# 神经网络设计

# Neural Network Design

Martin T. Hagan
(美) Howard B. Demuth 著
Mark H. Beale

戴 葵 等译
李伯民 审校

机械工业出版社
China Machine Press

HZ BOOKS
华章教育

国外经典教材

Classical Texts From Top Universities

# 前　言

本书介绍了神经网络的基本结构和学习规则，重点是对这些神经网络的数学分析、训练方法和神经网络在模式识别、信号处理以及控制系统等工程实践问题中的应用。

本书尽力用清晰和一致的方式来组织材料，以易于阅读和使用。对每个讨论的主题，使用大量例题来阐明。

由于这是一本关于神经网络设计的书，因此在选择主题时我们依据了两个原则：首先，尽量采用最实用的神经网络结构、学习规则和训练方法；其次，尽量保证该书的完整性，使读者从一章到下一章的学习感觉流畅。为此，在特定主题前，都会有一些相关的介绍性材料和应用数学基础的章节。总之，在我们选择的主题中，某些部分在神经网络实际应用中极其重要，而另一些部分对解释神经网络如何运算十分有用。

书中省去了很多本来可以收入的主题。比如，我们并没有把书写成有关所有已知神经网络结构和学习规则的分类和纲要，而是集中介绍一些基本概念。其次，我们没有讨论神经网络的实现技术，比如说 VLSI 实现、光学器件实现和并行计算机实现等。另外，我们也没有深入阐述神经网络的生物学和心理学基础。上述内容虽然重要，但本书并不包含这些内容，因为我们希望能集中力量把我们认为在神经网络设计中对读者最重要的主题阐述清楚。

本书是为高年级本科生或一年级研究生编写的半学期导论性课程教材（也适于作短期教程、自学或参考用书）。希望读者有一定的线性代数、概率论和微分方程的基础知识。

本书每一章都分为以下各节：目的、理论和实例、小结、例题、结束语、参考文献和习题。理论和实例部分是各章的主体部分，包括基本思想的发展和实例。小结部分列出了一些重要的公式和概念，以利于将本书作为实际工作的参考。每章大约三分之一的篇幅是例题部分，这一部分给出了所有关键概念的详细例题。

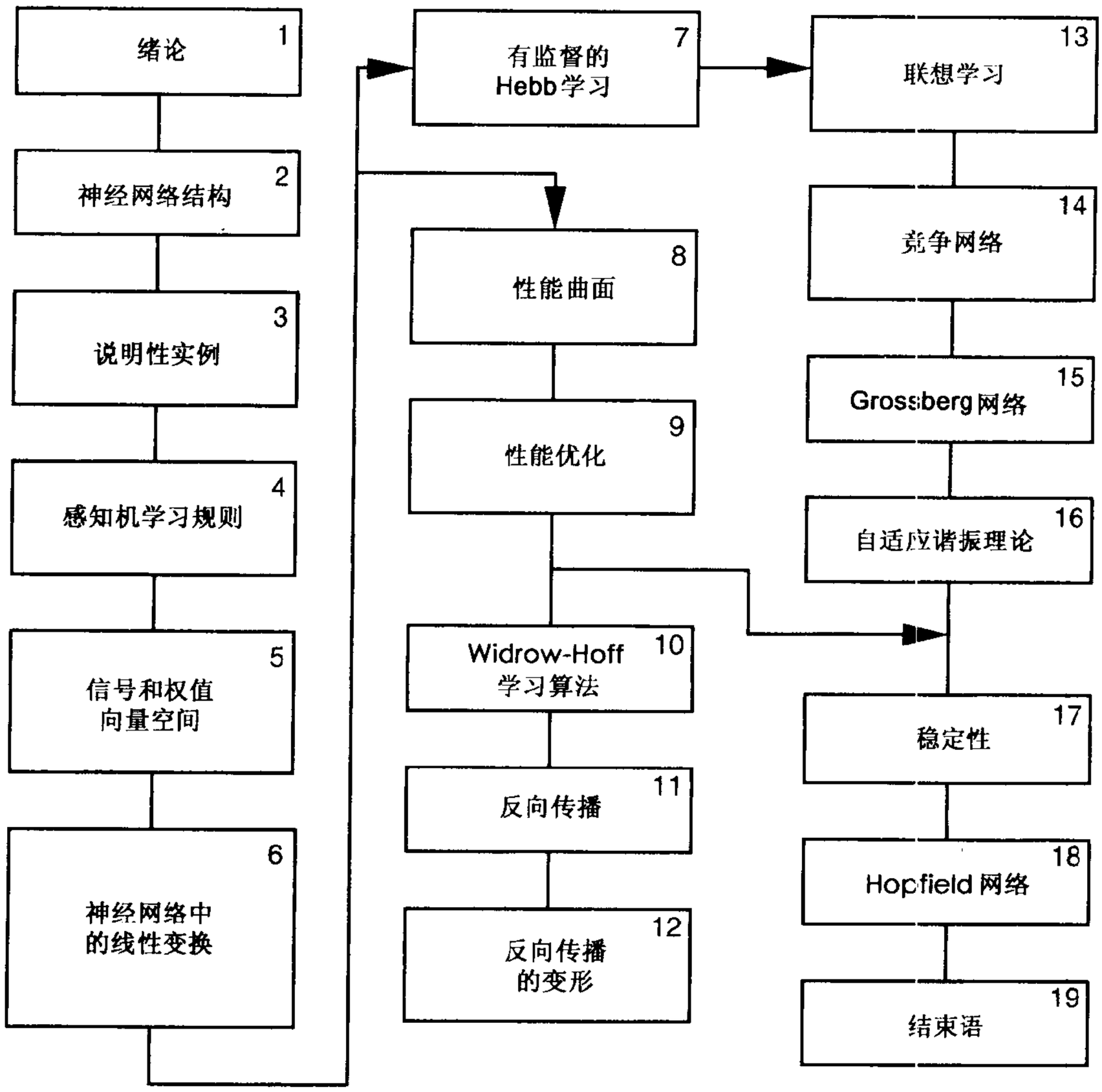
后面的图说明了各章之间的相互关系。

第 1 章到第 6 章覆盖了其余各章所需要的基本概念。第 1 章是绪论部分，简单介绍历史背景和一些基本生物学知识。第 2 章介绍基本的神经网络结构。全书都使用这一章给出的标记方法。第 3 章描述一个简单的模式识别问题，并说明怎样用三种不同类型的神经网络可以求解。这三种网络是本书所述其余神经网络类型的代表。另外，所引入的模式识别问题也为全书提供了一个实验的主线。

本书的许多重点是在使神经网络能完成各种任务的训练方法上。第 4 章介绍学习算法，并给出第一个实用算法（感知机学习规则）。虽然感知机网络存在一些基本的局限性，但它有其自身的历史重要性，并且对于导入某些关键概念也是十分有用的工具，这些概念将要用于其后各章讨论功能更加强大的网络中。

本书的主要目的是解释神经网络的基本工作原理。为此，我们将结合神经网络这一主题给出其他的一些介绍性材料。比如，第 5 和第 6 章要复习线性代数，它是理解神经网络的数学基础的核心。这两章讨论的概念在其余各章被广泛地用到。

第 7 章和第 13 章到第 16 章介绍主要由生物学和心理学的启示所得到的神经网络和学习




规则。它们主要分为两类：相联网络和竞争网络。第 7 章和第 13 章介绍基本概念，第 14 章到第 16 章论述更先进的网络。

第 8 章到第 12 章提出一类叫性能学习（performance learning）的学习方法，用它训练网络以优化网络的性能。第 8 章和第 9 章介绍性能学习的基本概念。第 10 章到第 12 章将这些概念用于前馈神经网络中，这将增强网络的能力，但同时也会增加学习的复杂性。

第 17 章和第 18 章讨论递归网络，这些含有反馈连接的网络是一种动态系统。第 17 章研究这些系统的稳定性；第 18 章描述 Hopfield 网络，它是目前最有影响的递归网络之一。

在第 19 章，我们对本书所给出的各种网络进行小结，并讨论它们同本书没有涉及的其他网络之间的关系。同时，我们也要为读者指明进一步研究的一些其他参考资料。如果您想知道“我从此将走向何处?”，请看第 19 章。

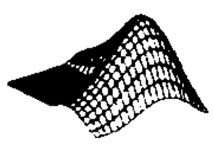
## 软件

MATLAB 软件包不是使用本书必需的。上机练习可以用其他任何编程语言实现，而且 *Neural Network Design Demonstration*（神经网络演示）虽然对读者有帮助，但也是理解本

书涉及材料的关键所在。

然而，我们还是把 MATLAB 软件包作为这本教科书的附件。由于该软件包含矩阵/向量的表示法和图形显示，所以它也为神经网络的实验提供了一种方便的环境。我们以两种不同的方法使用 MATLAB。第一，书中含有大量需要读者用 MATLAB 完成的习题。神经网络许多重要特征只有在解决大规模问题时才能体现出来，这些密集的计算不适于手算。用 MATLAB 能很快实现神经网络算法，并对大量问题方便地进行测试。(若没有 MATLAB，用其他语言也可以完成这些练习。)

使用 MATLAB 的第二种方法是用本书所附磁盘中的 *Neural Network Design Demonstrations*（神经网络设计演示），这些交互式演示阐述了每章的重要概念。使用时，把软件拷到 MATLAB 目录中，在 MATLAB 的提示符下，敲击 nnd 即可激活。所有演示都可以在主菜单下访问。



用左边所示的图标，指明正文中对这些演示的引用。演示需要 MATLAB 4.0 或更高版本，或者 MATLAB 4.0 学生版。另外，一些演示需要用 MathWorks 公司的 *Neural Network Toolbox*（神经网络工具箱）1.0 版本或更高版本。如何使用演示软件请参阅附录 C。

为了帮助使用本书的教师，我们还准备了投影胶片和习题答案。每一章的投影胶片（用微软的 Powerpoint 格式）可以从网址 www.pws.com/pwsftp.html 获得。也可以获取习题答案。

## 致谢

我们深切地感谢那些阅读了本书全部或部分稿件的人以及对软件的各种不同版本进行测试的人，特别要感谢的是 Canterbury 大学的 John Andreae 教授，AT&T 的 Dan Foresee，Oklahoma 州立大学的 Carl Latino 博士，MCI 的 Jack Hagan，SRI 的 Gerry Andeen 博士，以及 Idaho 大学的 Joan Miller 和 Margie Jenks。我们从 Oklahoma 州立大学的 ECEN 5713 班及 Canterbury 大学的 ENEL 621 班的研究生们那里得到了许多建设性的意见，他们阅读了最早的初稿，并测试了软件，对本书的改进提供了极有帮助的建议。另外，我还要感谢那些虽然没有留下姓名，但对该书提出了许多有益建议的人。

我们竭诚感谢 Peter Gough 博士，感谢他邀请我们加入 Canterbury 大学的电气和电子工程系，感谢 Mike Surety 在计算机方面的帮助，以及全系同仁的大力支持。在 Oklahoma 州立大学的学术假和离开 Idaho 大学的这一年时间里，我们完成了这本书。感谢 TI 公司特别是 Bill Harland 对我们在神经网络研究上的支持。感谢 MathWorks 公司允许我们使用 *Neural Network Toolbox*（神经网络工具箱）中的材料。

我们非常感谢 Joan Pilgram 的鼓励和她在商业方面的建议，感谢 Bernice Hewitt 夫人的热情支持。

最后，我们由衷地表达对 PWS 出版公司的同仁们的感激，特别是 Bill Barter，Pam Rockwell，Amy Mayfield，Ken Morton 和 Nathan Wilbur。感谢 Vanessa Pineiro 为本书设计的非常具有艺术品位的封面。

# 目　录


出版者的话
专家委员会
译者序
前言
第 1 章　绪论 …… 1
1.1　目的 …… 1
1.2　历史 …… 1
1.3　应用 …… 3
1.4　生物学的启示 …… 5
参考文献 …… 6
第 2 章　神经元模型和网络结构 …… 8
2.1　目的 …… 8
2.1　理论和实例 …… 8
2.2.1　符号 …… 8
2.2.2　神经元模型 …… 8
2.2.3　网络结构 …… 13
2.3　小结 …… 17
2.4　例题 …… 20
2.5　结束语 …… 22
习题 …… 22
第 3 章　一个说明性实例 …… 23
3.1　目的 …… 23
3.2　理论和实例 …… 23
3.2.1　问题描述 …… 23
3.2.2　感知机 …… 24
3.2.3　Hamming 网络 …… 28
3.2.4　Hopfield 网络 …… 30
3.3　结束语 …… 32
习题 …… 32
第 4 章　感知机学习规则 …… 34
4.1　目的 …… 34
4.2　理论和实例 …… 34
4.2.1　学习规则 …… 35
4.2.2　感知机的结构 …… 35
4.2.3　感知机学习规则 …… 39
4.2.4　收敛性证明 …… 44
4.3　小结 …… 47
4.4　例题 …… 47
4.5　结束语 …… 56
参考文献 …… 56
习题 …… 57
第 5 章　信号和权值向量空间 …… 60
5.1　目的 …… 60
5.2　理论和实例 …… 60
5.2.1　线性向量空间 …… 60
5.2.2　线性无关 …… 62
5.2.3　生成空间 …… 62
5.2.4　内积 …… 63
5.2.5　范数 …… 63
5.2.6　正交性 …… 64
5.2.7　向量展开式 …… 65
5.3　小结 …… 68
5.4　例题 …… 70
5.5　结束语 …… 76
参考文献 …… 76
习题 …… 76
第 6 章　神经网络中的线性变换 …… 79
6.1　目的 …… 79
6.2　理论和实例 …… 79
6.2.1　线性变换 …… 79
6.2.2　矩阵表示 …… 80
6.2.3　基变换 …… 82
6.2.4　特征值和特征向量 …… 85
6.3　小结 …… 88
6.4　例题 …… 89
6.5　结束语 …… 96
参考文献 …… 96
习题 …… 97
第 7 章　有监督的 Hebb 学习 …… 99

7.1 目的 …… 99
7.2 理论和实例 …… 99
7.2.1 线性联想器 …… 100
7.2.2 Hebb 规则 …… 100
7.2.3 仿逆规则 …… 103
7.2.4 应用 …… 104
7.2.5 Hebb 学习的变形 …… 106
7.3 小结 …… 106
7.4 例题 …… 107
7.5 结束语 …… 116
参考文献 …… 116
习题 …… 117
第 8 章 性能曲面和最优点 …… 119
8.1 目的 …… 119
8.2 理论和实例 …… 119
8.2.1 泰勒级数 …… 119
8.2.2 方向导数 …… 121
8.2.3 极小点 …… 122
8.2.4 优化的必要条件 …… 124
8.2.5 二次函数 …… 126
8.3 小结 …… 131
8.4 例题 …… 132
8.5 结束语 …… 141
参考文献 …… 141
习题 …… 141
第 9 章 性能优化 …… 143
9.1 目的 …… 143
9.2 理论和实例 …… 143
9.2.1 最速下降法 …… 143
9.2.2 牛顿法 …… 148
9.2.3 共轭梯度法 …… 152
9.3 小结 …… 155
9.4 例题 …… 156
9.5 结束语 …… 165
参考文献 …… 166
习题 …… 166
第 10 章 Widrow - Hoff 学习算法 …… 168
10.1 目的 …… 168
10.2 理论和实例 …… 168
10.2.1 ADALINE 网络 …… 168
10.2.2 均方误差 …… 170
10.2.3 LMS 算法 …… 171
10.2.4 收敛性分析 …… 173
10.2.5 自适应滤波 …… 175
10.3 小结 …… 181
10.4 例题 …… 182
10.5 结束语 …… 193
参考文献 …… 193
习题 …… 194
第 11 章 反向传播 …… 197
11.1 目的 …… 197
11.2 理论和实例 …… 197
11.2.1 多层感知机 …… 197
11.2.2 反向传播算法 …… 201
11.2.3 例子 …… 205
11.2.4 反向传播 …… 207
11.3 小结 …… 211
11.4 例题 …… 212
11.5 结束语 …… 221
参考文献 …… 222
习题 …… 222
第 12 章 反向传播算法的变形 …… 227
12.1 目的 …… 227
12.2 理论和实例 …… 227
12.2.1 BP 算法的缺点 …… 227
12.2.2 BP 算法的启发式改进 …… 232
12.2.3 数值优化技术 …… 235
12.3 小结 …… 244
12.4 例题 …… 246
12.5 结束语 …… 255
参考文献 …… 255
习题 …… 257
第 13 章 联想学习 …… 259
13.1 目的 …… 259
13.2 理论和实例 …… 259
13.2.1 简单联想网络 …… 260
13.2.2 无监督的 Hebb 规则 …… 261
13.2.3 简单的识别网络 …… 264
13.2.4 instar 规则 …… 265
13.2.5 简单回忆网络 …… 268
13.2.6 outstar 规则 …… 268
13.3 小结 …… 271
13.4 例题 …… 272
13.5 结束语 …… 279

参考文献…… 280
习题…… 280
第 14 章 竞争网络 …… 285
14.1 目的…… 285
14.2 理论和实例…… 285
14.2.1 Hamming 网络 …… 286
14.2.2 竞争层…… 287
14.2.3 生物学意义上的竞争层…… 291
14.2.4 自组织特征图…… 292
14.2.5 学习向量量化…… 295
14.3 小结…… 299
14.4 例题…… 301
14.5 结束语…… 310
参考文献…… 310
习题…… 311
第 15 章 Grossberg 网络 …… 315
15.1 目的…… 315
15.2 理论和实例…… 315
15.2.1 生物学的启发：视觉…… 316
15.2.2 基本非线性模型…… 321
15.2.3 两层竞争网络…… 323
15.2.4 与 Kohonen 规则的关系…… 331
15.3 小结…… 332
15.4 例题…… 335
15.5 结束语…… 342
参考文献…… 343
习题…… 344
第 16 章 自适应谐振理论…… 346
16.1 目的…… 346
16.2 理论和实例…… 346
16.2.1 自适应谐振概述…… 346
16.2.2 第一层…… 347
16.2.3 第二层…… 351
16.2.4 调整子系统…… 354
16.2.5 学习规则：L1 - L2 …… 356
16.2.6 学习规则：L2 - L1 …… 358
16.2.7 ART1 算法小结 …… 359
16.2.8 其他 ART 体系结构 …… 360
16.3 小结…… 361
16.4 例题…… 364
16.5 结束语…… 374
参考文献…… 375
习题…… 375
第 17 章 稳定性 …… 378
17.1 目的…… 378
17.2 理论和实例…… 378
17.2.1 递归网络…… 378
17.2.2 稳定性概念…… 379
17.2.3 Lyapunov 稳定性定理 …… 381
17.2.4 单摆例子…… 381
17.2.5 LaSalle 不变性定理 …… 385
17.3 小结…… 390
17.4 例题…… 391
17.5 结束语…… 396
参考文献…… 396
习题…… 397
第 18 章 Hopfield 网络 …… 399
18.1 目的…… 399
18.2 理论和实例…… 399
18.2.1 Hopfield 模型 …… 400
18.2.2 Lyapunov 函数 …… 401
18.2.3 增益效应…… 406
18.2.4 Hopfield 网络设计 …… 409
18.3 小结…… 413
18.4 例题…… 415
18.5 结束语…… 421
参考文献…… 422
习题…… 423
第 19 章 结束语 …… 426
19.1 目的…… 426
19.2 理论和实例 …… 426
19.2.1 前馈和联想网络…… 426
19.2.2 竞争网络…… 429
19.2.3 动态联想存储器网络…… 429
19.2.4 神经网络的经典基础…… 430
19.2.5 参考书目和杂志 …… 431
19.3 结束语…… 432
参考文献…… 432
附录 A 文献目录 …… 439
附录 B 符号 …… 447
附录 C 软件 …… 452
索引 …… 456

# 第1章 绪 论

## 1.1 目的

当你现在看这本书的时候，就正在使用一个复杂的生物神经网络。你有一个约为 $10^{11}$ 个神经元的高度互连的集合帮助你完成阅读、呼吸、运动和思考。你的每一个生物神经元都是生物组织和化学物质的有机结合。若不考虑其速度的话，可以说每个神经元都是一个复杂的微处理器。你的某些神经结构是与生俱来的，而其他一些则是在实践中形成的。

科学家们才刚刚开始对生物神经网络工作机理有所认识。一般认为，包括记忆在内的所有生物神经功能，都存储在神经元和及其之间的连接上。学习被看作是在神经元之间建立新的连接或对已有的连接进行修改的过程。这便将引出下面一个问题：既然我们已经对生物神经网络有一个基本的认识，那么能否利用一些简单的人工“神经元”构造一个小系统，然后对其进行训练，从而使它们具有一定有用功能呢？回答是肯定的。本书正是要讨论有关人工神经网络工作机理的一些问题。

我们在这里考虑的神经元不是生物神经元。它们是对生物神经元极其简单的抽象，可以用程序或硅电路实现。虽然由这些神经元组成的网络的能力远远不及人脑的那么强大，但是可对其进行训练，以实现一些有用的功能。本书所要介绍的正是有关于这样的神经元，以及包含这些神经元的网络及其训练方法。 1-1

## 1.2 历史

在人工神经网络的发展历程中，涌现了许多在不同领域中富有创造性的传奇人物，他们艰苦奋斗几十年，提出了许多至今仍然让我们受益的概念。许多作者都记载了这一历史。一本特别有趣的书是由 John Anderson 和 Edward Rosenfeld 撰写的《神经计算：研究的基础》(*Neurocomputing: Foundations of Research*)。在该书中，他们收集并编辑了一组由 43 篇具有特别历史意义的论文，每一篇前面都有一段历史观点的导言。

