k\bar{f} \\ f & f < k\bar{f} \end{cases}$$

其中 $k > 1$ 。该修正的作用在于降低那些适配值太大的串的影响, 以防过早地收敛。或者说它的作用是减慢收敛速度而扩大搜索空间。按照模式理论, 随着遗传算法的逐代演化, 平均适配值将越来越大, 问题将逐渐朝着优化的方向发展。

变异概率采用如下的方法进行修正

$$p_m(i+1) = \begin{cases} p_{mh} & \text{当连续 } N \text{ 代的最高适配值均相同} \\ p_m(i) & \text{当 } k_1 p_m(i) \leq p_{ml} \\ k_1 p_m(i) & \text{其它} \end{cases}$$

其中 i 表示遗传算法演化的代数, p_{mh} 和 p_{ml} 表示变异概率的高限和低限, k_1 是小于 1 的常数。上面的修正方法可以使得当出现过早收敛时自动加大变异概率, 以便扩大搜索空间。同时在每次迭代演化时保留最好的样本, 以防止由于较高的变异概率而破坏获得的最好结果。根据经验, 通常取

$$p_{mh} \in [0.5, 1], p_{ml} \in [0, 0.1], k \in [1, 10], k_1 \in [0.8, 1], N \in [1, 100]$$

这里设将上述改进的遗传算法(MGA)应用于多层前馈神经网络的连接权参数的确定。这些参数可以如前面介绍的那样采用二进制编码, 也可以为了减小串的长度而直接用十进制数来表示。对于用十进制来表示的字符串, 变异操作则采用对一个数位附加一个随机数的方法来实现。

2. 倒立摆的神经网络控制

设控制对象为如图 7.11 所示的单摆系统。其动力学方程为

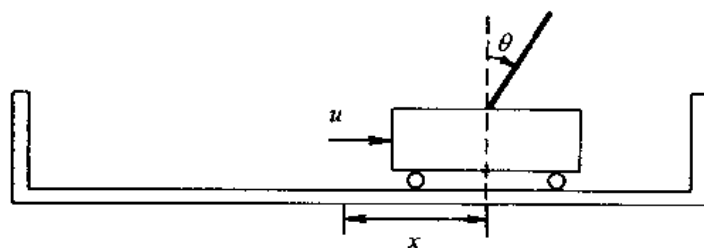


图 7.11 单倒立摆系统

$$\ddot{\theta} = \frac{(M+m)g\sin\theta - \cos\theta[u + ml\dot{\theta}^2\sin\theta]}{(4/3)(M+m)l - ml(\cos\theta)^2}$$

$$\ddot{x} = \frac{u + ml[\dot{\theta}^2\sin\theta - \ddot{\theta}\cos\theta]}{M+m}$$

其中 $M=1.0\text{kg}$ 是小车的质量, $m=0.1\text{kg}$ 是杆的质量, $l=0.5\text{m}$ 是杆长的一半。 θ 是摆角, x 是小车偏离中心位置的距离。

控制系统采用如图 7.12 所示的结构。其中 NNC 表示神经网络控制器, 该控制器的输入为 4 个量: $\theta, \dot{\theta}, x, \dot{x}$, 输出为控制量。设控制器采用多层前馈网络来实现。这里由于不能获得期望的控制输出, 因此不能直接应用 BP 学习算法。从而本例中采用上面给出的 MGA 来帮助确定神经网络的连接权。

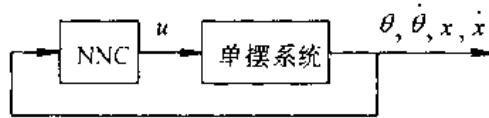


图 7.12 单摆系统的神经网络控制

这里的神经网络采用三层的前馈网络, 其输入层有四个结点, 它们分别对应于四个输入: $\theta, \dot{\theta}, x, \dot{x}$, 输出层一个结点, 对应于控制量 u , 隐层设有 10 个结点。隐层结点的非线性激发函数采用

$$f_1(x) = \frac{2}{1 + e^{-x}} - 1$$

输出层结点的激发函数为

$$f_2(x) = 10 \left(\frac{2}{1 + e^{-x}} - 1 \right)$$

以使得输出量能在 -10N 和 $+10\text{N}$ 之间连续变化。

这里将所有的连接权及阈值参数均采用十进制编码, 每个参数均在 -10 到 10 之间变化。网络的样本取为 100, MGA 的其它参数取为: $p_i=0.8, p_{mh}=0.5, p_{ml}=0.03, N=5, k=2.5, k_i=0.9$ 。