本书各章开始包括了一些主要神经网络研究人员的历史，所以这里不必赘述。但是，还是有必要简单地回顾一下神经网络的主要发展历史。

对技术进步而言，有两点是必需的：概念与实现。首先，必须有一个思考问题的概念，根据这些概念明确所面临的问题。这就要求概念包含一种简单的思想，或者更具特色，并且引入数学描述。为了理解这一点，让我们看看心脏的研究历史。在不同时期，心脏被看成灵魂的中心或身体的热源。17 世纪的医生们认识到心脏是一个血泵，于是科学家们开始设计实验，研究泵的行为。这些实验最终开创了循环系统理论。可以说，没有泵的概念，就不会有人们对心脏的深入认识。

概念及其相应的数学描述还不足以使新技术走向成熟，除非能通过某种方式实现这种系统。比如，虽然多年前就从数学上知道根据计算机辅助层析成像(CAT)扫描可以重构图像，但是直到有了高速计算机和有效的算法才使其走向实用，并最终实现了有用的 CAT 系统。

神经网络的发展史同时包含了概念创新和实现开发的进步。但是这些成果的取得并不是一帆风顺的。

神经网络领域研究的背景工作始于 19 世纪末和 20 世纪初。它源于物理学、心理学和神经生理学的跨学科研究，主要代表人物有 Herman Von Helmholts，Ernst Mach 和 Ivan Pavlov。这些早期研究主要还是着重于有关学习、视觉和条件反射等一般理论，并没有包含有关神经元工作的数学模型。

1-2

现代对神经网络的研究可以追溯到 20 世纪 40 年代 Warren McCulloch 和 Walter Pitts 的工作[McPi43]。他们从原理上证明了人工神经网络可以计算任何算术和逻辑函数。通常认为他们的工作是神经网络领域研究工作的开始。

在 McCulloch 和 Pitts 之后，Donald Hebb [Hebb49]指出，经典的条件反射(由 Pavlov 发现)是由单个神经元的性质引起的。他提出了生物神经元的一种学习机制(参见第 7 章)。

人工神经网络第一个实际应用出现在 20 世纪 50 年代后期，Frank Rosenblatt [Rose58]提出了感知机网络和联想学习规则。Rosenblatt 和他的同事构造了一个感知机网络，并公开演示了它进行模式识别的能力。这次早期的成功引起了许多人对神经网络研究的兴趣。不幸的是，后来研究表明基本的感知机网络只能解决有限的几类问题。(有关 Rosenblatt 和感知机学习规则，请参见第 4 章。)

同时，Bernard Widrow 和 Ted Hoff [WiHo60]引入了一个新的学习算法用于训练自适应线性神经网络。它在结构和功能上类似于 Rosenblatt 的感知机。Widrow-Hoff 学习规则至今仍然还在使用。(关于 Widrow-Hoff 学习请参见第 10 章。)

但是，Rosenblatt 和 Widrow 的网络都有同样的固有局限性。这些局限性在 Marvin Minsky 和 Symour Papert 的书[MiPa69]中有广泛的论述。Rosenblatt 和 Widrow 也十分清楚这些局限性，并提出了一些新的网络来克服这些局限性。但是他们没能成功找到训练更加复杂网络的学习算法。

许多人受到 Minsky 和 Papert 的影响，相信神经网络的研究已走入了死胡同。同时由于当时没有功能强大的数字计算机来支持各种实验，从而导致许多研究者纷纷离开这一研究领域。神经网络的研究就这样停滞了十多年。

即使如此，在 20 世纪 70 年代，科学家们仍然在该领域开展了许多重要的工作。1972 年 Teuvo Kohonen [Koho72]和 James Anderson [Ande72]分别独立提出了能够完成记忆的新型神经网络。(有关 Kohonen 网络更加详细的内容请参见第 13 章和第 14 章。)这一时期，Stephen Grossberg [Gros76]在自组织网络方面的研究也十分活跃。(参见第 15 章和第 16 章。)

前面我们说过，在 60 年代，由于缺乏新思想和用于实验的高性能计算机，曾一度动摇了人们对神经网络的研究兴趣。到了 80 年代，随着个人计算机和工作站计算能力的急剧增强和广泛应用，以及不断引入新的概念，克服了摆在神经网络研究面前的障碍，人们对神经网络的研究热情空前高涨。

1-3

有两个新概念对神经网络的复兴具有极其重大的意义。其一是：用统计机理解释某些类型的递归网络的操作，这类网络可作为联想存储器。物理学家 John Hopfield 的研究论文[Hopf82]论述了这些思想。(第 17 章和第 18 章讨论 Hopfield 网络。)

其二是：在 20 世纪 80 年代，几个不同的研究者分别开发出了用于训练多层感知机的反

传算法。其中最具影响力的反传算法是 David Rumelhart 和 James McClelland [RuMc86]提出的。该算法有力地回答了 60 年代 Minsky 和 Papert 对神经网络的责难。(有关反传算法详细内容请参见第 11 章和第 12 章。)

这些新进展对神经网络研究领域重新注入了活力。在过去的 10 年中，人们发表了成千上万的神经网络研究论文，神经网络也有了很多应用。许多理论和实践工作蜂拥而至，以致于我们至今还不十分清楚这将会把我们带向何方。

以上简略的历史回顾并没有列出所有对神经网络作出重要贡献的人，但它能使读者知道神经网络是如何发展而来的。读者或许会注意到，这个发展趋势并不总是“缓慢而坚定”的，而是曾经有急剧发展的时期，也有相对停滞的时期。

许多神经网络研究进展都与新概念的提出有关，如革新的神经网络结构和训练规则。同样十分重要的是，高性能计算机的出现使新概念能够得到检验。

好了，对神经网络的历史就说这么多。真正的问题是：“以后的 10 到 20 年会怎样?”神经网络将演变为一个永久的数学/工程工具，还是像许多曾大有希望的技术那样退出历史舞台？目前来看，似乎神经网络不仅有兴旺的时日，而且能取得一个永久的地位，即使它不能解决所有问题，但在某些适当的场合还是非常有用的工具。另外，要记住我们现在对人脑的认识仍很肤浅，相信将来某一天神经网络将会取得最重要的进展。

尽管很难预料神经网络今后能否成功，但这种新技术的大量而广泛应用还是令人鼓舞的。下面一节将介绍一些神经网络应用。 1-4

## 1.3 应用

最近报纸报道 Aston 大学用神经网络来进行文献研究。这篇报道说“神经网络可以用来识别个人的写作风格,研究人员用它比较了莎士比亚和他同时代人的著作”。一个大众科学电视节目最近报道了某意大利的研究结构用神经网络测试橄榄油的纯度。这些例子从一个侧面说明神经网络有极其广泛的应用领域。正是因为它适合于解决实际问题，所以其应用领域在不断扩大，它不仅可以广泛应用于工程、科学和数学领域，也可广泛应用于医学、商业、金融和文学等领域。神经网络在许多领域的广泛应用，使其极具吸引力。同时，基于高速计算机和快速算法，也可以用神经网络解决过去许多计算量很大的复杂工业问题。

以下神经网络的应用说明来源于 MATLAB 用到的 *Neural Network Toolbox* (神经网络工具箱)，已经得到了 MathWorks 公司的允许。

1988 年，在 DARPA 的“神经网络研究报告”(Neural Network Study)[DARP88]中列举了各种神经网络的应用。其中第一个应用就是大约在 1984 年的自适应频道均衡器。这个设备在商业上取得了极大的成功。它用一个单神经元网络来稳定电话系统中长距离传输的声音信号。DARPA 报告还列举了其他一些神经网络在商业领域中的应用，包括一个小规模的单词识别器、过程监测器、声纳分类器和一个风险分析系统。

自 DARPA 报告问世以来，神经网络已被用于许多领域。在文献中所列举的一些应用如下：

**航空**

高性能飞行器自动驾驶仪，飞行路径模拟，飞机控制系统，自动驾驶优化器，飞行部件模拟，飞行器部件故障检测器

**汽车**

汽车自动导航系统，担保行为分析器

**银行**

1-5 支票和其他公文阅读器，信贷申请的评估器

**国防**

武器操纵，目标跟踪，目标辨识，面部识别、新型的传感器，声纳、雷达和图像信号处理(包括数据压缩、特征提取、噪声抑制、信号/图像的识别)

**电子**

代码序列预测，集成电路芯片布局，过程控制，芯片故障分析，机器视觉，语音综合，非线性建模

**娱乐**

动画，特技，市场预测

**金融**

不动产评估，借贷咨询，抵押审查，公司证券分级，投资交易程序，公司财务分析，通货价格预测

**保险**

政策应用评估，产品优化

**制造**

生产流程控制，产品设计和分析，过程和机器诊断，实时微粒识别，可视质量监督系统，啤酒检测，焊接质量分析，纸张质量预测，计算机芯片质量分析，磨床运转分析，化工产品设计分析，机器性能分析，项目投标，计划和管理，化工流程系统动态建模

**医疗**

乳房癌细胞分析，EEG 和 ECG 分析，修复设计，移植次数优化，医院费用节流，医院质量改进，急诊室检查建议

**石油和天然气**

1-6 探查

**机器人**

轨道控制，铲车机器人，操作手控制器，视觉系统

**语音**

语音识别，语音压缩，元音识别，文本到语音的综合

**有价证券**

市场分析，自动证券分级，股票交易咨询系统

**电信**

图像和数据压缩，自动信息服务，实时语言翻译，客户支付处理系统

**交通**

卡车制动器诊断系统，车辆调度，运送系统

结论

神经网络应用的数量、投入到神经网络软硬件上的资金和公众对这些设计的兴趣都在快

速增长。 1-7

## 1.4 生物学的启示

本书所讲的人工神经网络与它对应的生物神经网络有很大区别。本节我们将简单介绍人脑功能中那些对人工神经网络研究有启示的特征。

人脑由大量(约 $10^{11}$个)高度互连的单元(每个单元约有 $10^4$ 个连接)组成。这些单元被称为神经元。就研究的目的来看，这些神经元由三部分组成：树突、细胞体和轴突。树突是树状的神经纤维接收网络，它将电信号传送到细胞体，细胞体对这些输入信号进行整合并进行阈值处理。轴突是单根长纤维，它把细胞体的输出信号导向其他神经元。一个神经细胞的轴突和另一个神经细胞树突的结合点称为突触。神经元的排列和突触的强度(由复杂的化学过程决定)确立了神经网络的功能。图 1-1 是两个生物神经元的简化图示。

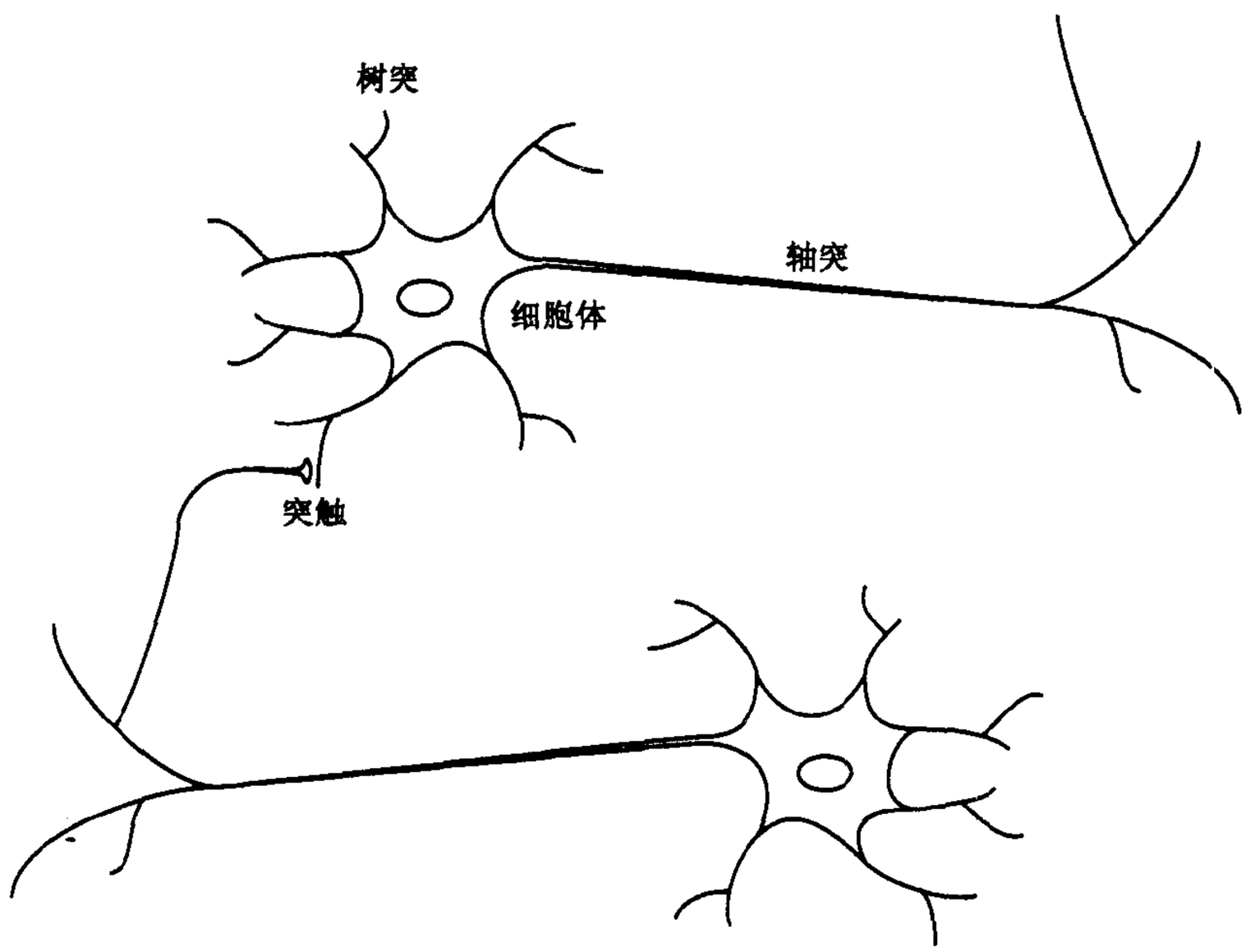



图 1-1 生物神经元简图

一些神经结构是与生俱来的，而其他部分则是在学习的过程中形成的。在学习的过程中，可能会产生一些新的连接，一些连接也可能会消失。这个过程在生命早期最为显著。比如，如果在某一段关键的时期内禁止一只小猫使用它某一只眼睛，则它的这只眼在以后很难形成正常的视力。 1-8

神经结构在整个生命期内不断地进行着改变，后期的改变主要是加强或减弱突触连接。例如，现在已经确认，新记忆的形成是通过改变突触强度而实现的。所以，认识一位新朋友面孔的过程中包含了各种突触的改变过程。

人工神经网络却没有人脑那么复杂，但它们之间有两个关键相似之处。首先，两个网络的构成都是可计算单元的高度互连(虽然人工神经元比生物神经元简单得多)。其次，处理单元之间的连接决定了网络的功能。本书的根本目标就是在人工神经网络中采用合适的连接来

解决特定的问题。

值得注意的是，虽然生物神经元相对于电子电路来说非常慢($10^{-3}$秒相对于 $10^{-9}$秒)，人脑却能以比现有计算机快得多的速度完成许多任务。这主要是因为生物神经网络具有巨大的并行性，即所有的神经元能同时操作。即使大多数人工神经网络是在传统的数字计算机上实现的，但并行处理结构使它们适合于采用 VLSI、光学器件和并行处理技术实现。

下一章我们将介绍基本的人工神经元，并将解释如何将这些神经元组合起来形成网络。
1-9 这主要是为第 3 章提供背景知识，在第 3 章中我们将会看到能实际工作的神经网络。

## 参考文献

[Ande72] J. A. Anderson, "A simple neural nerwork generating an interactive memory," *Mathematical Biosciences*, vol. 14, pp. 197 - 220, 1972.

Anderson 为相联存储器提出了一种"线性相联器"模型。这种模型使用推广的 Hebb 训练规则，它能学习输入向量和输出向量之间的相联关系。它强调网络从生理学的角度出发考虑网络的合理性。Kohonen 同时发表了一篇与该文相近的论文[Koho72]，不过他们是独立进行研究的。

[AnRo88] J. A. Anderson and E. Rosenfeld: *Neurocomputing: foundations of Research*, Cambridge, MA: MIT Press, 1989.

该书是有关神经网络计算的一本非常有价值的参考书，其中录入了四十多篇有关神经网络计算的重要文献，每篇文章都附有一个简介，说明其研究结果以及该文献在神经计算研究领域的历史地位。

[DARP88] *DARPA Neural Network Study*, Lexington, MA: MIT Lincoln Laboratory, 1988.

这是关于到 1988 年为止对有关神经网络知识的一个摘要总结。其中介绍了神经网络的理论基础，并讨论了它们当时的应用，包含相联存储器、递归网络、视频、语音识别和机器人等方面的知识。最后还讨论了模拟工具和实现技术。

[Gros76] S. Grossberg, "Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors," *Biological Cybernetics*, Vol. 23, pp. 121 - 134, 1976.

Grossberg 介绍了一种基于可视系统的自组织神经网络。这种网络中有两种记忆机制：短期记忆和长期记忆，是一种连续时间竞争网络。它是构成自适应谐振理论
1-10 (ART)网络的基础。

[Gros80] S. Grossberg, "How does the brain build a cognitive code?" *Psychological Review*, Vol. 88, pp. 375 - 407, 1980.

Grossberg 的 1980 年论文所提出的神经网络结构和机制可以解释许多生理行为，如空间频率自适应，双目竞争等。他的系统可以在无外界帮助的情况下完成自动校正。

[Hebb49] D. O. Hebb, *The Organization of Behavior*. New York: Wiley, 1949.

这本重要著作的主要假定是行为可以由神经元的行为来解释。在这本书中 Hebb 提出了第一个学习规则，这是在细胞一级进行学习的基本原理。Hebb 从单个神经元的属性出发提出了典型的生物条件反射是存在的。

[Hopf82] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences*, Vol. 79, pp. 2554 - 2558, 1982.

Hopfield 提出了按内容寻址的神经网络。同时还给出了有关他的网络是如何工作的以及该网络能够做什么的完整描述。

[Koho72] T. kohonen, "Correlation matrix memories," *IEEE transactions on Computers*, Vol. 21, pp. 353 - 359, 1972.

Kohonen 为联想存储器提出了联想矩阵模型。这种模型用 Hebb 规则进行训练，从而学会在输入和输出向量之间的相关性。这里强调的是网络的数学结构。同时，Anderson 也发表了一篇内容相近的独立完成论文[Ande72]。

[McPi43] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous acrivity ," *Bulletin of Mathematical Biophysics.*, Vol. 5, pp. 115 - 133, 1943.

该文介绍了第一个神经元数学模型。在这里神经元的输入信号加权和与阀值比较再决定神经元是否输出。这是在那时侯对计算元素认识的基础上，第一次对大脑工作 1-11
原理描述的尝试。它证明了简单的神经网络可以计算任何数学和逻辑函数。

[MiPa69] M. Minsky and S. Papert, *Perceptrons*, Canvridge, MA: MIT Press, 1969.

一本标志性的著作，其中包含了对什么样的感知机能够学习这一问题的严密研究。感知机的形式化处理不仅要解释感知机的局限性，并且要指明克服它们的方向。不幸的是，该书悲观地预言感知机的局限性说明神经网络研究方向是行不通的。这种偏颇的看法在此后若干年内为神经网络的研究和基金投资泼了凉水。

[Rose58] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and orgnization in the brain," *Psychological Review*, Vol. 65, pp. 388 - 408, 1958.

本文提出了第一个应用神经网络——感知机。

[RuMc86] D. E. Rumelhart and J. L. McClelland, eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1, Cambridge, MA: MIT Press, 1986.

在 20 世纪 80 年代对神经网络重新引起人们重视起到关键作用的两本著作之一。在其主题中，特别提出了训练多层神经网络的反传算法。

[WiHo60] B. Widrow and M. E. Hoff, "Adaptive switching circuits," *1960 IRE WESCON Convention Record*, New York: IRE Part 4, pp. 96 - 104, 1960.