该网络的适配值取为该倒立摆系统能够处于正常运行状态的仿真时间。所谓正常运行是指摆角 θ 不超过 $\pm 15^\circ$ 。

利用上述 MGA, 当迭代演化到第 200 代时, 所获得的神经网络控制器可以使系统正常运行 100 000 时间拍而不出现失败的情况。从而获得了一个较好的神经网络控制器。

3. 机器人控制

设控制对象为如图 7.13 所示的单臂机械手系统, 其动力学方程为

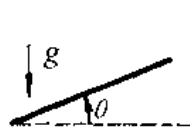


图 7.13 单臂机械手

$$\tau = \frac{ml^2}{3}\ddot{\theta} + \frac{mgl}{2}\cos\theta$$

其中 $m=10\text{kg}$ 为杆的质量, $l=1.0\text{m}$ 是杆的长度, τ 是控制力矩, g 是重力加速度, θ 是关节角。

控制系统采用如图 7.14 所示的结构。其中 NNC 表示神经网络控制器。该控制器的输入量为两个: $\bar{\theta}=\theta_d-\theta$ 和 $\dot{\bar{\theta}}=\dot{\theta}_d-\dot{\theta}$, 输出为控制量 τ

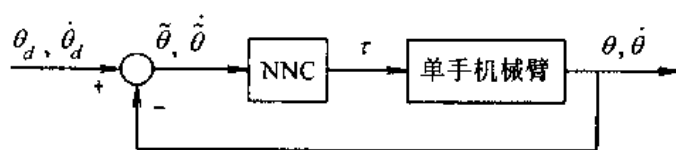


图 7.14 单臂机械手的神经网络控制

这里的神经网络采用三层前馈网,其输入层有 2 个结点,分别对应于两个输入: θ 和 $\dot{\theta}$,输出层有 1 个结点,对应于控制量 τ 。隐层设有 4 个结点。

该网络的适配值取为

$$f = \frac{1}{\int_0^1 (\tilde{\theta}^2 + \dot{\tilde{\theta}}^2) dt}$$

期望的运动轨迹取为

$$\theta_d(t) = 6t^5 - 15t^4 + 10t^3 - \pi/2 \quad t \in [0,1]$$

它满足初始及终了速度均为 0 的约束条件。

同样利用上面的 MGA 来训练该神经网络,所有的参数设置均与前面倒立摆系统的例子相同。图 7.15 显示了所设计的神经网络控制器的仿真结果,其中所有误差的初始条件均取为 0。从图中可见,该神经网络控制器具有满意的控制性能。

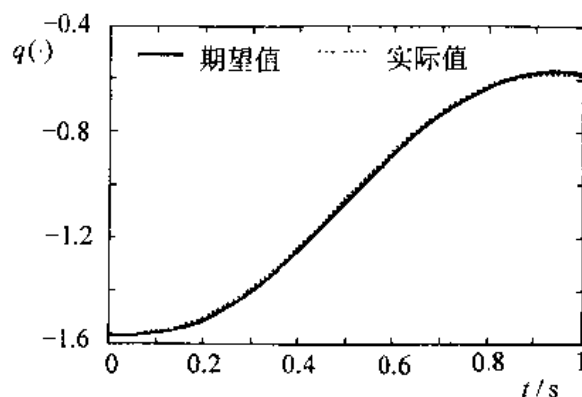


图 7.15 单臂机械手系统的仿真结果

在用遗传算法训练多层前馈神经网络时,有时会碰到结构冗余性问题,即存在许多功能等效而结构不同的网络。也就是说,对于一组最优的连接权和阈值参数,它们可能以不同的次序出现在许多不同的网络结构中。理论上对于一个有 N 个隐层结点的多层前馈网,则对于一个特定的映射总共有 $2^N N!$ 个冗余网络。因此有可能出现这样的情况:两个性能良好的父母样本经交叉操作后,可能产生两个很差的子女样本。因此若能在每次进行交叉操作之前,先进行再排序,则可避免这样的情况发生,从而进一步改善遗传算法的性能。

采用遗传算法有可能获得全局的最优点,这是它的最大优点,但是由于受编码字长的

限制,它的分辨率往往不高。因此在使用它训练多层前馈神经网络时,可以将它与常规的 BP 学习算法相结合,利用遗传算法获得大致的全局最优点,再由 BP 学习算法对其进行精心的调整。

前面只是介绍了利用遗传算法来选择神经网络参数。同样可以利用遗传算法来帮助确定神经网络的拓扑结构,这里将不再对此作详细讨论。

7.4.3 用遗传算法进行路径规划

无碰撞路径规划问题就是寻找一条从起点到终点的能够避开障碍物的最短路径。在结构化的空间中可采用如下的一种算法:将路径考虑成一系列的路径点,用人工势场法进行规划,用网络结构并行实现。在实时性方面,这种算法具有很大的优势。然而,这种算法对于全局最优解的寻找却无能为力。因此可引入遗传算法来帮助寻找全局的最优解。以往也有一些方法引入遗传算法进行路径规划,这些方法是将规划空间离散化,然后进行编码,按照常规的遗传算法进行寻优。这种方法对于非常大的规划空间,要么以粗粒度离散化,使寻优结果不够精确;要么以细粒度离散化,但计算量大大增加。下面介绍的算法则是直接对连续的规划空间进行寻优,同时在将遗传算法引入并行路径规划算法的时候,考虑了不降低原算法的并行程度,并使算法的实现尽量简单,寻优的效率较高。

1. 基于网络结构的并行路径规划算法

本算法的基本思想是构造规划空间的势函数,然后利用一阶梯度寻优来求得极短路径。势函数由碰撞罚函数和路径距离函数两部分组成。规划空间中每个点的罚函数是点到障碍物的距离的函数,此函数可表示为

$$E_c = \frac{1}{1 + e^{-D/T}}$$

其中 D 表示规划空间中点到障碍物的距离, T 表示温度,是罚函数形状的一个重要参数。当温度高时,罚函数比较平缓;当温度低时,罚函数比较陡峭。根据这个特性,一般在开始的时候选择较高的温度以使路径尽快地避开障碍物,然后逐渐降低温度,使路径紧贴障碍物,从而使路径尽量短。将路径考虑成一系列路径点的连线,每个路径点的势函数将是由路径点所处点的函数加上与相邻路径点的距离函数所构成。即

$$E_R = E_A + \beta E_c$$

其中 E_A 表示距离函数, E_c 表示罚函数, β 是加权系数。每个路径点可以独立地寻找各自的最低势能位置。由于路径点之间的相互独立性,所以可以用一种网络结构来并行地实现,各个路径点寻优使用人工势场法。算法分为三层,最低层所求的是第 k 个路径点到第 i 个障碍物的第 j 条边的距离;中间层所求的是第 k 个路径点在第 i 个障碍物的罚函数场中的梯度值;最上层计算出每个路径点在规划空间中势能函数的梯度,这一层包括罚函数梯度和距离函数梯度两部分的计算。

经过上述三个层次的计算,可以得到各个路径点在势场中的梯度值,沿负梯度方向以一定的步长移动路径点,最终可得到一个躲避障碍物的局部极值解。

2. 用遗传算法进行路径规划

上述路径规划方法只能获得局部最优解,为此引入遗传算法以获得全局的最优解,下

面具体介绍如何应用遗传算法来解决路径规划中的局部极值问题。

(1) 初始路径集的产生。利用前面所介绍的基于网络结构的并行路径规划算法,由不同的初始路径点序列产生一系列的路径。初始路径点是这样选择的:第一条路径的初始路径点序列选择从起点到终点所连的直线上的均匀分布的点列作为初始路径点序列。其它路径的初始路径点序列选择规划空间内随机分布的路径点集作为初始路径点序列。用基于网络结构并行路径规划算法使初始路径点序列收敛为不同的避障路径。由此产生一组路径集。