在这篇重要的文献中介绍了能进行快速、准确学习的类感知机神经网络。作者假设系统一定有输入和对每一个输入有所希望的输出分类，然后系统计算实际输出与所希望的输出的误差。用梯度下降法进行权值调整，以达到最小均方误差。(最小均方误差算法或 LSM 算法。) 1-12

该文在[Anro88]中被重印。

# 第2章　神经元模型和网络结构

## 2.1　目的

第 1 章给出了生物神经元和神经网络的简述。现在来介绍简化的神经元数学模型，并解释这些人工神经元如何相互连接形成各种网络结构。另外，本章还将通过几个简单的实例阐述这些网络如何工作。本书中将使用本章所引入的概念和符号。

这一章没有覆盖该书中所用到的所有结构，但是给出了其他结构所要用到的基本模块。更复杂的结构将在后面几章中用到的地方介绍和讨论。不过这里也会给出它们的许多细节。注意，我们不要求读者第一次阅读就记住本章的所有内容，但要把它作为你开始着手学习的实例和以后要温习的资料。

2-1

## 2.2　原理和实例

### 2.2.1　符号

神经网络是门新兴学科，迄今为止，人们还并没有对其建立严格的数学符号和结构化表示。另外，神经网络方面的论文和书籍均是来自诸如工程、物理、心理学和数学等许多不同领域，作者都习惯使用本专业的特殊词汇。于是，神经网络的许多文献都难以阅读，概念也较实际情况更为复杂。这实在令人感到惭愧，因为这些妨碍了许多重要思想的传播，并且导致了不止一次的“重复发明”。

在本书中，我们尽可能地使用标准符号，在不失严格的条件下使之简单明了。特别地，这里将尽力保留已有的使用习惯，并使其前后一致。

本书中的图、数学公式以及解释图和数学公式的正文，将使用以下符号：

- 标量：小写的斜体字母，如$a$，$b$，$c$。
- 向量：小写的黑正体字母，如$\mathbf{a}$，$\mathbf{b}$，$\mathbf{c}$。
- 矩阵：大写的黑正体字母，如$\mathbf{A}$，$\mathbf{B}$，$\mathbf{C}$。

本章将引入有关网络结构的其他一些符号。全书中用到的所有符号都可以在附录 B 中查到。所以，如果你有何疑问，可以查附录 B。

### 2.2.2　神经元模型

1. 单输入神经元

**权值　偏置(值)　净输入　传输函数**　一个单输入神经元如图 2-1 所示。标量输入 $p$ 乘上标量权值 $w$ 得到 $wp$，再将其送入累加器。另一个输入 1 乘上偏置值 $b$，再将其送入累加器。累加器输出 $n$ 通常被称为净输入，它被送入一个传输函数 $f$，在 $f$ 中产生神经元的标量输出 $a$。(也有一些作者将该传输函数称为“活跃函数”，将偏置值称为“偏移量”。)

2-2

若将这个简单模型和前面第 1 章所讨论的生物神经元相对照，则权值 $w$ 对应于突触的

连接强度，细胞体对应于累加器和传输函数，神经元输出 $a$ 代表轴突的信号。

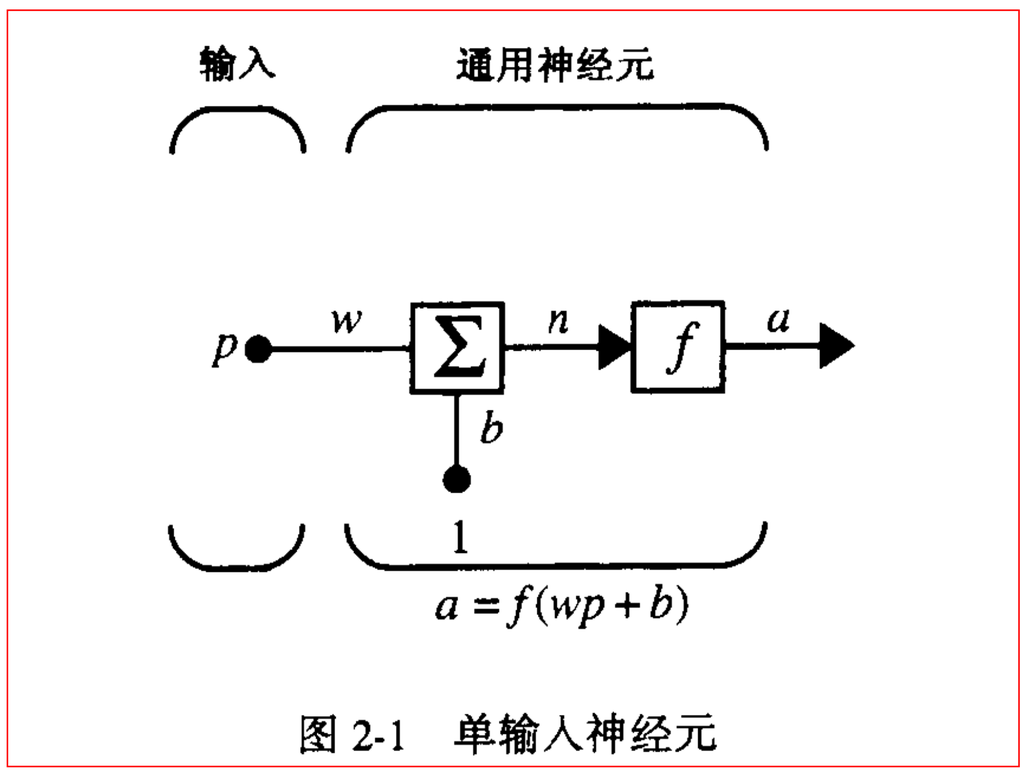



图 2-1　单输入神经元

神经元输出按下式计算：

$$a = f(wp + b)$$

例如，若 $w = 3$，$p = 2$，$b = -1.5$，则

$$a = f(3(2) - 1.5) = f(4.5)$$

实际输出取决于所选择的特定传输函数。下面一节将讨论传输函数。

偏置值除了有常数输入值 1 之外，它很像一个权值。但是，如果不想在神经元中使用偏置值，也可以忽略它。在后面第 3 章、第 7 章和第 14 章中将出现这样的情况。

注意，$w$ 和 $b$ 是神经元的可调整标量参数。设计者也可以选择特定的传输函数，在一些学习规则中调整参数 $w$ 和 $b$，以满足特定的需要(参见第 4 章学习规则)。正如将在下一节所讨论的，依据不同目的可以选择不同的传输函数。

2. 传输函数

图 2-1 中的传输函数可以是 $n$ 的线性或非线性函数。可以用特定的传输函数满足神经元要解决的特定问题。

本书包括了各种不同的传输函数。下面将讨论其中最常用的三种。

**硬极限传输函数**　硬极限传输函数如图 2-2 中的左图所示，当函数的自变量小于 0 时，函数的输出为 0；当函数的自变量大于或等于 0 时，函数的输出为 1。用该函数可以把输入分成两类。第 4 章将广泛使用该传输函数。 2-3

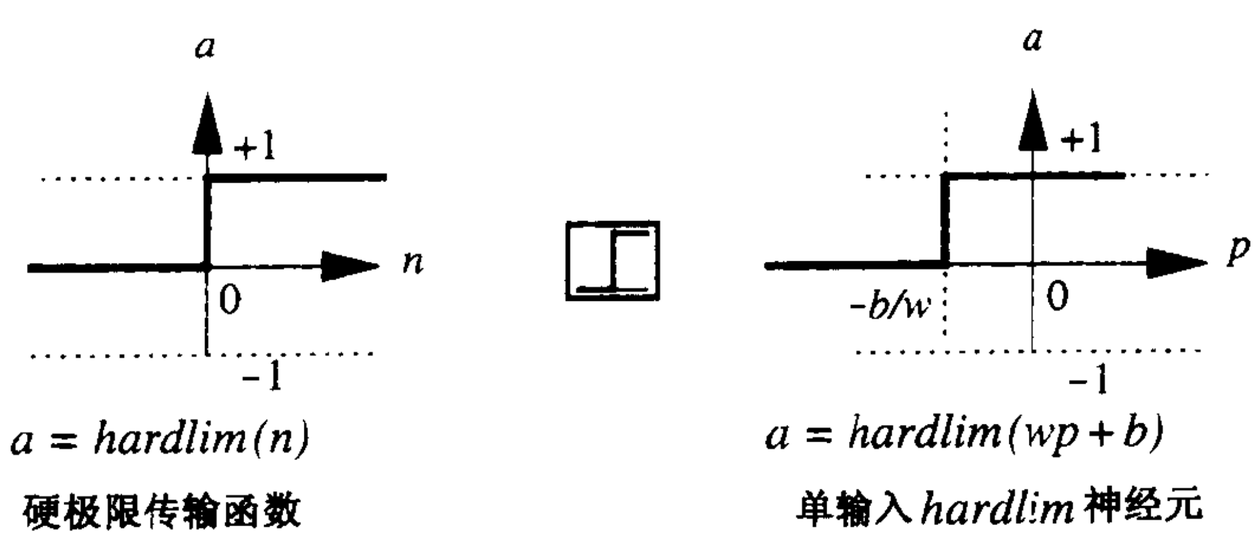



图 2-2　硬极限传输函数

图 2-2 的右图描述了使用硬极限传输函数的单输入神经元的输入/输出特征曲线。从中可看出权值和偏置值的影响。注意，两图之间的图标代表硬极限传输函数。在网络图中的这个图标表示使用了该传输函数。

**线性传输函数** 线性传输函数的输出等于输入(如图 2-3 所示)：

$$a = n \tag{2.1}$$

在第 10 章讨论的 ADALINE 网络中，神经元使用的是该传输函数。

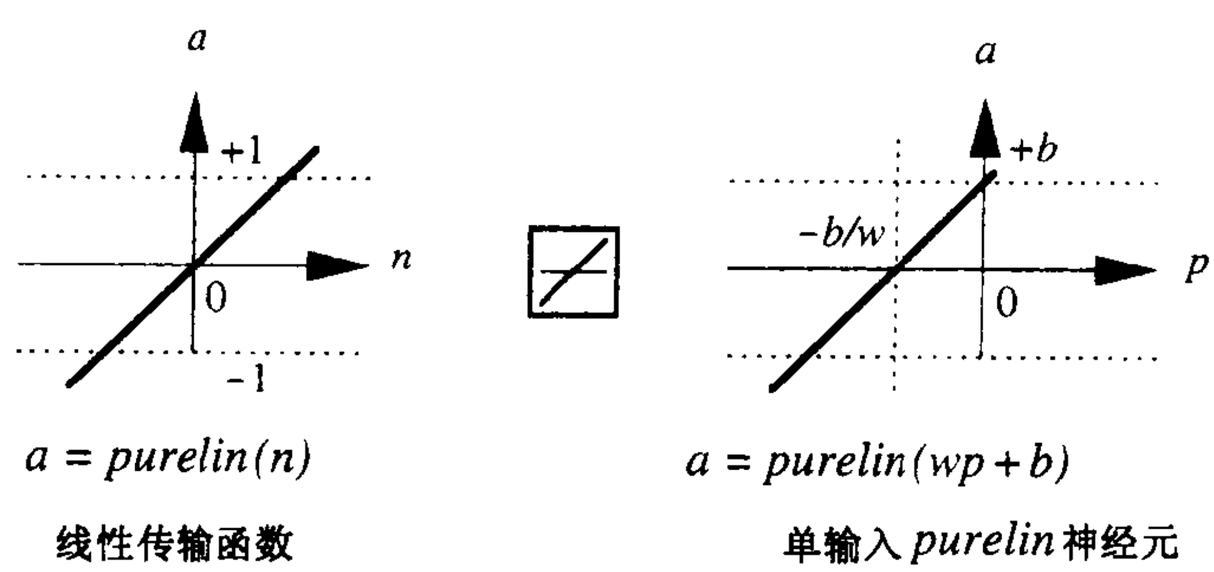



图 2-3 线性传输函数

2-4

图 2-3 右图是带偏置值的单输入线性神经元的输入/输出特征曲线。

**对数-S 形传输函数** 对数-S 形(logsig)传输函数如图 2-4 所示。

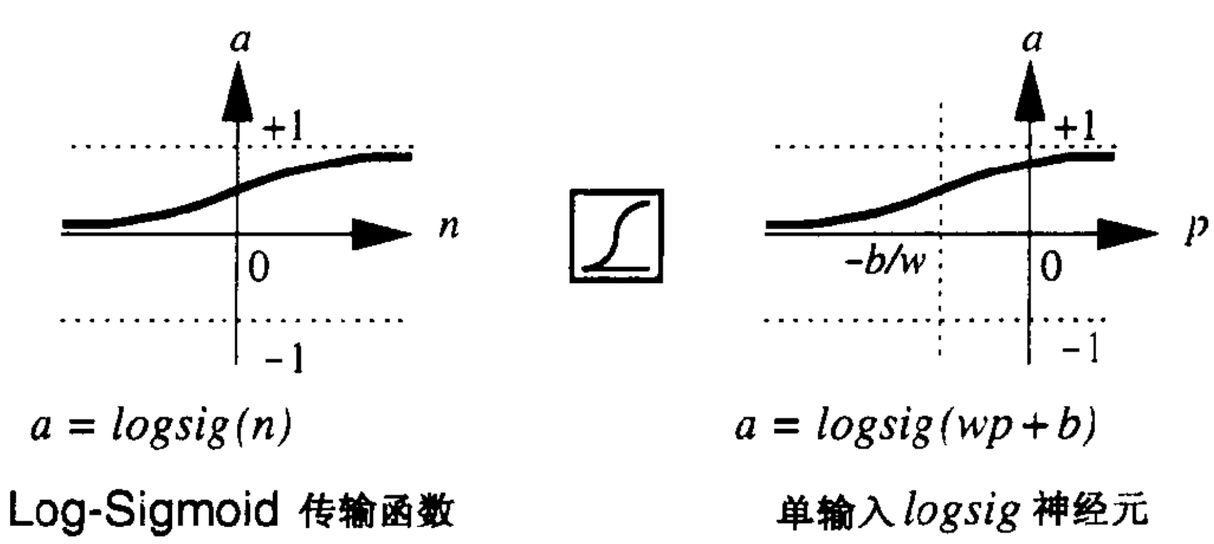



图 2-4 对数-S 形传输函数

该传输函数的输入在(－∞，∞)之间取值，输出则在 0 到 1 之间取值，其数学表达式为：

$$a = \frac{1}{1 + e^{-n}} \tag{2.2}$$

在某种程度上可以说，正是由于对数-S 形函数是可微的，所以用于反传(BP)算法训练的多层网络才采用了该传输函数。

本书所用的大多数传输函数在表 2-1 中都可以找到。当然，你也可以定义不同于表 2-1 的传输函数。

2-5

要对单输入神经元进行实验，可使用 *Neural Network Design Demonstration One-Input Neuron* (**nnd2n1**)。

**表 2-1　传输函数**

| 名　　称 | 输入/输出关系 | 图　　标 | MATLAB 函数 |
|---|---|---|---|
| 硬极限函数 | $a=0,\ n<0$<br>$a=1,\ n\geqslant 0$ | | hardlim |
| 对称硬极限函数 | $a=-1,\ n<0$<br>$a=+1,\ n\geqslant 0$ | | hardlims |
| 线性函数 | $a=n$ | | purelin |
| 饱和线性函数 | $a=0,\ n<0$<br>$a=n,\ 0\leqslant n\leqslant 1$<br>$a=1,\ n>1$ | | satlin |
| 对称饱和线性函数 | $a=-1,\ n<-1$<br>$a=n,\ -1\leqslant n\leqslant 1$<br>$a=1,\ n>1$ | | satlins |
| 对数-S 形函数 | $a=\frac{1}{1+e^{-n}}$ | | logsig |
| 双曲正切 S 形函数 | $a=\frac{e^{n}-e^{-n}}{e^{n}+e^{-n}}$ | | tansig |
| 正线性函数 | $a=0,\ n<0$<br>$a=n,\ n\geqslant 0$ | | poslin |
| 竞争函数 | $a=1$，具有最大 $n$ 的神经元<br>$a=0$，所有其他神经元 | C | compet |

2-6

3．多输入神经元

**权值矩阵**　通常，一个神经元有不止一个输入。具有 $R$ 个输入的神经元如图 2-5 所示。其输入 $p_1$，$p_2$，…，$p_R$ 分别对应权值矩阵 $\mathbf{W}$ 的元素 $w_{1,1}$，$w_{1,2}$，…，$w_{1,R}$。

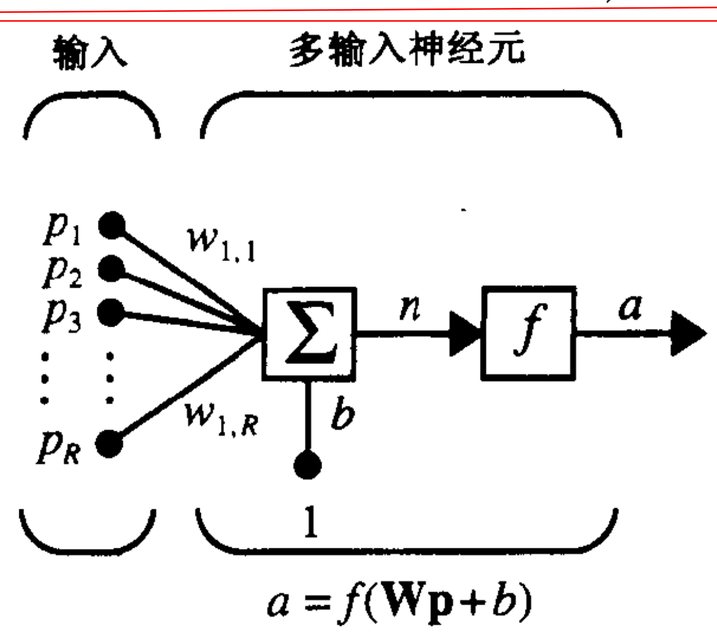



图 2-5　多输入神经元

该神经元有一个偏置值 $b$，它与所有输入的加权和累加，从而形成净输入 $n$：

$$n = w_{1,1}p_1 + w_{1,2}p_2 + \cdots + w_{1,R}p_R + b \tag{2.3}$$

这个表达式也可以写成矩阵形式：

$$n = \mathbf{Wp} + b \tag{2.4}$$

其中单个神经元的权值矩阵 $\mathbf{W}$ 只有一列元素。

神经元的输出可以写成：

$$a = f(\mathbf{Wp} + b) \tag{2.5}$$

所幸的是，神经网络通常可以用矩阵来描述。全书也将采用这种矩阵描述方法。请不要为矩阵和向量运算担心，我们将在第 5，6 章复习这些内容，并给出一些例题及其求解方法。

**权值下标** 本书将采用习惯的方法表示权值矩阵元素的下标。权值矩阵元素下标的第一个下标表示权值相应连接所指定的目标神经元编号，第二个下标表示权值相应连接的源神经元编号。据此，$w_{1,2}$的含义是：该权值表示从第二个神经元到第一个神经元的连接。在本章稍后就会看到，这种习惯表示法在有多个神经元时很有用。

2-7

**简化符号** 我们可以画出一个由几个神经元组成的网络，每个神经元都有几个输入。而且，一个网络还可以有几层神经元。可以想像，当画出所有神经元之间的连接后，网络将会多么复杂。网络中的大量连接会使得网络难被看懂，而且对连接的详细描述也会掩盖网络的主要特征，所以本书将采用简化符号来表示神经元。图 2-6 为利用这种符号所表示的多输入神经元。

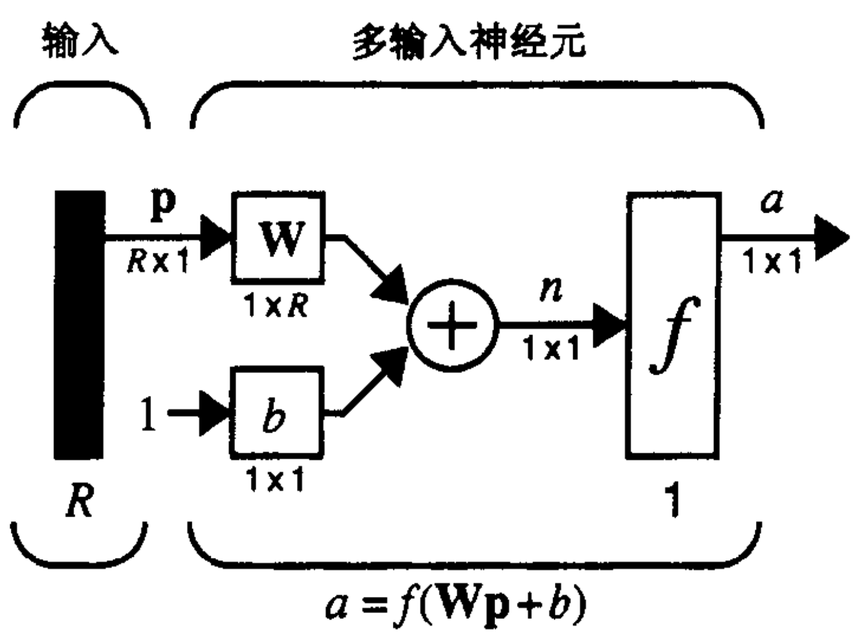



图 2-6 具有 $R$ 个输入的神经元的简化符号

在图 2-6 中，左边垂直的实心条表示输入向量 $\mathbf{p}$，$\mathbf{p}$ 下面的变量 $R \times 1$ 表示 $\mathbf{p}$ 的维数，也即输入是由 $R$ 个元素组成的一维向量。这些输入被送入权值矩阵 $\mathbf{W}$，$\mathbf{W}$ 有 1 行 $R$ 列。常量 1 则作为输入与标量偏置值 $b$ 相乘。传输函数 $f$ 的净输入是 $n$，它是偏置值 $b$ 与积 $\mathbf{Wp}$ 的和。在这种情况下，神经元的输出 $a$ 是一个标量。如果网络有多个神经元，那么网络输出就可能是一个向量。

在简化符号图中，一般要标出变量的维数，这样可以立即知道该变量是一个标量，还是一个向量，抑或是一个矩阵，而不必费心去猜变量的类型或维数。

请注意，网络的输入是由问题的外部描述决定的。比如要设计神经网络来预测风筝飞行条件。输入应该是空气的温度、风速、湿度，这样神经网络就有三个输入。

2-8

要对两输入神经元进行实验，可使用 *Neural Network Design Demonstration Two-Input Neuron* (**nnd2n2**)。

### 2.2.3 网络结构

一般来说，有多个输入的单个神经元并不能满足实际应用的要求。在实际应用中需要有多个并行操作的神经元，这里将这些可以并行操作的神经元组成的集合称为“层”。下面将对这个概念进行讨论。