令 PPP(Parallel Path Planning)代表前面所介绍的并行路径规划算法, r_i^1 表示第一代第 i 个初始路径点序列集,即

$$r_i^1 = \{(\bar{x}_1^1, \bar{y}_1^1), (\bar{x}_2^1, \bar{y}_2^1), \dots, (\bar{x}_m^1, \bar{y}_m^1)\}$$

R_i^1 表示第一代第 i 条避障路径,即

$$R_i^1 = \{(x_1^1, y_1^1), (x_{i2}^1, y_{i2}^1), \dots, (x_m^1, y_m^1)\}$$

从而有 r_i^1 和 R_i^1 的关系为

$$r_i^1 \xrightarrow{PPP} R_i^1 \quad i = 1, 2, \dots, N$$

令 $\phi^1 = \{R_1^1, R_2^1, \dots, R_N^1\}$ 表示第一代路径集。

(2) 计算路径集中每一条路径的长度

$$L_i^1 = \|R_i^1\| = \sum_{j=1}^{n-1} \sqrt{(x_{i(j+1)}^1 - x_{ij}^1)^2 + (y_{i(j+1)}^1 - y_{ij}^1)^2} \quad i = 1, 2, \dots, N$$

再计算这组路径中的最短路径,即

$$L_{\min}^1 = \min_{i=1}^N (L_i^1)$$

(3) 对初始路径点序列进行复制操作。具体步骤为:首先根据路径长度确定各条路径的适配值,这里取

$$f_i^1 = 1/L_i^1$$

其次根据适配值按概率

$$p_i^1 = f_i^1 / \sum_{j=1}^N f_j^1$$

复制出 $2m$ 条路径, $2m < N$ 。

(4) 对复制后的路径进行交叉重组操作。交叉重组操作是将交叉区中两个路径进行部分互换产生两个新的初始路径点序列,然后用势场法使新的序列收敛成避障路径并计算其长度。路径相匹配的方法是对交叉区中的路径根据路径间距离按概率相配对。具体方法为,首先在交叉区中任选一条路径,设为 $R_{i_0}^1$,求出它与交叉区中别的路径的距离

$$D_{i_0 i}^1 = \sum_{j=1}^n \sqrt{(x_{i_0 j}^1 - x_{ij}^1)^2 + (y_{i_0 j}^1 - y_{ij}^1)^2} \quad i = 1, 2, \dots, N$$

则第 i 条路径被选中的概率为

$$q_{i_0 i}^1 = D_{i_0 i}^1 / \sum_{j=1}^N D_{i_0 j}^1$$

可以看出,当两条路径的距离较远时,它们进行交叉重组的可能性便较大。这样做的目的

主要是为了扩大搜索空间,避免过早收敛。

在完成了一条路径的匹配操作后,对交叉区中的剩余路径进行与上面类似的匹配操作,最后得到 m 对路径。在完成了全部匹配过程后,随机产生 m 个小于 n (n 为路径中路径点的个数)的随机正整数,然后将这 m 对路径以这些随机正整数为交叉点进行交叉重组,这样就可以得到 $2m$ 个路径点序列 $r_k^2 (k=1,2,\dots,2m)$ 。对这 $2m$ 个路径点序列运用势场法,从而得到 $2m$ 个新路径 R_k^2 ,即

$$r_k^2 \xrightarrow{PPP} \bar{R}_k^2 \quad k=1,2,\dots,m$$

(5) 交叉重组之后获得 $2m$ 条新的路径,再将上一代中路径较短的 $N-2m$ 条路径直接传至下一代,即获得新一代 N 条路径集。具体步骤为:设 $R_j^1 (j=1,2,\dots,N-2m)$ 为 $\phi^1 = \{R_1^1, R_2^1, \dots, R_N^1\}$ 中较短的 $N-2m$ 条路径,则

$$\phi^2 = \{R_1^2, R_2^2, \dots, R_N^2\} = \{R_1^1, R_2^1, \dots, R_{N-2m}^1, \bar{R}_1^2, \bar{R}_2^2, \dots, \bar{R}_{2m}^2\}$$

对新一代的路径集重复(2)~(5)的操作,直至最短路径符合要求或连续几代最短路径不再改变。

为了更好地寻找全局最短路径,在算法中也可小概率引入“变异”和“移民”。在本算法中,变异的实现是将路径点序列中的某几个点移动至规划空间内随机的位置上,然后用人工势场法来进行规划。“变异”的结果,可能仍收敛到原来的路径上,也可能成功地产生新的路径。在本算法中,“移民”就是将随机产生的路径点序列作为初始路径点序列,用人工势场法产生的路径加入到路径集中进行复制和交叉重组。