1. 神经元的层

**层**　图 2-7 是由 $S$ 个神经元组成的单层网络。注意，$R$ 个输入中的每一个均与每个神经元相连，权值矩阵现在有 $S$ 行。

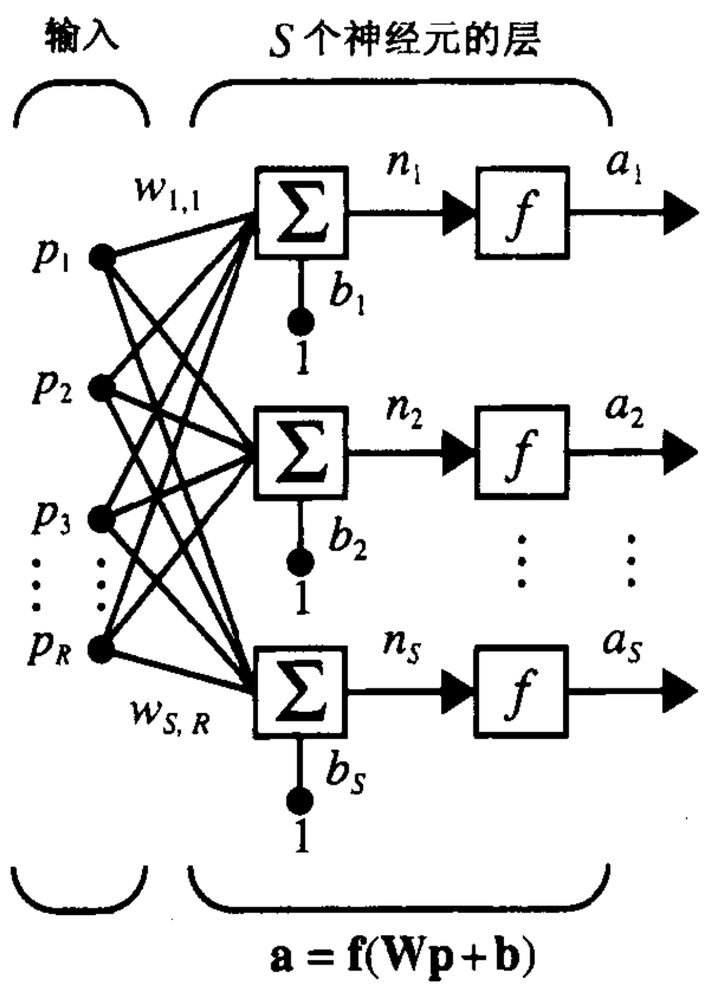



图 2-7　$S$ 个神经元组成的层

该层包括权值矩阵、累加器、偏置值向量 **b**、传输函数框和输出向量 **a**。一些作者也把输入看作是另外一层，但这里并不这样认为。

输入向量 **p** 的每个元素均通过权值矩阵 **W** 和每个神经元相连。每个神经元有一个偏置值 $b_i$、一个累加器、一个传输函数 $f$ 和一个输出 $a_i$。将所有神经元的输出结合在一起，可以得到一个输出向量 **a**。

通常，每层的输入个数并不等于该层中神经元的数目(即是 $R \neq S$)。

也许可能有人要问，同一层中所有神经元是否要有同样的传输函数？回答是否定的。可以把如上所述的两个并行操作网络组合在一起定义一种有不同传输函数的单个神经元(复合) 2-9
层。两个网络都有同样的输入，而每个网络只产生一部分输出。

输入向量通过如下权矩阵 **W** 进入网络：

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix} \tag{2.6}$$

如前所述，矩阵 **W** 中元素的行下标代表该权值相应连接输出的目的神经元，而列下标

代表该权值相应连接的输入源神经元。那么，$W_{3,2}$的下标表示该元素是从第二个神经元到第三个神经元的连接的权值。

同样，具有 $S$ 个神经元、$R$ 个输入的单层网络也能用简化的符号表示为如图 2-8 所示的形式。

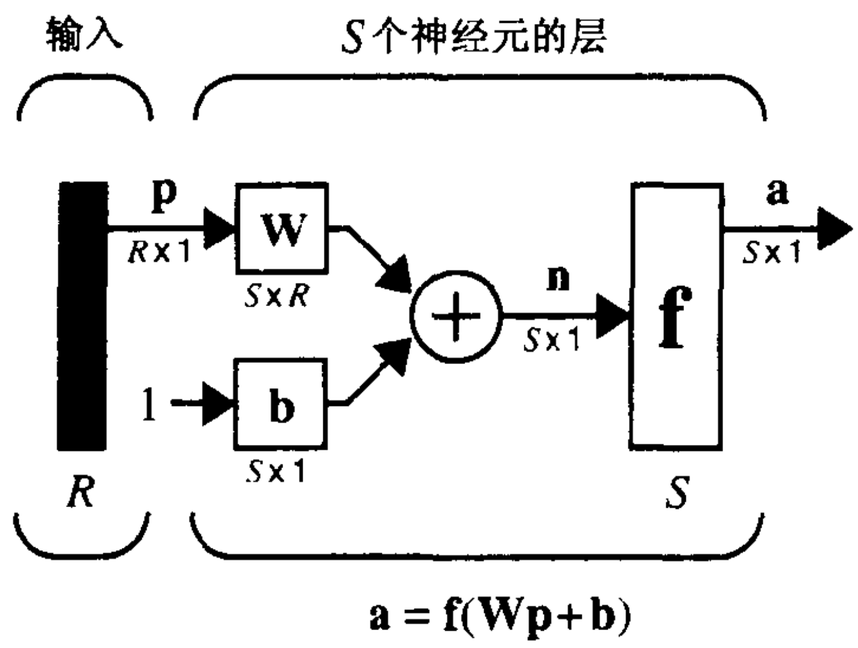


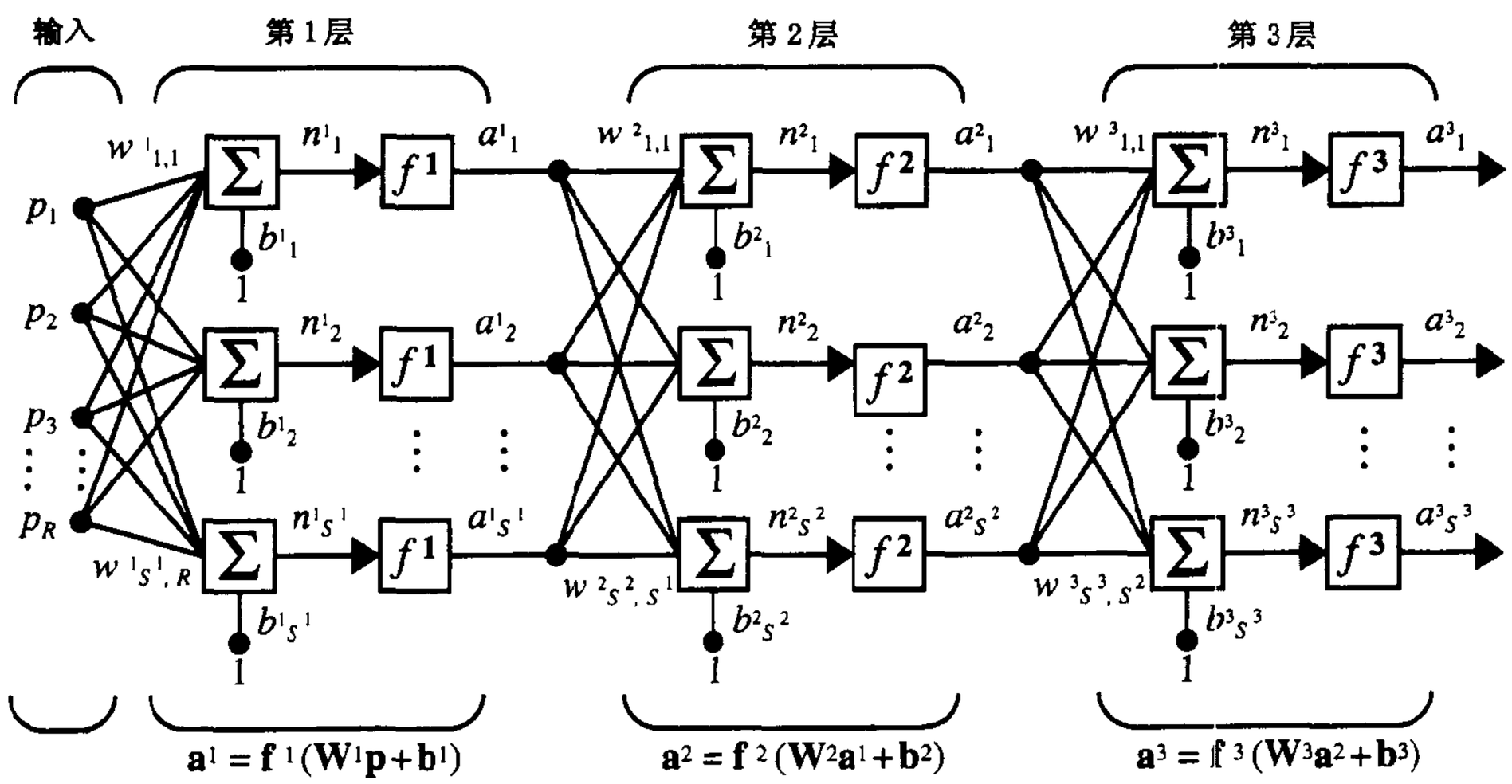

图 2-8　由 $S$ 个神经元组成的层的简化表示

上图中每个变量下的符号指明 **P** 是长度为 $R$ 的向量，**W** 是一个 $S \times R$ 矩阵，**a** 和 **b** 是长度为 $S$ 的向量。如前所述，该层包括权值矩阵、加操作和乘操作、偏置值向量 **b**、传输函数框和输出向量。

2．多层神经元

**层上标**　现在考虑具有几层神经元的网络。每层都有自己的权值矩阵 **W**、偏置值向量 **b**、净输入向量 **n** 和一个输出向量 **a**。这里需要引入额外的符号来区分这些层次。我们可以用上标来标注这些层次，即每个变量都附加一个上标来表示其所处层次。这样，第一层的权值矩阵可以写为 $\mathbf{W}^1$，第二层的权值矩阵可以写为 $\mathbf{W}^2$，等等。如图 2-9 所示的三层网络就使用了这种标记方法。

2-10




$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$

图 2-9　三层网络

如图所示，第一层有 $R$ 个输入、$S^1$ 个神经元，第二层有 $S^2$ 个神经元，等等。要注意不同层可以有不同数目的神经元。

第一层和第二层的输出分别是第二层和第三层的输入。据此，可以将第二层看作是一个单层网络，它有 $R = S^1$ 个输入，$S = S^2$ 个神经元，和一个 $S^1 \times S^2$ 维的权值矩阵 $\mathbf{W}^2$。第二层的输入是 $\mathbf{a}^1$，输出是 $\mathbf{a}^2$。

**输入层　隐含层**　如果某层的输出是网络的输出，那么称该层为输出层，而其他层叫隐含层。上图中的网络有一个输出层(第 3 层)和两个隐含层(第 1 层和第 2 层)。

前面讨论的三层网络同样也可以用简化的符号表示，如图 2-10 所示。 2-11

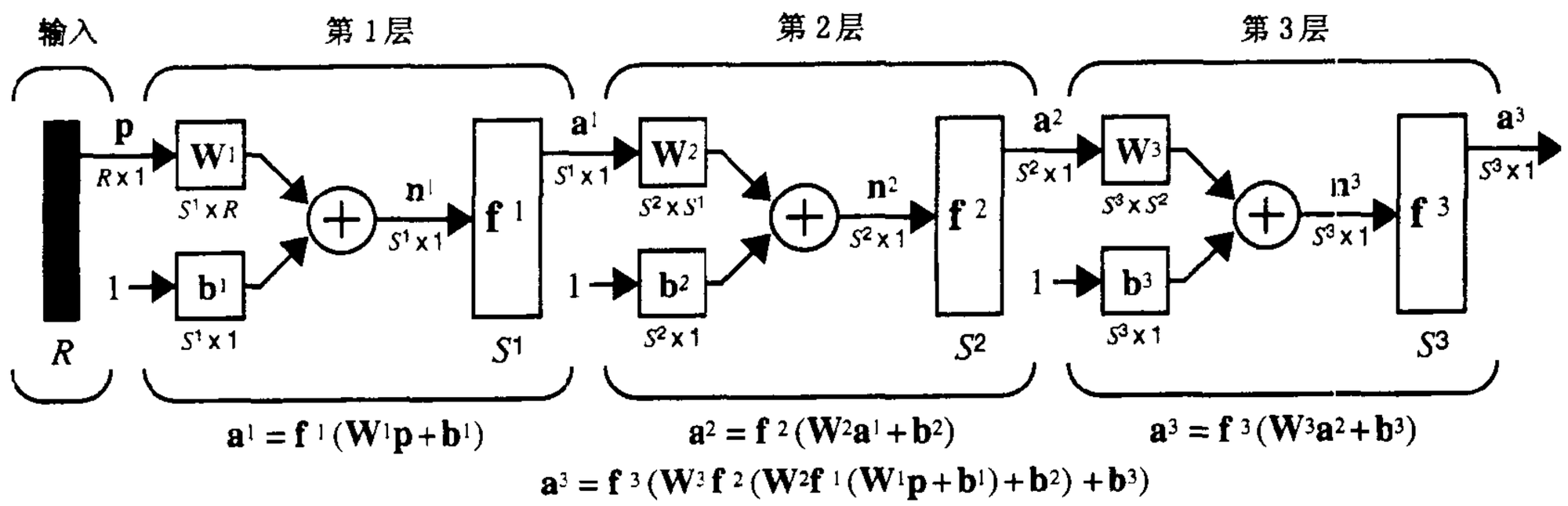



图 2-10　三层网络的简化表示

多层网络的功能要比单层网络强大得多。比如，一个第一层具有 S 形传输函数、第二层具有线性传输函数的网络，经过训练可对大多数函数达到任意精度的逼近，而单层网络则不能做到这一点。

从上面讨论可以看出，决定一个网络的神经元个数非常重要。现在我们就来考虑这个问题。这个问题并不像看起来那么复杂。首先，可以回忆一下，网络的输入/输出神经元的数量是由问题外部描述定义的。所以，如果有 4 个外部变量作为网络输入，那么网络就有 4 个输入。同样，如果网络有 7 个输出，那么网络的输出层就应该有 7 个神经元。最后，输出信号所期望的特征有助于选择输出层的传输函数。如果一个输出要么是 −1，要么是 1，那么该输出神经元就可以用对称硬极限传输函数。所以，单层网络结构完全由问题描述决定，包括特定的输入/输出数和输出信号的特征。

那么，如果网络有两层以上的神经元时，又将如何确定各层的神经元数目？其实问题的关键在于外部问题并没有直接指明隐含层需要的神经元数目。实际上，精确预测隐含层所需要的神经元的数目至今仍然存在一些在理论上还没有解决的问题。这个问题是一个十分活跃的研究领域。在第 11 章中讨论反传算法时，将对此进行深入探讨。

至于网络中的神经元层数，大多数实际的神经网络仅仅只有 2 到 3 层神经元，很少有 4 层或更多层。

这里还应该讨论一下偏置值的使用问题。是否使用偏置值是可以选择的。偏置值给网络提供了额外的变量，从而使得网络具有更强的能力，事实也的确是如此。例如，如果没有偏置值，当网络输入 $\mathbf{p}$ 为 0 时，一个神经元的净输入 $n$ 总是为 0。这是不希望出现的，可以通过用偏置值来避免。本书将在第 3 章、第 4 章和第 5 章中讨论偏置值的影响。 2-12

在后面的各章中，一些例题和演示将省略偏置值。在一些情况下，这种简化可以减少网络的参数。如果只有两个变量，非常容易在一个二维平面上画出系统的收敛情况。但是对于3个或更多个变量而言，显示系统的状态将变得比较困难。

3. 递归网络

**延时** 在讨论递归网络前，首先介绍一些简单的构造模块。第一种是延时模块，如图2-11所示。

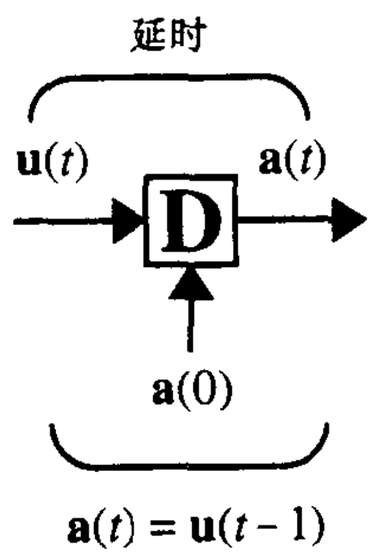



图 2-11 延时模块

延时输出$\mathbf{a}(t)$由输入$\mathbf{u}(t)$根据下式计算得到：

$$\mathbf{a}(t) = \mathbf{u}(t-1) \tag{2.7}$$

所以，输出延时了一个时间步的输入（假设时间以离散步的形式更新，且只取整数值）。等式(2.7)要求在$t=0$时对输出进行初始化。初始条件由图2-11中指向延时块底部的箭头来表示。

**积分器** 另一种将用于第15章至第18章中的连续时间递归网络的构造模块是积分器，

2-13 如图2-12所示：

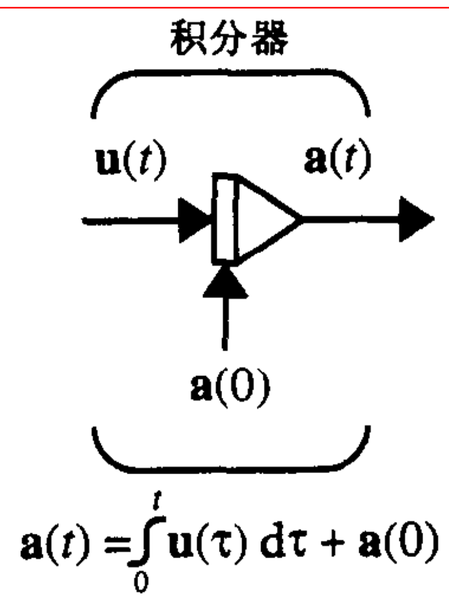


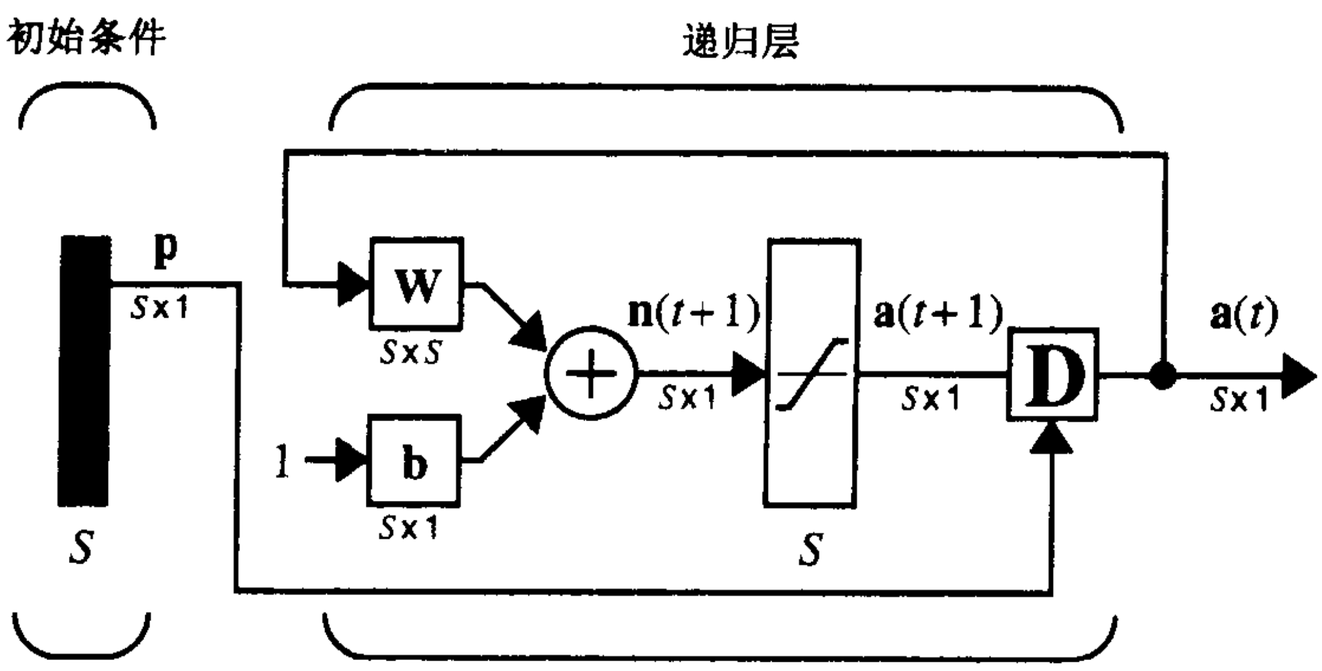

图 2-12 积分器模块

积分器的输出$\mathbf{a}(t)$由输入$\mathbf{u}(t)$根据下式计算得到：

$$\mathbf{a}(t) = \int_0^t \mathbf{u}(\tau)\mathrm{d}\tau + \mathbf{a}(0) \tag{2.8}$$

初始条件$\mathbf{a}(0)$由指向积分器模块底部的箭头来表示。

**递归网络** 利用上述模块就可以构造出递归网络。一个递归网络是一个带反馈的网络，其部分输出连接到它的输入。这与前面所讨论的没有反馈连接的严格前馈网络有很大不同。图2-13给出了一种类型的离散时间递归网络。




$$\mathbf{a}(0)=\mathbf{p} \qquad \mathbf{a}(t+1)=\mathbf{satlins}(\mathbf{W}\mathbf{a}(t)+\mathbf{b})$$