3. 仿真结果

下面给出初始路径集中路径的数目为 6 时的仿真结果。

初始路径点序列分两种情况:其中一条为起点到终点所连直线上均匀分布的点列,其余的路径点序列是随机分布的。6 个路径点序列设为: $r_k^1 (k=1,2,\dots,6)$ 。由初始路径点序列可收敛成初始路径,设为 $R_k^1 (k=1,2,\dots,6)$ 。图 7.16 和图 7.17 表示了其中的 R_2^1 和 R_4^1 两条路径。

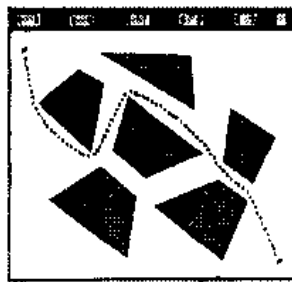


图 7.16 初始路径 R_2^1

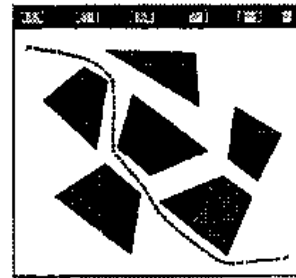


图 7.17 初始路径 R_4^1

至此,求得了第一代路径集: $\phi^1 = \{R_1^1, R_2^1, R_3^1, R_4^1, R_5^1, R_6^1\}$ 。计算各条路径长度,其结果如表 7.5 所示。进而求得 $L_{\min}^1 = 739.66$ 。然后选择匹配,这里 R_2^1 与 R_4^1 , R_1^1 和 R_5^1 相匹配,对它们进行交叉重组后得路径点序列 r_1^2, r_2^2, r_3^2 和 r_4^2 。图 7.18 和图 7.19 表示了其中的两条

路径: r_1^2 和 r_2^2 。该路径点序列运用势场法得到 4 个新路径, 即 $r_k^2 \xrightarrow{PPP} \bar{R}_k^2, k=1,2,3,4$ 。图 7. 20 和图 7. 21 表示了其中的两条: \bar{R}_1^2 和 \bar{R}_2^2 。

表 7. 5 第一代路径集及其交叉重组后的各路径长度

R_1^1	R_2^1	R_3^1	R_4^1	R_5^1	R_6^1	\bar{R}_1^2	\bar{R}_2^2	\bar{R}_3^2	\bar{R}_4^2
795. 28	851. 44	739. 66	773. 34	911. 51	761. 70	703. 26	738. 89	738. 27	703. 31

至此, 可以得到第二代路径集

$$\psi^2 = \{R_1^2, R_2^2, R_3^2, R_4^2, R_5^2, R_6^2\} = \{R_3^1, R_6^1, \bar{R}_1^2, \bar{R}_2^2, \bar{R}_3^2, \bar{R}_4^2\}$$

并由表 7. 5 可得 $I_{min}^2=703. 26$

将 ψ^1 和 ψ^2 相比较可得, 其最短长度由 739. 66 变为 703. 26, 平均长度由 805. 49 变为 730. 85, 这正是优胜劣汰机制的引入所导制的结果。由复制和交叉重组不断地产生新路径, 由于在该算法中直接保留了上一代的最短路径, 因而确保了最短路径的长度是一个非增的序列。“变异”和“移民”的产生概率较小, 这里就不列举了。