图 2-13　递归网络

2-14

在该网络中，向量 $\mathbf{p}$ 给出了其初始条件(即$\mathbf{a}(0)=\mathbf{p}$)。网络根据其前一次输出计算当前的输出：

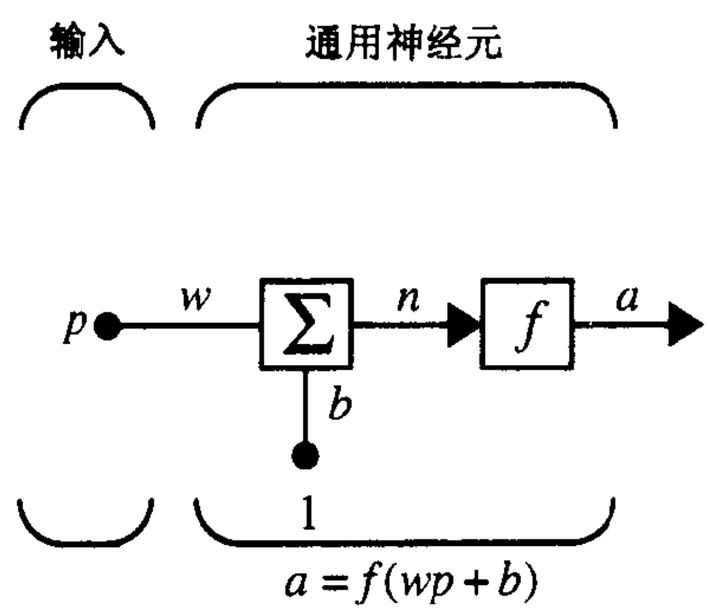
$$\mathbf{a}(1)=\mathbf{satlins}(\mathbf{W}\mathbf{a}(0)+\mathbf{b}),\mathbf{a}(2)=\mathbf{satlins}(\mathbf{W}\mathbf{a}(1)+\mathbf{b}),\cdots$$

递归网络比前馈网络在本质上具有更强的能力，它可以表现出时间性行为。本书的第 3 章和第 15 章至第 18 章将讨论这种类型的网络。

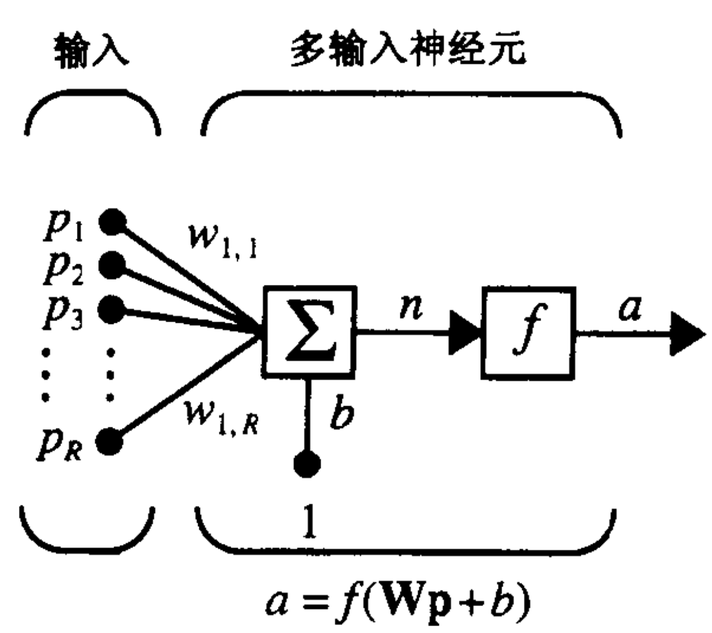
2-15

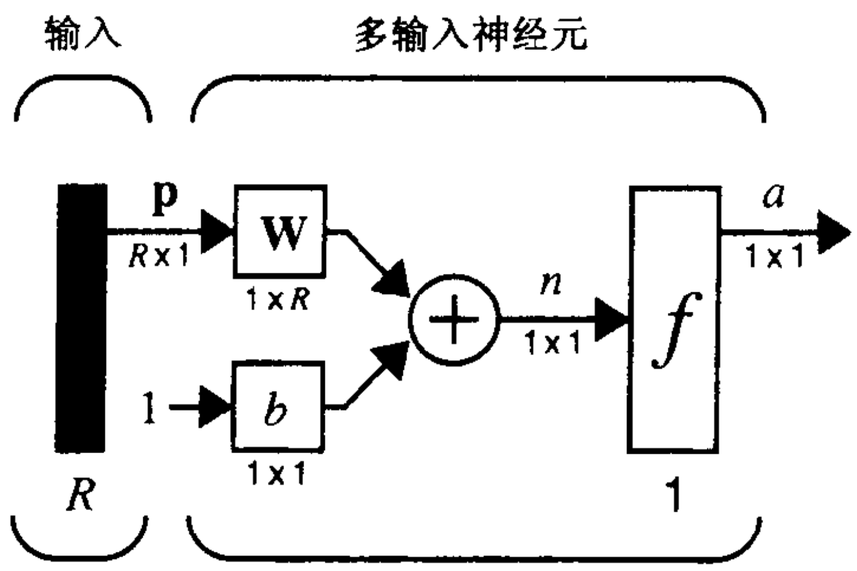
## 2.3　小结

### 单输入神经元




### 多输入神经元






2-16

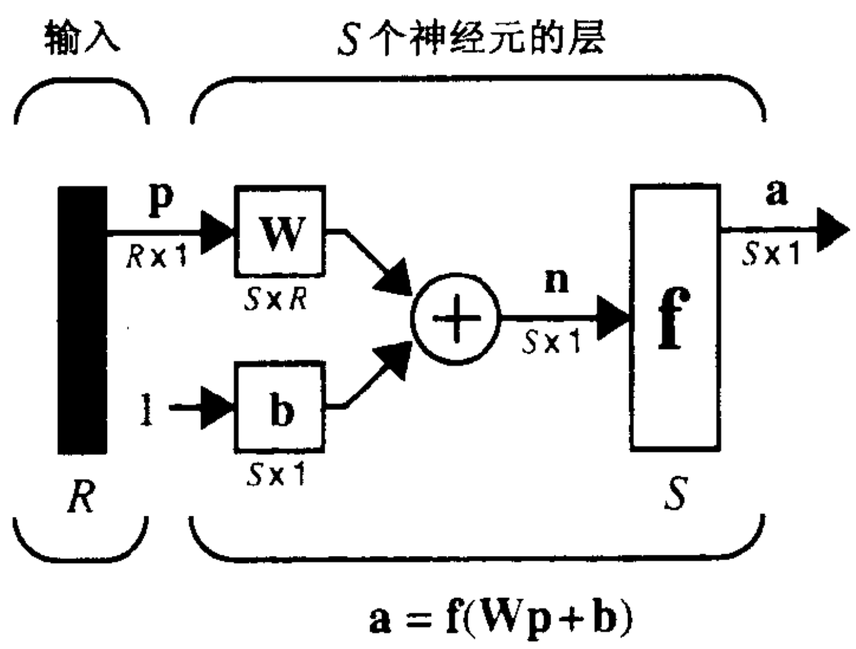
$$a = f(\mathbf{W}\mathbf{p} + b)$$

## 传输函数

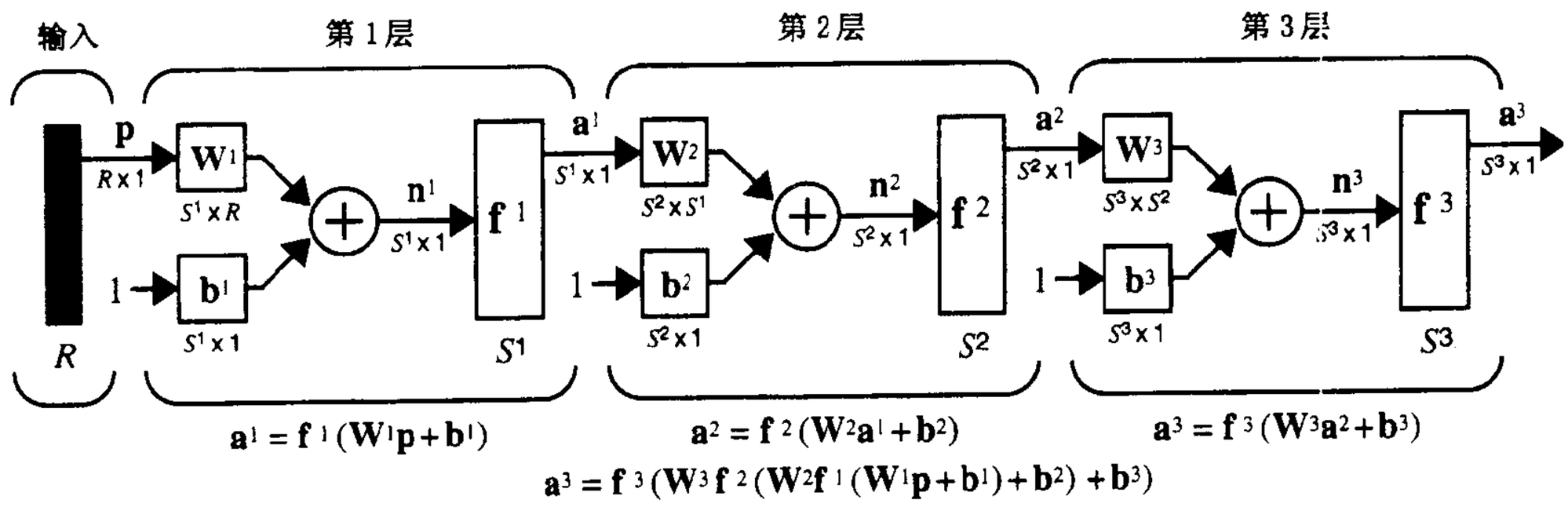
| 名　　称 | 输入/输出关系 | 图　　标 | MATLAB 函数 |
|---|---|---|---|
| 硬极限函数 | $a=0,\ n<0$<br>$a=1,\ n\geqslant 0$ | | hardlim |
| 对称硬极限函数 | $a=-1,\ n<0$<br>$a=+1,\ n\geqslant 0$ | | hardlims |
| 线性函数 | $a=n$ | | purelin |
| 饱和线性函数 | $a=0,\ n<0$<br>$a=n,\ 0\leqslant n\leqslant 1$<br>$a=1,\ n>1$ | | satlin |
| 对称饱和线性函数 | $a=-1,\ n<-1$<br>$a=n,\ -1\leqslant n\leqslant 1$<br>$a=1,\ n>1$ | | satlins |
| 对数-S 形函数 | $a=\dfrac{1}{1+e^{-n}}$ | | logsig |
| 双曲正切 S 形函数 | $a=\dfrac{e^{n}-e^{-n}}{e^{n}+e^{-n}}$ | | tansig |
| 正线性函数 | $a=0,\ n<0$<br>$a=n,\ n\geqslant 0$ | | poslin |
| 竞争函数 | $a=1$，具有最大 $n$ 的神经元<br>$a=0$，所有其他神经元 | C | compet |

2-17

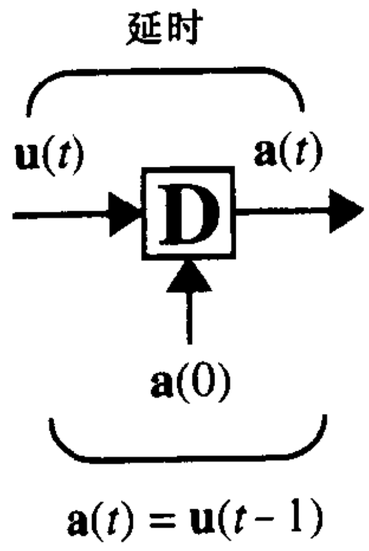
## 神经元层




## 三层神经元

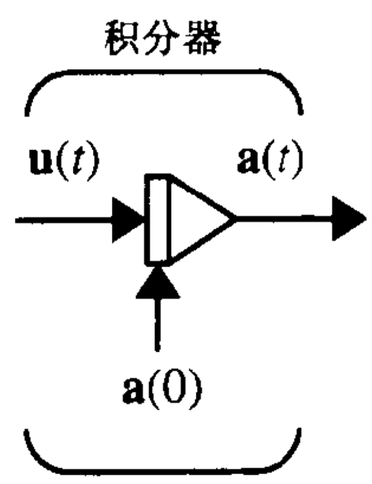



## 延时模块



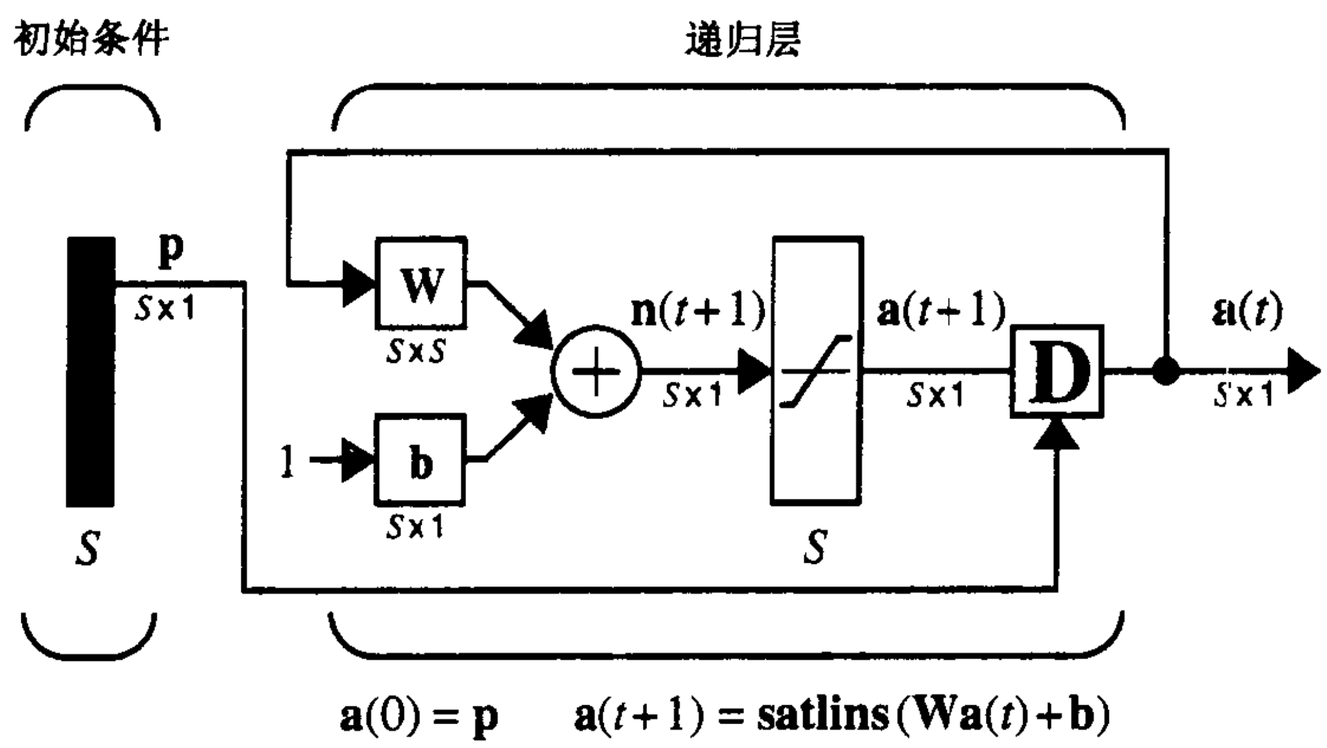

2-18

**积分器模块**




$$\mathbf{a}(t) = \int_0^t \mathbf{u}(\tau)\,d\tau + \mathbf{a}(0)$$

**递归网络**




**如何选取一种网络结构**

应用问题的描述从如下几个方面非常有助于定义网络的结构：

1）网络的输入个数 = 应用问题的输入数；

2）输出层神经元的数目 = 应用问题的输出数目；

2-19 3）输出层的传输函数选择至少部分依赖于应用问题的输出描述。

## 2.4 例题

**P2.1** 一个单输入神经元的输入是 2.0，其权值是 2.3，偏置值是 −3。

（i）传输函数的净输入是多少？

（ii）神经元的输出是多少？

**解**

（i）传输函数的网络输出由下式给出：

$$n = wp + b = (2.3)(2) + (-3) = 1.6$$

（ii）因为未指定传输函数，所以不能确定该神经元的输出。

**P2.2** 如果 P2.1 中的神经元分别具有如下传输函数，请问其输出值分别是多少？

(i) 硬极限函数

(ii) 线性函数

(iii) 对数-S 形(*logsig*)函数

**解**

(i) 对硬极限传输函数有

$$a = hardlim(1.6) = 1.0$$

(ii) 对线性传输函数有

$$a = purelin(1.6) = 1.6$$

(iii) 对对数-S 形传输函数有

$$a = logsig(1.6) = \frac{1}{1 + e^{-1.6}} = 0.8320$$

请用 MATLAB 和目录 MININNET 下的函数 *logsig* 验证此结果(参见附录 B)。

**P2.3** 给定一个具有如下参数的两输入神经元：$b = 1.2, \mathbf{W} = [3 \quad 2], \mathbf{p} = [-5 \quad 6]^T$，试依据下列传输函数计算神经元输出：

(i) 对称硬极限传输函数。

(ii) 饱和线性传输函数。 2-20

(iii) 双曲正切 S 形(*tansig*)传输函数。

**解**

首先计算净输入 $n$：

$$n = \mathbf{Wp} + b = [3 \quad 2]\begin{bmatrix} -5 \\ 6 \end{bmatrix} + (1.2) = -1.8$$

现针对每种传输函数计算该神经元的输出。

(i) $a = hardlims(-1.8) = -1$

(ii) $a = satlin(-1.8) = 0$

(iii) $a = tansig(-1.8) = -0.9468$

**P2.4** 现有一个单层神经网络，具有 6 个输入和 2 个输出。输出被限制为 0 到 1 之间的连续值。叙述该网络的结构，请说明：

(i) 需要多少个神经元?

(ii) 权值矩阵的维数是多少?

(iii) 能够采用什么传输函数?

(iv) 需要采用偏置值吗? 2-21

**解**

该问题的求解结果如下：

(i) 需要两个输出神经元，每个输出一个。

(ii) 对应 2 个神经元和 6 个输入，权值矩阵应有 2 行 6 列(乘积 **Wp** 是一个二元向量)。

(iii) 根据前面所讨论的传输函数性质，选用 *logsig* 传输函数是最适合的。

(iv) 题中未能给出足够的条件以确定是否需要偏置值。

## 2.5 结束语

本章介绍了一种简单的人工神经元，并展示了如何通过不同的连接方式将一些神经元组连接起来构造出不同的神经网络。本章的一个主要目的是介绍一些基本表示方法。在随后各章中更为详细讨论各种神经网络时，可能还需回到第 2 章继续熟悉这些基本的表示方法。

本章并未对所讨论的网络进行完整的介绍。完整的介绍将在后面各章展开。在第 3 章中，将会给出使用本章一些网络的一个简单例子，以展示网络的实际运行情况。第 3 章演示的网络是后面所讨论的网络类型的典型代表。

2-22

### 习题

**E2.1** 一个单输入神经元的输入是 2.0，其输入连接的权值是 1.3，偏置值是 3.0。如果它的输出分别为如下一些值，请根据表 2-1 回答，它分别可以采用哪些传输函数？

(i) 1.6

(ii) 1.0

(iii) 0.9963

(iv) −1.0

**E2.2** 假设一个具有偏置值的单输入神经元，现希望当输入值小于 3 时输出是 −1，而输入值大于等于 3 时，其输出值为 +1。请问：

(i) 需要什么类型的传输函数？

(ii) 偏置值应该取多大？它与输入连接的权值相关吗？如果相关，如何相关？

(iii) 通过指定传输函数的名称、描述偏置值和权值来概括该网络。请画出该网络的图形。用 MATLAB 验证网络的性能。

**E2.3** 给定一个具有如下权值矩阵和输入向量的两输入神经元：$\mathbf{W}=[3 \quad 2]$，且 $\mathbf{p}=[-5 \quad 7]^T$。希望其输出值为 0.5。请问是否存在偏置值和传输函数的某种组合可以满足这一要求？

(i) 若偏置值为 0，表 2-1 中有能够实现上述功能的传输函数吗？

(ii) 如果使用线性传输函数，存在能够实现上述功能的偏置值吗？如果有，请说明偏置值是什么？

(iii) 如果使用对数-S 形传输函数，存在能够实现上述功能的偏置值吗？如果有，请说明偏置值是什么？

(iv) 如果使用对称硬极限传输函数，存在能够实现上述功能的偏置值吗？如果有，请说明偏置值是什么？

**E2.4** 一个两层神经网络有 4 个输入和 6 个输出。输出值为取值 0 到 1 之间的连续值。对于该网络的结构可以说些什么？特别是：

2-23

(i) 每一层中需要有多少神经元？

(ii) 第一层和第二层的权值矩阵分别是几维？

(iii) 每一层可用哪种类型的传输函数？

2-24

(iv) 每层中都需要偏置值吗？

# 第3章　一个说明性实例

## 3.1　目的

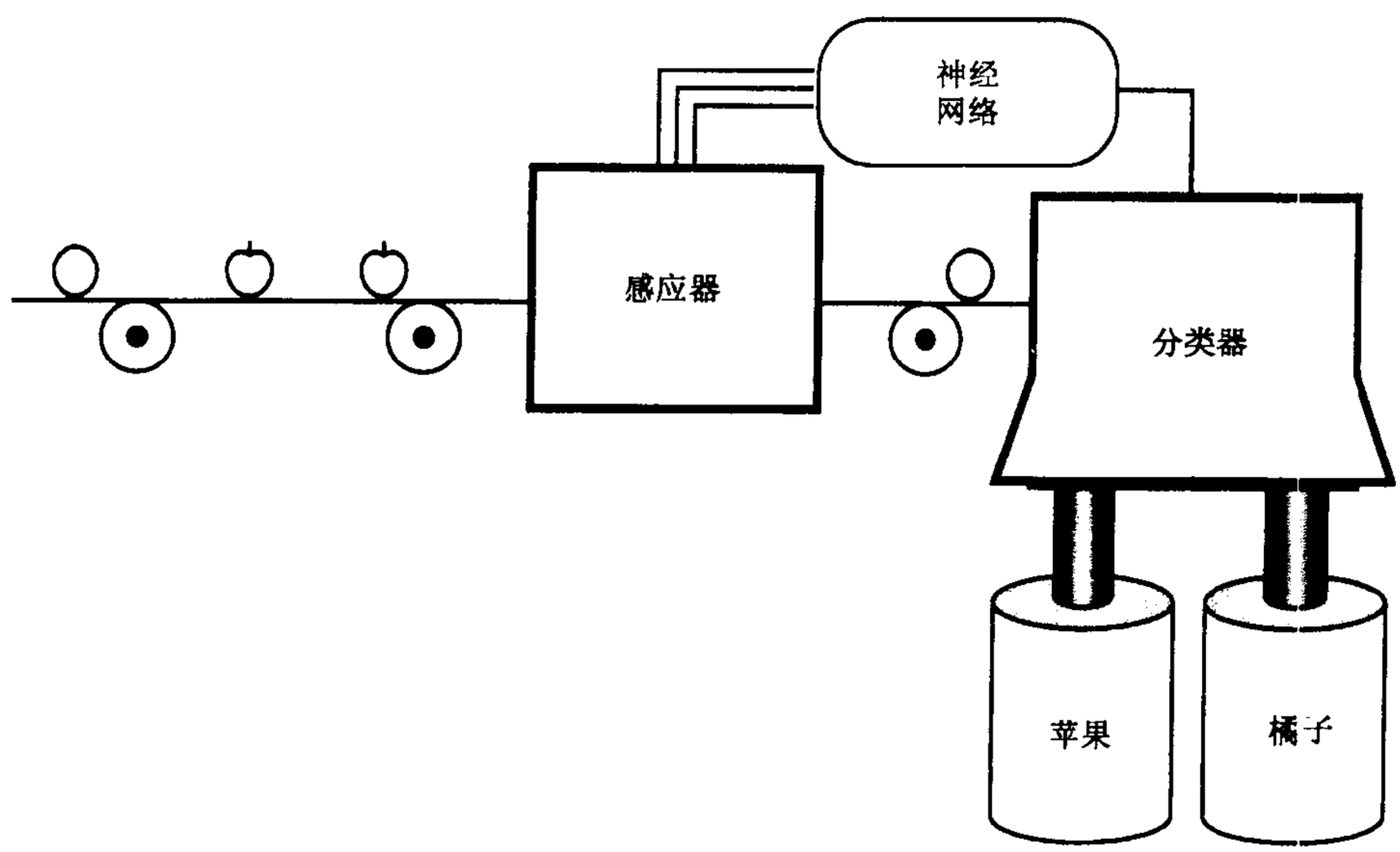
读者可以将这一章看作是后面各章的一个前奏。这里将给出一个模式识别的简单问题，并说明如何用三种不同结构的神经网络来求解这个问题。这将提供一个了解如何利用上一章所给出的网络结构解决实际问题的机会(尽管这个实例过于简单)。不过，也不要期望通过本章的学习就可以完全理解这三种网络。这里之所以直接地给出它们，仅仅是希望读者能够对神经网络的功能有一个感性认识，同时也想说明对给定问题的求解有许多种类型的网络可供使用。

在其余各章还会详细讨论本章所给出的三种网络：前馈网络(这里以感知机为代表)、竞争网络(这里以 Hamming 网络为代表)以及递归联想存储网络(这里以 Hopfield 网络为代表)。 3·1

## 3.2　理论和实例

### 3.2.1　问题描述

某商贩有一个存储各种水果和蔬菜的货仓。当将水果放进货仓时，不同类型的水果可能会混淆在一起，所以商贩非常希望能够有一台能够帮他将水果自动分类摆放的机器。假设从水果卸车的地方到货仓之间有一条传送带。传送带要通过一组特定的传感器，这组传感器可以分别测量水果的三个特征：外形、质地和重量(如下图)。这些传感器功能比较简单。如果



水果基本上是圆形的，外形传感器的输出就为 1；如果水果更接近于椭圆，那么外形传感器的输出就为 - 1。如果水果表面光滑，质地传感器的输出就是 1；如果水果表面比较粗糙，那么质地传感器的输出就为 - 1。当水果重量超过 1 磅时，重量传感器的输出为 1；水果重量轻于 1 磅时，重量传感器的输出为 - 1。

然后，这三个传感器的输出将会输入到神经网络。网络的功能就是要确定传送带上是什么类型的水果，这样才能把不同类型的水果分别送到相应的储存仓内。为了使问题更加简单，现假设传送带上只有两种类型的水果：苹果和橘子。

当每个水果通过这些传感器后，它就可以用一个如式(3.1)所示的三维向量来表示。该
3-2 向量的第一个元素表示外形，第二个元素表示质地，第三个元素表示重量：

$$\mathbf{p} = \begin{bmatrix} \text{外形} \\ \text{质地} \\ \text{重量} \end{bmatrix} \tag{3.1}$$

所以，一个标准橘子可表示为：

$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \tag{3.2}$$

一个标准苹果可表示为：

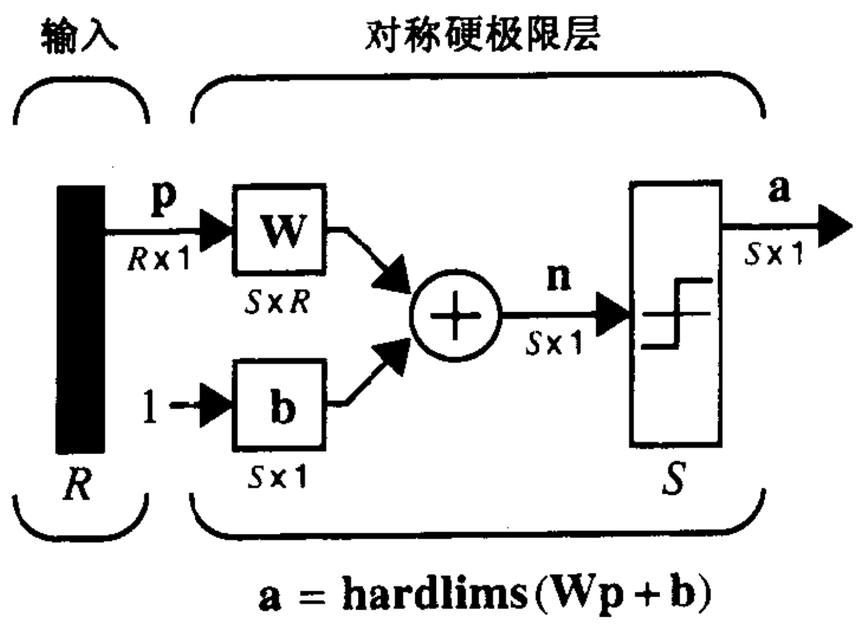
$$\mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} \tag{3.3}$$

对传送带上的每个水果而言，神经网络都可接收到一个三维输入向量，并且必须判断它是一个橘子($\mathbf{p}_1$)还是一个苹果($\mathbf{p}_2$)。

既然对这个简单的普通模式识别问题进行了定义，下面首先讨论求解该问题所要用到的三种不同类型的神经网络。对问题的简化有助于理解这些网络的工作原理。

### 3.2.2 感知机

这里要讨论的第一个网络就是感知机。图 3-1 给出了采用对称硬极限传输函数 *hardlims* 的单层感知机。




3-3 图 3-1 单层感知机

1. 两输入的情况

在用感知机求解橘子/苹果问题之前(它需要一个三输入感知机，也即 $R=3$)，有必要研究一下两输入单神经元感知机($R=2$)的能力。很容易用图示的方法对其进行分析。两输入感知机如图 3-2 所示。

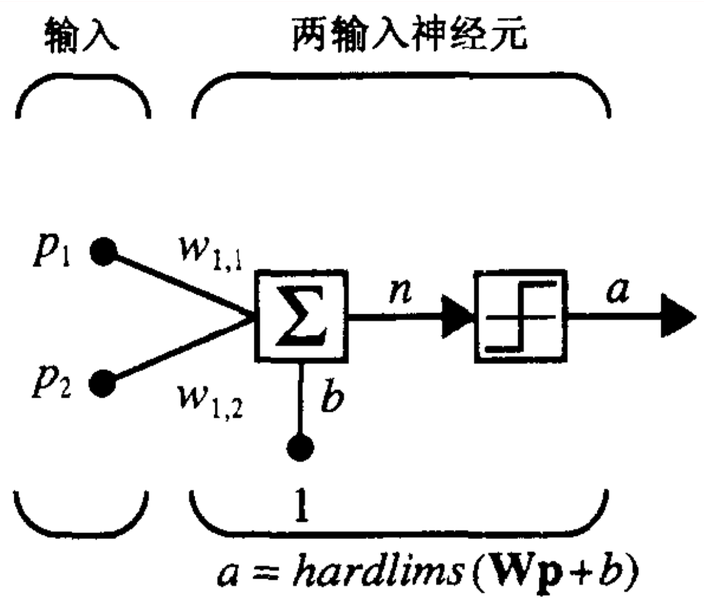



图 3-2　两输入的单神经元感知机

单神经元感知机可将输入向量分成两类。例如，对一个两输入感知机而言，如果 $w_{1,1}=-1$，且 $w_{1,2}=1$，那么

$$a = hardlims(n) = hardlims([-1 \quad 1]\mathbf{p} + b) \tag{3.4}$$

所以，如果权值矩阵(这里是一个只有一行的向量)与输入向量的内积大于等于 $-b$，感知机的输出为 1；如果权值向量和输入的内积小于 $-b$，那么感知机的输出为 $-1$。这就将输入空间划分为两个部分，图 3-3 表明了当 $b=-1$ 的情况下，该感知机对输入空间的这种划分情况。图中的斜线表示净输入 $n$ 等于 0 的各点：

$$n = [-1 \quad 1]\mathbf{p} - 1 = 0 \tag{3.5}$$

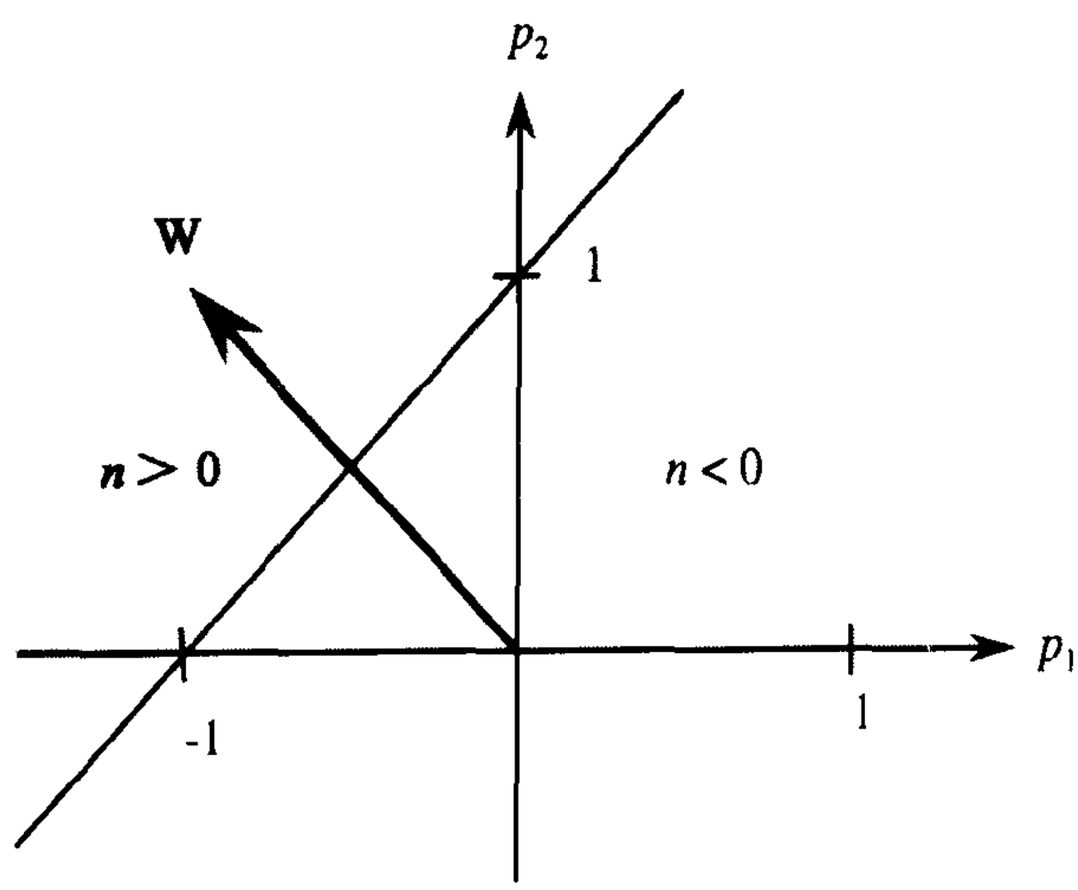



图 3-3　感知机判定边界

请注意该判定边界总是和权值矩阵正交，且边界的位置随 $b$ 的改变而上下移动。(一般来说，W 是由多个行向量组成的矩阵，每一行向量的使用方法都如式(3.5)所示。W 的每一行都会形成一个判定边界。对该问题的详细讨论请参见第 4 章)。阴影区包含的是所有网络

3-4 输出为 1 的输入向量，而对其他输入向量而言，该感知机的输出都为 -1。

所以，单神经元感知机的关键性质是它能够将输入向量分为两类。类与类之间的判定边界由下式给定：

$$\mathbf{Wp} + b = 0 \tag{3.6}$$

因为边界必须是线性的，所以单层感知机只能用于识别一些线性可分(能够用一个线性边界区分)的模式。这些概念将在第 4 章进行更加详细的讨论。

2. 模式识别实例

现在回到前面所给出的橘子/苹果模式识别问题。因为仅仅只有两个类别，所以可采用单神经元感知机。向量输入是三维的($R=3$)，该感知机的输入/输出关系由下式描述：

$$a = hardlims\left(\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \end{bmatrix}\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + b\right) \tag{3.7}$$

现在希望选择适当的偏置值 $b$ 和权值矩阵元素，使得该感知机能够将苹果和橘子区分开来。比如说，如果输入是苹果时，希望该感知机的输出为 1；如果输入是橘子时，希望该感知机的输出为 -1。下面将讨论如何应用图 3-3 所给出的概念，找到一个线性边界将橘子和苹果区分开来。两个标准向量(请参考式(3.2)和式(3.3))的空间表示如图 3-4 所示。从图
3-5 中可以看出对称区分这两个向量的线性边界是 $p_1$ 和 $p_3$ 两个平面。

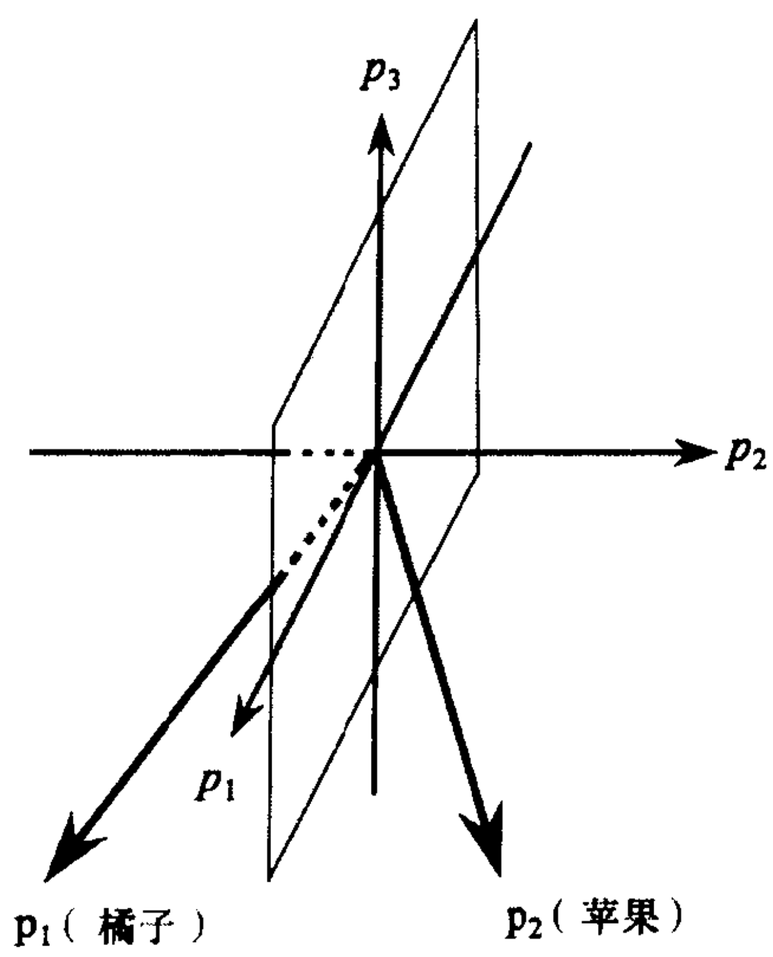



图 3-4 标准向量

$p_1$ 和 $p_3$ 两个平面就是所求的判定边界，可以将其分别表示为

$$p_2 = 0 \tag{3.8}$$

或

$$\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + 0 = 0 \tag{3.9}$$

由此可知权值矩阵和偏置值分别是：

$$\mathbf{W} = [0 \quad 1 \quad 0], b = 0 \tag{3.10}$$

权值矩阵和判定边界正交，且指向含有标准模式 $\mathbf{p}_2$(苹果)的空间区域，在该区域中感知机的输出为 1。由于判定边界通过坐标轴原点，所以偏置值为 0。

下面将对该感知机模式分类器进行测试。 3-6

当输入是**橘子**时，有

$$a = hardlims\left([0 \quad 1 \quad 0]\begin{bmatrix}1\\-1\\-1\end{bmatrix} + 0\right) = -1(\text{橘子}) \tag{3.11}$$

当输入是**苹果**时，有

$$a = hardlims = \left([0 \quad 1 \quad 0]\begin{bmatrix}1\\1\\-1\end{bmatrix} + 0\right) = 1(\text{苹果}) \tag{3.12}$$

由此可以看出，该感知机能够正确区分苹果和橘子。但是，当将一个并不是十分标准的橘子放在分类器中，感知机的输出又将会是什么呢？如果一个椭圆形的橘子通过传感器，那么感知机的输入向量为

$$\mathbf{p} = \begin{bmatrix}-1\\-1\\-1\end{bmatrix} \tag{3.13}$$

网络的响应将是

$$a = hardlims\left([0 \quad 1 \quad 0]\begin{bmatrix}-1\\-1\\-1\end{bmatrix} + 0\right) = -1(\text{橘子}) \tag{3.14}$$

实际上，任何输入向量如果相对于苹果的标准向量而言，更加接近于橘子的标准向量(按欧基里德距离)，那么该输入向量都将被划为橘子一类(反之亦然)。

*要试验感知机网络和苹果/橘子分类问题，可使用 Neural Network Design Demonstration Perceptron Classification*(**nnd3pc**)。

该实例阐明了感知机网络的某些特征，但对感知机并没有进行深入全面的研究。后面第 4 章到第 12 章还会对该网络及其变形进行深入讨论。这里先简要介绍一下今后要讨论的问题。

在苹果/橘子分类问题中，可以通过选择明确划分模式的判定边界用图形方式设计一个网络。但在实际问题中，如果输入空间维数较高，又将如何设计网络呢？第 4 章、第 7 章、第 10 章和第 11 章将介绍用一组反映网络行为的实例训练网络的学习算法，以解决复杂问题。 3-7

单层感知机的关键特性是它构造了一个线性判定边界对输入向量进行分类。但是如果输入类别不能用线性边界进行划分，又将如何呢？这个问题将在第 11 章讨论，其中将介绍一种能够求解任意复杂度分类问题的多层感知机。

### 3.2.3 Hamming 网络

下面将要讨论的是 Hamming 网络[Lipp87]。它是专门为求解二值模式识别问题而设计的(问题中输入向量的每个元素只能是两个可能值中的一个，这里取 -1 和 +1 两个值)。由于该网络同时采用了在第 2 章中所介绍的前馈层和递归(反馈)层，因此该网络有许多特殊的特性。标准的 Hamming 网络如图 3-5 所示。请注意：图中第一层的神经元的数目和第二层的神经元数目相同。

Hamming 网络的目标是判定哪个标准向量最接近于输入向量。判定结果由递归层的输出表示。每个标准模式均对应递归层中的一个神经元，当递归层收敛后，递归层中只有一个神经元的输出值为非 0 值，该神经元指明了哪一个标准模式与输入向量最接近。下面将对两层 Hamming 网络进行深入研究。