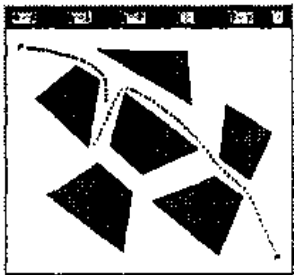


图 7. 18 R_3^1 与 R_4^1 交叉重组后得到的新路径点序列 r_1^2

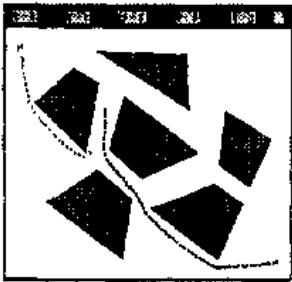


图 7. 19 R_3^1 和 R_2^1 交叉重组后得到的新路径点序列 r_2^2

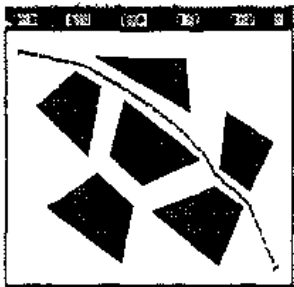


图 7. 20 r_1^2 经用势场法后得到的新路径 R_1^2

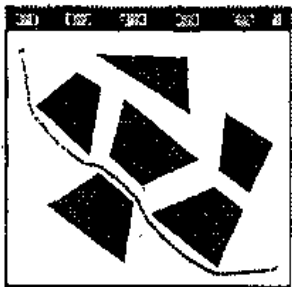


图 7. 21 r_2^2 经用势场法后得到的新路径 R_2^2

以上是进行一次交叉重组后的情况。对于该例实际上已找到了最短路径。一般情况下, 也许要进行多次交叉重组, 如果连续几次的结果没有改变, 则认为寻优结束; 否则, 每经过一次迭代, 其结果都有所改进, 因而也能很快找到最优解。

参 考 文 献

- [1] Goldberg D E. Genetic Algorithms on Search, Optimization and Machine Learning. Addison-Wesley, 1989
- [2] 万骞. 遗传算法的研究. 清华大学计算机系学士论文, 1995
- [3] Sun Zengqi, Wan Qian. A Modified Genetic Algorithm; Meta-Level Control of Migration in a Distributed GA. Proc. of IEEE International Conference on Evolutionary Computation. 1995
- [4] Karr C. Applying Genetics to Fuzzy Logic. AI Expert, 1991, 3
- [5] Jenog I K *et al.* A Modified Genetic Algorithm for Neurocontrollers. Proc. of IEEE International Conference on Evolutionary Computation, 1995
- [6] 吴晓涛, 孙增圻. 用遗传算法进行路径规划. 清华大学学报, 1995, 35(5)
- [7] 孙增圻, 万骞. 遗传算法及其在智能控制中的应用. 第二届全国智能控制专家讨论会, 北京, 1994

参 考 文 献

- [1] Goldberg D E. Genetic Algorithms on Search, Optimization and Machine Learning. Addison-Wesley, 1989
- [2] 万骞. 遗传算法的研究. 清华大学计算机系学士论文, 1995
- [3] Sun Zengqi, Wan Qian. A Modified Genetic Algorithm; Meta-Level Control of Migration in a Distributed GA. Proc. of IEEE International Conference on Evolutionary Computation. 1995
- [4] Karr C. Applying Genetics to Fuzzy Logic. AI Expert, 1991, 3
- [5] Jenog I K *et al.* A Modified Genetic Algorithm for Neurocontrollers. Proc. of IEEE International Conference on Evolutionary Computation, 1995
- [6] 吴晓涛, 孙增圻. 用遗传算法进行路径规划. 清华大学学报, 1995, 35(5)
- [7] 孙增圻, 万骞. 遗传算法及其在智能控制中的应用. 第二届全国智能控制专家讨论会, 北京, 1994

参 考 文 献

- [1] Goldberg D E. Genetic Algorithms on Search, Optimization and Machine Learning. Addison-Wesley, 1989
- [2] 万骞. 遗传算法的研究. 清华大学计算机系学士论文, 1995
- [3] Sun Zengqi, Wan Qian. A Modified Genetic Algorithm; Meta-Level Control of Migration in a Distributed GA. Proc. of IEEE International Conference on Evolutionary Computation. 1995
- [4] Karr C. Applying Genetics to Fuzzy Logic. AI Expert, 1991, 3
- [5] Jenog I K *et al.* A Modified Genetic Algorithm for Neurocontrollers. Proc. of IEEE International Conference on Evolutionary Computation, 1995
- [6] 吴晓涛, 孙增圻. 用遗传算法进行路径规划. 清华大学学报, 1995, 35(5)
- [7] 孙增圻, 万骞. 遗传算法及其在智能控制中的应用. 第二届全国智能控制专家讨论会, 北京, 1994