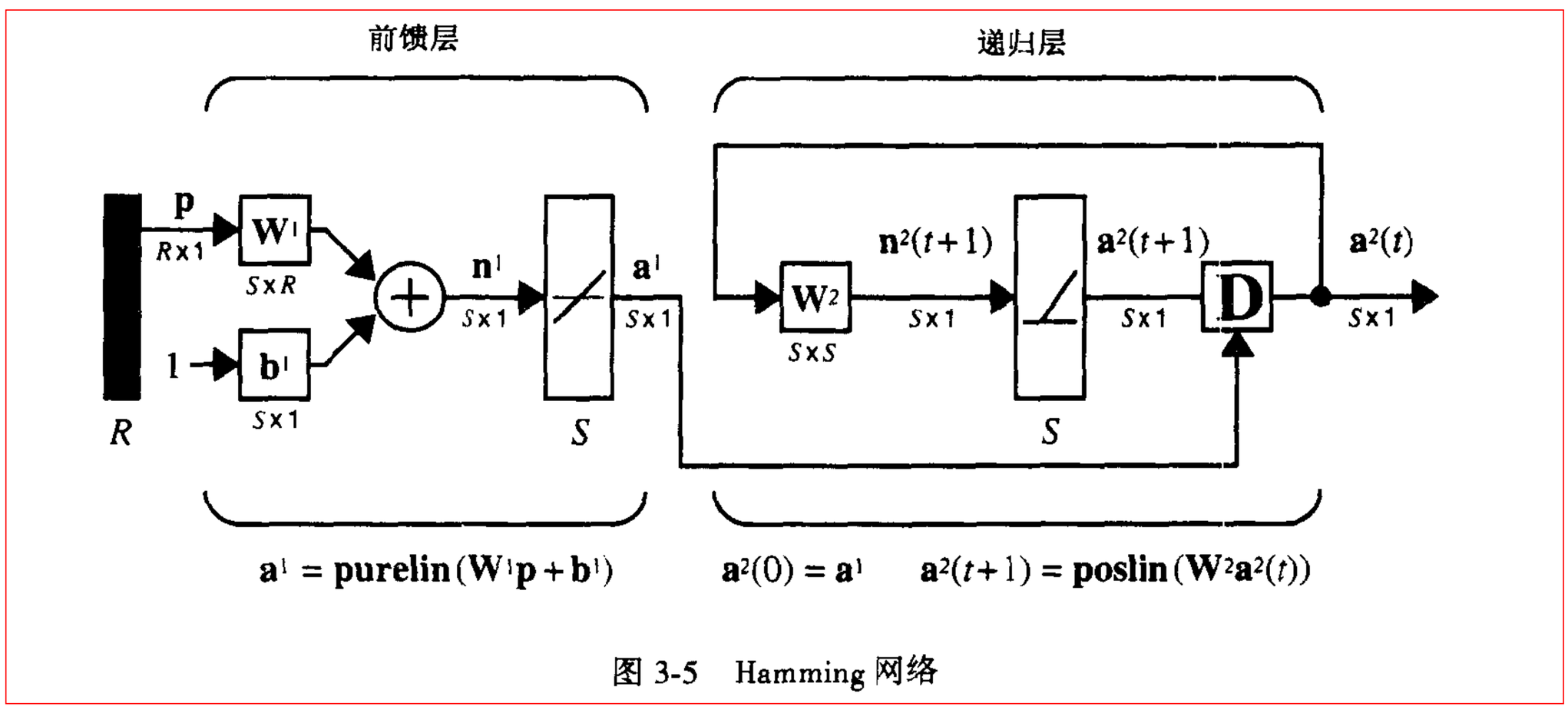



图 3-5 Hamming 网络

1. 前馈层

3-8 前馈层用于实现每个标准模式和输入模式之间的相关检测或求内积(参见式(3.17))。为了使得前馈层能够完成其功能，可以用标准模式设置其权值矩阵的行，该权值矩阵用连接矩阵 $\mathbf{W}^1$ 表示。对于苹果/橘子实例而言，有

$$\mathbf{W}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix} \tag{3.15}$$

前馈层采用的是线性传输函数，偏置值向量中的每个元素均等于 $R$。其中，$R$ 是输入向量中的元素个数。据此，可以将该实例中偏置值向量设置为

$$\mathbf{b}^1 = \begin{bmatrix} 3 \\ 3 \end{bmatrix} \tag{3.16}$$

用权值矩阵和偏置值向量的这些选择，前馈层的输出为

$$\mathbf{a}^1 = \mathbf{W}^1\mathbf{p} + \mathbf{b}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix}\mathbf{p} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1^T\mathbf{p} + 3 \\ \mathbf{p}_2^T\mathbf{p} + 3 \end{bmatrix} \tag{3.17}$$

注意：前馈层的输出等于输入和每个标准模式的内积加 $R$。对于这两个等长(范数)向

量而言，内积在两个向量指向同方向时最大，指向相反时值最小(这个概念将在第 5 章、第 8 章和第 9 章进行深入讨论)。把内积加上 $R$ 是为了保证前馈层的输出不会都是负数值，这是递归层正常操作所必需的。

之所以称该网络为 Hmming 网，是因为在前馈层中具有最大输出的神经元正好对应于与输入模式 Hamming 距离最近的标准模式(两个向量的 Hamming 距离等于其向量中不同的元素个数。请注意这只是针对于二进制向量而言的)。请读者自行验证一下前馈层的输出是否等于 2R 减去标准模式和输入模式之间的两倍 Hamming 距离。

2. *递归层*

Hamming 网的递归层就是所谓的“竞争”层。该层的神经元用前馈层的输出进行初始化，此输出指出标准模式和输入向量之间的关系。然后递归层中的神经元相互竞争以决定谁是胜利者。竞争后只有一个神经元的输出值不等于 0。竞争取胜的神经元就表示提供给网络的输入的类别（比如在我们的实例中，就是苹果和橘子两种类别）。描述竞争的等式为 3-9

$$\mathbf{a}^2(0) = \mathbf{a}^1 \qquad \text{(初始条件)} \tag{3.18}$$

和

$$\mathbf{a}^2(t+1) = \mathbf{poslin}(\mathbf{W}^2\mathbf{a}^2(t)) \tag{3.19}$$

请注意：这里的上标表示第二层，而不是 2 次幂。*poslin* 传输函数对于正值而言是线性函数，对于负值而言取值为 0。权值矩阵 $\mathbf{W}^2$ 的形式为

$$\mathbf{W}^2 = \begin{bmatrix} 1 & -\varepsilon \\ -\varepsilon & 1 \end{bmatrix} \tag{3.20}$$

其中 $\varepsilon$ 为小于 $1/(S-1)$的一个数，$S$ 为递归层的神经元的个数。(读者能够说明为什么 $\varepsilon$ 必须小于 $1/(S-1)$吗?)

递归层的每次迭代过程可以用下式表示：

$$\mathbf{a}^2(t+1) = \mathbf{poslin}\left(\begin{bmatrix} 1 & -\varepsilon \\ -\varepsilon & 1 \end{bmatrix}\mathbf{a}^2(t)\right) = \mathbf{poslin}\left(\begin{bmatrix} a_1^2(t) - \varepsilon a_2^2(t) \\ a_2^2(t) - \varepsilon a_1^2(t) \end{bmatrix}\right) \tag{3.21}$$

从上式可以看出，向量中每个元素都减去另一个元素的一部分，而减少的比例相同，均为 $\varepsilon$。由此可知，具有较大值的元素减去的量较少，而具有较小值的元素减去的量较大，这将导致元素值的大小差别进一步扩大，最终使得除了初始值最大的元素的值继续保持较大的值之外，其他元素的值将逐步变为 0。而输出值大于 0 的元素所对应的神经元便对应于以 Hamming 距离和输入模式最靠近的标准模式。

这里将再次以前面测试感知机的椭圆形橘子为例进一步说明 Hamming 网络的机理。一个椭圆形橘子可以用向量表示为：

$$\mathbf{p} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \tag{3.22}$$

前馈层的输出为： 3-10

$$\mathbf{a}^1 = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix}\begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} (1+3) \\ (-1+3) \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix} \tag{3.23}$$

这就是递归层的初始化条件。

递归层的权值矩阵由式(3.20)给出，其中 $\varepsilon = 1/2$(实际上，这里也可采用其他任何小于 1 的数)。递归层的第一次迭代得到

$$\mathbf{a}^2(1) = \mathbf{poslin}(\mathbf{W}^2\mathbf{a}^2(0)) = \begin{cases} \mathbf{poslin}\left(\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}\begin{bmatrix} 4 \\ 2 \end{bmatrix}\right) \\ \mathbf{poslin}\left(\begin{bmatrix} 3 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 3 \\ 0 \end{bmatrix} \end{cases} \tag{3.24}$$

第二次迭代结果为

$$\mathbf{a}^2(2) = \mathbf{poslin}(\mathbf{W}^2\mathbf{a}^2(1)) = \begin{cases} \mathbf{poslin}\left(\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}\begin{bmatrix} 3 \\ 0 \end{bmatrix}\right) \\ \mathbf{poslin}\left(\begin{bmatrix} 3 \\ -1.5 \end{bmatrix}\right) = \begin{bmatrix} 3 \\ 0 \end{bmatrix} \end{cases} \tag{3.25}$$

由于递归层在后面的迭代过程中得到的都是同样的结果，这表明网络已经收敛。这时只有第一个神经元的输出为非 0 值，因此选择第一个神经元所代表的标准模式(橘子)作为匹配结果($\mathbf{a}^1$ 的第一个元素是($\mathbf{p}_1^T\mathbf{p}+3$))。由于橘子标准向量和该输入模式的 Hamming 距离为 1，而苹果标准向量和该输入模式的 Hamming 距离为 2，据此可以看出该网络的识别结果是正确的。

要试验 Hamming 网络和苹果/橘子分类问题，请使用 *Neural Network Design Demonstration Hamming Classification*(**nnd3hamc**)。

有很多网络都是按 Hamming 网络的相同原理工作，也即在内积操作层(前馈层)之后紧跟一个动态竞争层。第 13 章到第 16 章将讨论这些竞争网络。这些网络又称自组织网络，它们能够根据所提供的输入调节其标准向量。

3-11

### 3.2.4 Hopfield 网络

本章最后要简单讨论的网络就是 Hopfield 网络。它是有些类似于 Hamming 网络递归层的一种递归网络，但它能有效地实现 Hamming 网络的两层所完成的工作。Hopfield 网络如图 3-6 所示。(实际上该图表示的是标准 Hopfield 网络的一种变形。使用这个模型是因其比较简单，同时也有利于阐明一些基本概念)。

这个网络利用输入向量对网络中的神经元进行初始化，然后网络不断迭代直至收敛。如果网络运行正确，那么最终的输出结果将是一个标准向量。所以，Hamming 网络是用取值不为 0 的神经元表明选择的是哪个标准模式，而 Hopfield 网络则生成一个标准模式作为其输出。

描述该网络操作的等式为

$$\mathbf{a}(0) = \mathbf{p} \tag{3.26}$$

和

$$\mathbf{a}(t+1) = \mathbf{satlins}(\mathbf{W}\mathbf{a}(t) + \mathbf{b}) \tag{3.27}$$

其中 *satlins* 为[ - 1，1]区间上的线性传输函数；当输入大于 1 时，函数输出恒为 1；当输入小于 - 1 时，函数输出恒为 - 1。

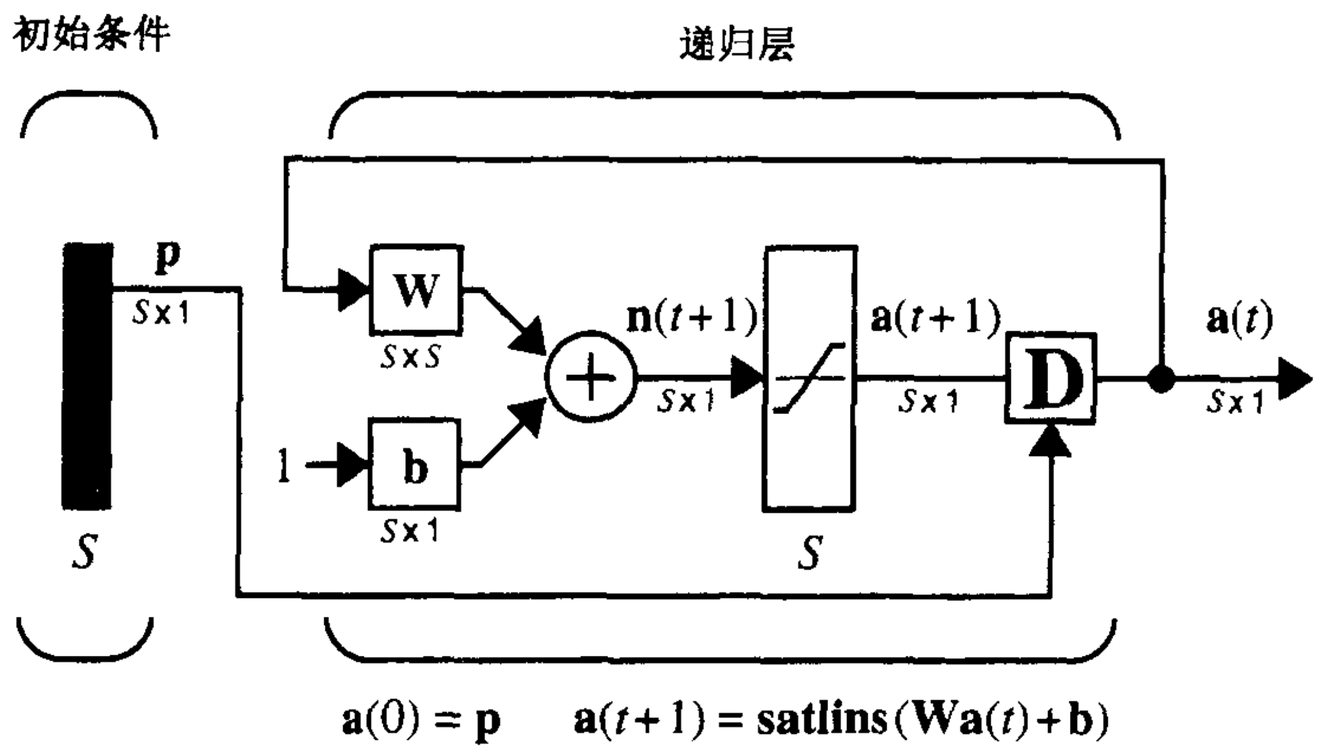



图 3-6　Hopfield 网络

Hopfield 网络的权值矩阵和偏置值向量的设置要比 Hamming 网络复杂得多，在 Hamming 网络中前馈层的权值为标准模式。本书将在第 18 章详细讨论 Hopfield 的设计过程。 3-12

为了说明该网络的工作过程，这里不妨指定一个能解决苹果/橘子识别问题的权值矩阵和偏置值矩阵。它们由下式给出：

$$\mathbf{W}=\begin{bmatrix}0.2 & 0 & 0\\ 0 & 1.2 & 0\\ 0 & 0 & 0.2\end{bmatrix},\quad \mathbf{b}=\begin{bmatrix}0.9\\ 0\\ -0.9\end{bmatrix} \tag{3.28}$$

尽管计算 Hopfield 网络的权值和偏置值的过程并不是本章要讨论的内容，但还是可以看出式(3.28)为什么可以解决苹果/橘子问题的一些特征。

这里希望网络输出要么收敛于橘子的标准模式 $\mathbf{p}_1$，要么收敛于苹果的标准模式 $\mathbf{p}_2$。两个模式的标准向量的第一个元素均为 1，第三个元素均为 -1，两者的不同之处在于第二个元素。所以，无论给网络输入什么模式，均希望输出模式的第一个元素收敛于 1，第三个元素收敛于 -1，而第二个元素要么收敛于 1，要么收敛于 -1，使之最接近输入向量的第二个元素。

用式(3.28)给出的参数，可以将 Hopfield 网络的操作等式写成：

$$\begin{aligned}a_1(t+1) &= satlins(0.2a_1(t)+0.9)\\ a_2(t+1) &= satlins(1.2a_2(t))\\ a_3(t+1) &= satlins(0.2a_3(t)-0.9)\end{aligned} \tag{3.29}$$

无论 $a_i(0)$ 的初始值是多少，第一个元素的值将不断增加直到最后的值为 1，第三个元素将不断减少直到最后的值为 -1。第二个元素乘上一个大于 1 的数。所以如果第二个元素的初始值为负数，它将收敛于 -1；反之，如果初始值为正数，它将收敛于 1。

值得注意的是：(**W**，**b**)的取值并不是惟一的。读者可以试试其他的取值，看看是否能够完成预期的工作要求。

这里再次用椭圆形的橘子实例对 Hopfield 网络进行测试。前三个迭代过程结束时，Hopfield 网络的输出分别为： 3-13

$$\mathbf{a}(0)=\begin{bmatrix}-1\\ -1\\ -1\end{bmatrix},\quad \mathbf{a}(1)=\begin{bmatrix}0.7\\ -1\\ -1\end{bmatrix},\quad \mathbf{a}(2)=\begin{bmatrix}1\\ -1\\ -1\end{bmatrix},\quad \mathbf{a}(3)=\begin{bmatrix}1\\ -1\\ -1\end{bmatrix} \tag{3.30}$$

尽管和 Hamming 网络和感知机网络的工作方式不同，Hopfield 网络同样也最终收敛于橘子模式。感知机只有一个取值为 -1(橘子)或 1(苹果)的输出。Hamming 网络中也只有一个取值非 0 的神经元表示哪个标准模式为最佳匹配。如果第一个神经元取非 0 值，表示输入模式属于橘子类别；如果第二个神经元取非 0 值，则表示输入模式属于苹果类别。在 Hopfield 网络中，标准模式本身将会出现在网络的输出上。

要试验 Hopfield 网络和苹果/橘子分类问题，请使用 *Neural Network Design Demonstration Hopfield Classification*(**nnd3hopc**)。

尽管本章介绍了一些神经网络，请不要就此止步。实际上，还有很多问题有待深入讨论。例如：如何知道网络最终一定会收敛？递归网络有时也可能出现振荡情况和混沌行为。同样，这里也没有讨论设计权值矩阵和偏置值向量的一般方法。所有这些问题都将在第 17 章和第 18 章中讨论。

3-14

## 3.3 结束语

本章介绍的三种网络展示了全书将要讨论的结构的许多共同特性。

感知机仅仅是将在第 4 章、第 7 章、第 11 章和第 12 章中讨论的前馈网络的一个实例。在这些前馈网络中，网络的输出直接根据网络的输入计算出来，并不涉及到反馈。前馈网络可以用于诸如苹果/橘子区分之类的模式识别问题，也可用于函数拟合问题(请参见第 11 章)。在自适应滤波(参见第 10 章)和自动控制等领域均有函数拟合的应用场合。

这里以 Hamming 网络为代表的竞争网络有两个主要特点。其一是它们计算出已存储的标准模式和输入模式之间的距离测度。其二是通过竞争决定哪一个神经元表示的标准模式最接近于输入模式。在第 14 章到第 16 章所讨论的竞争网络中，当给网络提供新的输入时，要对标准模式进行调整。这种自适应网络学习如何将输入聚类到不同的类别。

诸如 Hopfield 之类的递归网络最初是从统计力学的研究发展而来的。它们主要用于联想存储中，其存储的数据能由相关的输入数据回忆出来，而无需用一个地址对其访问。另外，这些网络也可用于解决许多优化问题。第 17 章和第 18 章将对这些递归网络进行深入讨论。

希望本章已经激起读者对神经网络能力的好奇心，并提出了一些问题。后面各章将要回答的一些问题是：

1) 当输入较多而判定边界无法用图示方法表示的情况下，如何设计多输入感知机网络的权值和偏置值？(第 4 章和第 10 章)
2) 如果要识别的类别不是线性可分的，能否通过扩展标准感知机来解决这类问题？(第 11 章和第 12 章)
3) 当并不知道标准模式时，Hamming 网如何学习权值和偏置值？(第 14 章到第 16 章)
4) 如何确定 Hopfield 网络的权值矩阵和偏置值向量？(第 18 章)
5) 如何知道 Hopfield 网络最终是否会收敛？(第 17 章和第 18 章)

3-15

## 习题

**E3.1** 本章设计了三个不同的神经网络，根据传感器的三个测量值(外形、质地和权值)来区分橘子和苹果。现假设要区分香蕉和菠萝：

$$\mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \quad \text{（香蕉）}$$

$$\mathbf{p}_2 = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \quad \text{（菠萝）}$$

（i）试设计一个感知机来识别这两种模式。

（ii）试设计一个 Hamming 网络来识别这两种模式。

（iii）试设计一个 Hopfield 网络来识别这两种模式。

（iv）请用几个不同的输入模式来测试你所设计的网络的操作。并请讨论每种网络的优点和缺点。 3-16

# 第4章　感知机学习规则

## 4.1　目的

第3章曾提出了一个问题："当多输入神经元网络的判定边界无法用图形方式表示的情况下，如何确定权值矩阵和偏置值？"本章将介绍一种用于训练感知机网络的算法，使感知机能够学习求解分类问题。为此，这里将从介绍什么是学习规则开始，然后讨论如何设计感知机网络的学习规则。本章最后将对单层感知机网络的优点和局限性进行讨论。这些讨论将为以后各章奠定基础。



## 4.2　理论和实例

1943年，Warren McCulloch 和 Walter Pitts 最早提出了一种人工神经元模型[McPi43]。该模型的主要特点是把神经元输入信号的加权和与其阈值相比较以确定神经元的输出。如果加权和小于阈值，则该神经元的输出值为零；如果加权和大于阈值，则该神经元的输出值为1。Warren McCulloch 和 Walter Pitts 进一步证明了这些神经元网络原则上可以完成任何数学和逻辑函数的计算。与生物神经网络不同的是，由于没有找到训练这些网络的方法，所以必须设计出这些神经元网络的参数以实现特定的功能。但是，由于该模型使人们看到了生物学与数字计算机之间的某些联系，从而引起了人们的极大兴趣。

20世纪50年代末，Frank Rosenblatt 和其他几位研究人员提出了一种称为感知机的神经网络。这些网络中的神经元与 McCulloch 和 Pitts 提出的神经元模型十分相似。Rosenblatt 的主要贡献在于引入了用于训练神经网络解决模式识别问题的学习规则[Rose58]。他证明了只要求解问题的权值存在，那么其学习规则通常会收敛到正确的网络权值上。整个学习过程较为简单，而且是自动的。只要把反映网络行为的实例提交给网络，网络就能够根据实例从随机初始化的权值和偏置值开始自动地进行学习。

然而，感知机网络本身却具有其内在的局限性。在 Marvin Minsky 和 Seymour Papert 所著的《感知机》(*Perceptrons*)[MiPa69]一书中，对这些局限性进行了全面深入的分析，指出感知机网络不能实现某些基本的功能(如异或等)。该书的结论曾一度导致神经网络研究陷入低潮。直到80年代，改进的(多层)感知机网络和相应学习规则的提出才为克服这些局限性开辟了新的途径，并重新唤起人们对神经网络研究的兴趣。本书将在第11章和第12章中讨论多层感知机及其学习规则。

当前，人们仍然认为感知机网络是一种重要的神经网络。对于某些应用问题而言，这种神经网络仍不失为一种快速可靠的求解方法。另外，对感知机网络行为的理解将会为理解更加复杂的神经网络奠定良好基础。因此，这里讨论感知机网络及其联想学习规则是十分必要的。

下面首先将对学习规则的概念给出明确定义，然后解释感知机网络及其学习规则，并讨论感知机网络的局限性。

### 4.2.1　学习规则

**学习规则**　在开始讨论感知机的学习规则之前，首先来讨论一般的学习规则。所谓学习规则就是修改神经网络的权值和偏置值的方法和过程(也称这种过程是训练算法)。学习规则的目的是为了训练网络来完成某些工作。现在有很多类型的神经网络学习规则。大致可以将其分为三大类：有监督学习、无监督学习和增强(或分级)学习。 4-2

**有监督的学习　训练集　目标**　在有监督学习当中，学习规则由一组描述网络行为的实例集合(训练集)给出：

$$\{\mathbf{p}_1,\mathbf{t}_1\},\{\mathbf{p}_2,\mathbf{t}_2\},\cdots,\{\mathbf{p}_Q,\mathbf{t}_Q\} \tag{4.1}$$

其中，$\mathbf{p}_q$ 为网络的输入，$\mathbf{t}_q$ 为相应的正确(目标)输出。当输入作用到网络时，网络的实际输出与目标相比较，然后学习规则调整网络的权值和偏置值，从而使网络的实际输出越来越接近于目标输出。感知机的学习规则就属于这一类有监督学习。本书还将在第 7 章到第 12 章继续研究有监督学习算法。

**增强学习**　增强学习与有监督的学习类似，只是它并不像有监督的学习一样为每一个输入提供相应的目标输出，而是仅仅给出一个级别。这个级别(或评分)是对网络在某些输入序列上的性能测度。当前这种类型的学习要比有监督的学习少见。看起来它最为适合控制系统应用领域(请见[BaSu83]，[WhSo92])。

**无监督的学习**　在无监督的学习中，仅仅根据网络的输入调整网络的权值和偏置值，它没有目标输出。乍一看这种学习似乎并不可行：不知道网络的目的是什么，还能够训练网络吗？实际上，大多数这种类型的算法都是要完成某种聚类操作，学会将输入模式分为有限的几种类型。这种功能特别适合于诸如向量量化等应用问题。本书将在第 13 章到第 16 章讨论更多的无监督学习算法。

### 4.2.2　感知机的结构

在介绍感知机的学习规则之前，首先对在第 3 章中介绍的感知机网络进一步进行研究。感知机网络的一般结构如图 4-1 所示。

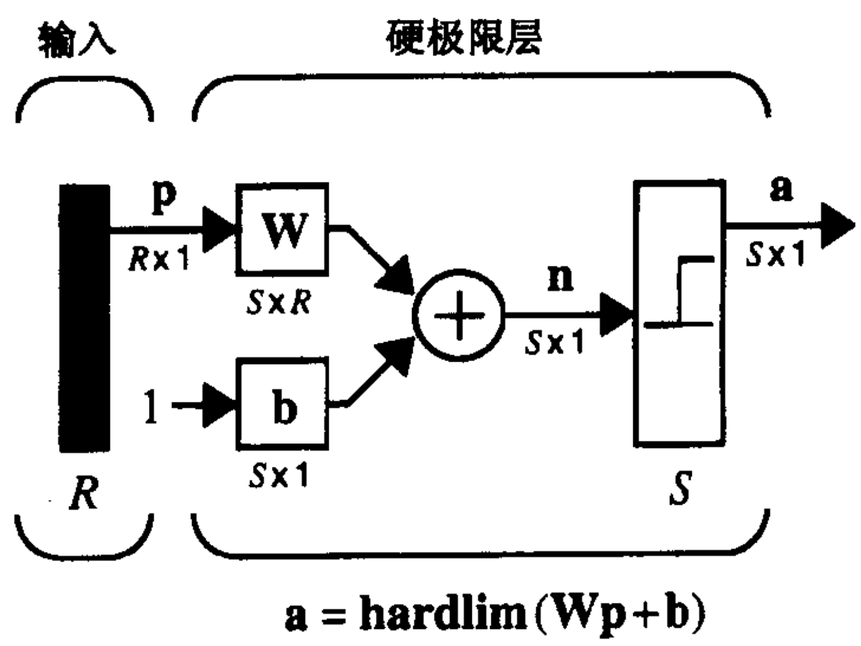



图 4-1　感知机网络

该网络的输出由下式给出：

$$\mathbf{a} = \mathbf{hardlim}(\mathbf{Wp} + \mathbf{b}) \tag{4.2}$$

(请注意：第 3 章使用的是 *hardlims* 传输函数，而不是 *hardlim* 传输函数，不过这并不
4-3 影响该网络的能力。请参见习题 E4.6。

式(4.2)在开发感知机的学习规则中十分有用，利用该公式可以方便地引用感知机网络输出中的单个元素。为此，首先考虑如下权值矩阵：

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix} \tag{4.3}$$

我们将构成 **W** 的第 $i$ 个行向量定义为：

$$_i\mathbf{w} = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,R} \end{bmatrix} \tag{4.4}$$

据此，可将权值矩阵 **W** 重写为：

$$\mathbf{W} = \begin{bmatrix} {}_1\mathbf{w}^T \\ {}_2\mathbf{w}^T \\ \vdots \\ {}_S\mathbf{w}^T \end{bmatrix} \tag{4.5}$$

4-4 这样就可以将网络输出向量的第 $i$ 个元素写成

$$a_i = hardlim(n_i) = hardlim({}_i\mathbf{w}^T\mathbf{p} + b_i) \tag{4.6}$$

由于 *hardlim* 传输函数(如图 4-2)的定义是：

$$a = hardlim(n) = \begin{cases} 1, \text{如果 } n \geq 0 \\ 0, \text{其他} \end{cases} \tag{4.7}$$

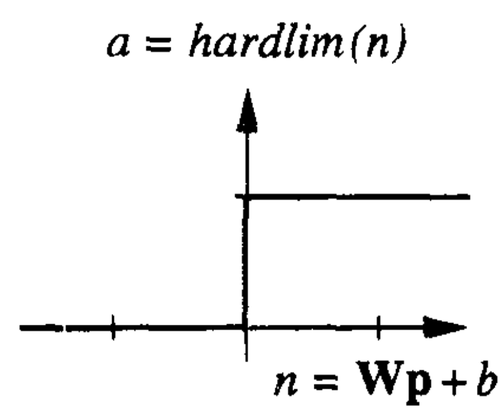



图 4-2 *hardlim* 传输函数

所以，如果权值矩阵的第 $i$ 个行向量与输入向量的内积大于等于 $-b_i$，该输出为 1，否则输出为 0。因此网络中的每个神经元将输入空间划分成两个区域。研究这些区域之间的边界是非常有用的。下面将从有两个输入的单神经元感知机开始，对此进行讨论。

### 1. 单神经元感知机

考虑如图 4-3 所示的两个输入的单神经元感知机。该网络的输出由下式所决定：

$$\begin{aligned} a &= hardlim(n) = hardlim(\mathbf{W}\mathbf{p} + b) \\ &= hardlim({}_1\mathbf{w}^T\mathbf{p} + b) = hardlim(w_{1,1}p_1 + w_{1,2}p_2 + b) \end{aligned} \tag{4.8}$$

**判定边界** 判定边界由那些使得净输入 $n$ 为零的输入向量确定：

$$n = {}_1\mathbf{w}^T\mathbf{p} + b = w_{1,1}p_1 + w_{1,2}p_2 + b = 0 \tag{4.9}$$

4-5 为了使该实例更加具体，现将权值和偏置值设置为：

$$w_{1,1} = 1, w_{1,2} = 1, b = -1 \tag{4.10}$$

那么判定边界是

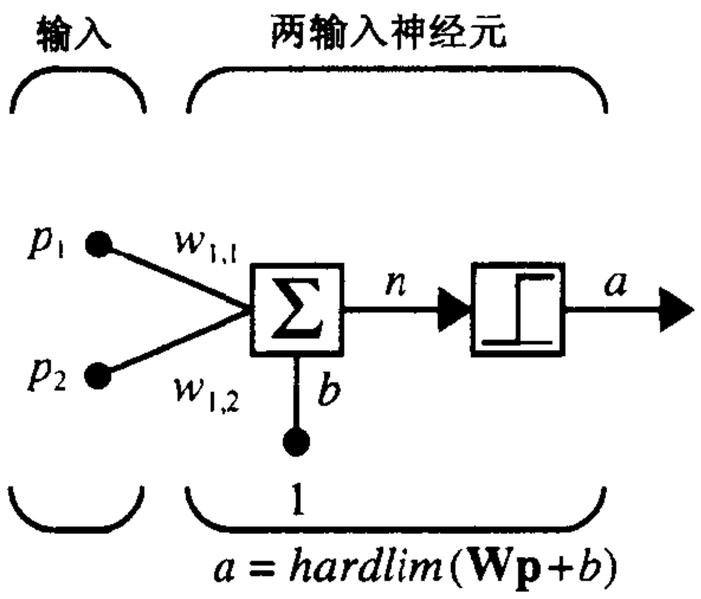



图 4-3　两输入/单输出神经元感知机

$$n = {}_1\mathbf{w}^T\mathbf{p} + b = w_{1,1}p_1 + w_{1,2}p_2 + b = p_1 + p_2 - 1 = 0 \tag{4.11}$$

式(4.11)在输入空间中定义了一条直线。该直线一侧的输入向量相应的网络输出为 0；而直线上和另一侧的输入向量相应的网络输出则为 1。为了画出这条直线，必须找到该直线穿过轴 $p_1$ 和 $p_2$ 的点。为了求该直线在轴 $p_2$ 上的截矩，令 $p_1=0$：

$$p_2 = -\frac{b}{w_{1,2}} = -\frac{-1}{1} = 1 \qquad (当\ p_1 = 0\ 时) \tag{4.12}$$

为了求该直线在轴 $p_1$ 上的截矩，令 $p_2=0$：

$$p_1 = -\frac{b}{w_{1,1}} = -\frac{-1}{1} = 1 \qquad (当\ p_2 = 0\ 时) \tag{4.13}$$

据此可得如图 4-4 所示的判定边界。

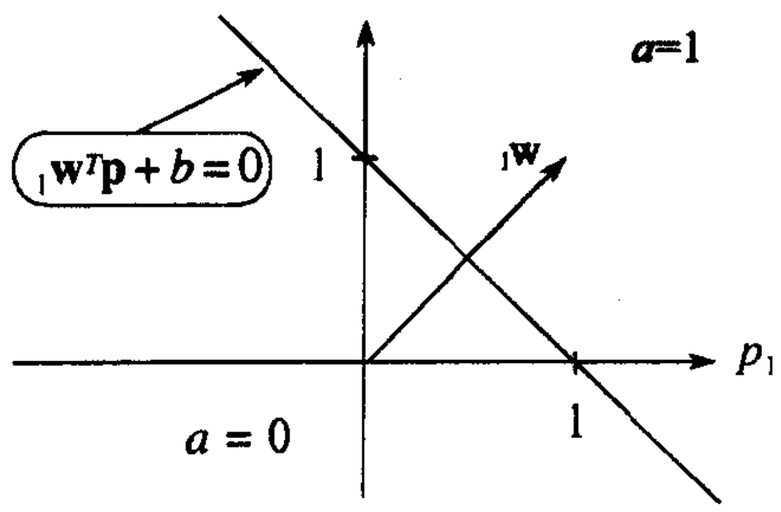



图 4-4　双输入感知机的判定边界

为了确定边界的哪一边对应的输出为 1，我们只需检测输入空间的一个点。对于输入 $\mathbf{p} = [2 \quad 0]^T$，网络的输出为

$$a = hardlim({}_1\mathbf{w}^T\mathbf{p} + b) = hardlim\left([1 \quad 1]\begin{bmatrix}2\\0\end{bmatrix} - 1\right) = 1 \tag{4.14}$$

所以，对于判定边界右上方的区域网络输出为 1。在图 4-4 中用阴影表示该区域。 4-6

另外，也可用图解的方法找到该网络相应的判定边界。首先必须注意的是该边界与 ${}_1\mathbf{w}$ 垂直(如图 4-5 所示)：

图 4-5 中的判定边界由下式定义：

$${}_1\mathbf{w}^T\mathbf{p} + b = 0 \tag{4.15}$$

图 4-5

对判定边界上的所有点而言，输入向量与权值向量的内积都是一样的。这意味着所有这些输入向量在权值向量上都有相同的投影，所以它们必须位于与权值向量正交的一条直线上(第 5 章将详细讨论这一概念)。另外，图 4-4 阴影区域中的任意输入向量都有大于 $-b$ 的内积，而无阴影区域中的输入向量则有小于 $-b$ 的内积。因此，权值向量 $_1\mathbf{w}$ 将总是指向神经元输出为 1 的区域。

一旦选择好具有正确角度指向的权值向量，就可以选择判定边界上满足式(4.15)的点来计算偏置值。

下面将运用上述一些概念设计出能够实现“与门”逻辑功能的感知机网络。与门的输入/目标对为：

$$\left\{\mathbf{p}_1=\begin{bmatrix}0\\0\end{bmatrix},t_1=0\right\}\left\{\mathbf{p}_2=\begin{bmatrix}0\\1\end{bmatrix},t_2=0\right\}\left\{\mathbf{p}_3=\begin{bmatrix}1\\0\end{bmatrix},t_3=0\right\}\left\{\mathbf{p}_4=\begin{bmatrix}1\\1\end{bmatrix},t_4=1\right\}$$

该问题可以用图 4-6 的方式进行描述。该图依据输入向量的目标值显示输入空间。目标值为 1 的输入向量用黑色圆圈●表示，而目标值为 0 的输入向量用空心圆圈○表示。

设计的第一步是选择一个判定边界。我们希望有一条直线将黑色圆圈和空心圆圈分隔在两个区域。能够实现这种划分的线有无穷条。不过似乎较为合理的选择是直线刚好处于这两类输入的正中(如图 4-7 所示)。

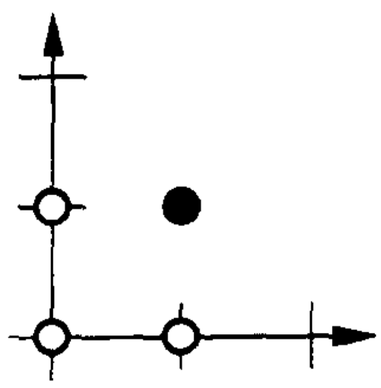


图 4-6

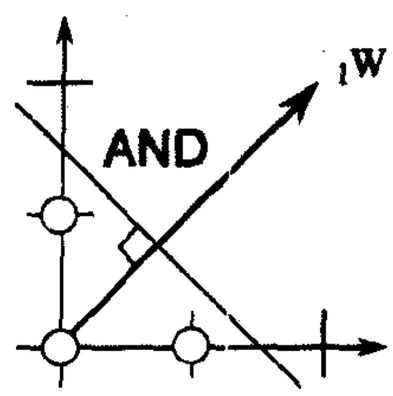



图 4-7

下面将选择一个与判定边界垂直的权值向量。由于该权值向量可以是任意长度的向量，它同样有无数可能的选择。这里选择

$$_1\mathbf{w}=\begin{bmatrix}2\\2\end{bmatrix} \tag{4.16}$$

4-7 (如图 4-7 所示)。

最后，为了求解偏置值 $b$，可以从判定边界上选取一个满足式(4.15)的点。如果选择 $\mathbf{p}=[1.5\quad 0]^T$，代入式(4.15)，有：

$$_1\mathbf{w}^T\mathbf{p}+b=[2\quad 2]\begin{bmatrix}1.5\\0\end{bmatrix}+b=3+b=0\quad\Rightarrow\quad b=-3 \tag{4.17}$$

现在可以通过选择上述的输入/目标对来对网络进行测试。如果选择 $\mathbf{p}_2$ 作为网络的输入，则输出为

$$a = hardlim({}_1\mathbf{w}^T\mathbf{p}_2 + b) = hardlim\left(\begin{bmatrix}-2 & 2\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} - 3\right) \tag{4.18}$$
$$= hardlim(-1) = 0$$

可以看出，网络的实际输出等于目标输出 $t_2$。请读者自行验证，该网络对所有的输入都能够进行正确的分类。

验证判定边界问题可使用 *Neural Network Design Demonstration Decision Boundaries*(**nnd4db**)。

2. 多神经元感知机

对于如图 4-1 所示的多神经元感知机而言，每个神经元都有一个判定边界。第 $i$ 个神经元的判定边界定义为

$$ {}_i\mathbf{w}^T\mathbf{p} + b_i = 0 \tag{4.19}$$

由于单神经元感知机的输出只能为 0 或 1，所以它可以将输入向量分为两类。而多神经元感知机则可以将输入分为许多类，每一类都由不同的输出向量来表示。由于输出向量的每个元素可以取值 0 或 1，所以共有 $2^S$ 种可能的类别，其中 $S$ 是多神经元感知机中神经元的数目。

### 4.2.3　感知机学习规则

至此我们已经考察了感知机网络的性能，从现在开始将讨论感知机的学习规则。由于其学习规则是有监督训练的一个实例，所以这里学习规则将提供一组能够正确反映网络行为的实例：

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \cdots, \{\mathbf{p}_Q, \mathbf{t}_Q\} \tag{4.20}$$

4-8

其中 $\mathbf{p}_q$ 是网络的输入，$\mathbf{t}_q$ 是该输入相应的目标输出。当每个输入作用到网络上时，网络的实际输出与目标相比较。然后学习规则调整该网络的权值和偏置值，使得网络的实际输出进一步靠近目标输出。

1. 测试问题

在讨论感知机学习规则中，首先将给出一个简单的测试实例，并对一些可能的学习规则进行测试，以使读者初步了解这些学习规则的工作机理。在该测试问题中，输入/目标对为：

$$\left\{\mathbf{p}_1 = \begin{bmatrix}1\\2\end{bmatrix}, t_1 = 1\right\}, \left\{\mathbf{p}_2 = \begin{bmatrix}-1\\2\end{bmatrix}, t_2 = 0\right\}, \left\{\mathbf{p}_3 = \begin{bmatrix}0\\-1\end{bmatrix}, t_3 = 0\right\}$$

此问题可以用图 4-8 说明，图中目标输出为 0 的两个输入向量用空心圆圈○表示，目标输出为 1 的输入向量用黑色圆圈●表示。从图中可以看出该问题实际上是一个非常简单的问题，通过一定的观察就可以得到问题的解。但是这种简单性能够帮助读者对感知机学习规则的基本概念有一个直观的理解。

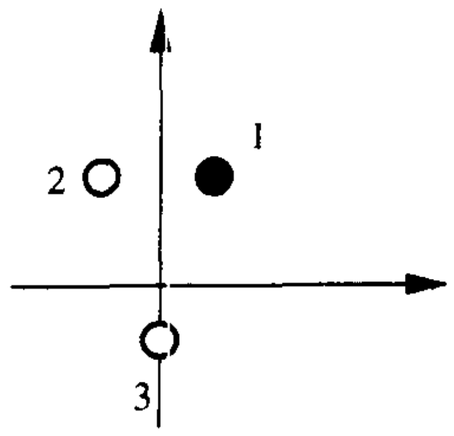



图 4-8

此问题相应的网络应该有两个输入和一个输出。为了简化其学习规则的开发，这里首先采用一种没有偏置值的网络。于是网络只需调整两个参数 $w_{1,1}$ 和 $w_{1,2}$(如图 4-9 所示)。

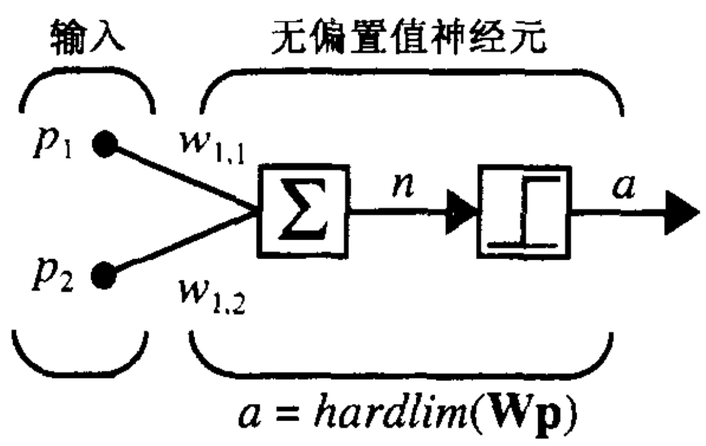



图 4-9 测试问题的网络

由于在网络中去掉了偏置值，所以网络的判定边界必定穿过坐标轴的原点(如图 4-10 所示)。为了保证简化后的网络仍然能够解决上面所给出的测试问题，这里必须找到一条判定边界将向量 $\mathbf{p}_1$ 同 $\mathbf{p}_2$，$\mathbf{p}_3$ 分开。从图中可以看出实际上有无数条可供选择的判定边界。

4-9

图 4-11 给出了这些判定边界相应的权值向量(记住权值向量与判定边界垂直)。我们希望学习规则能够找到指向这些方向中的一个权值向量。请注意：权值向量的长度无关紧要，重要的是它的方向。

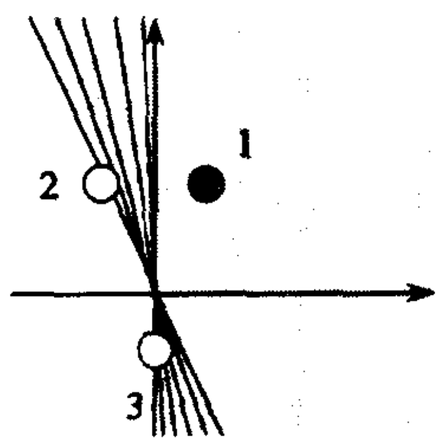



图 4-10

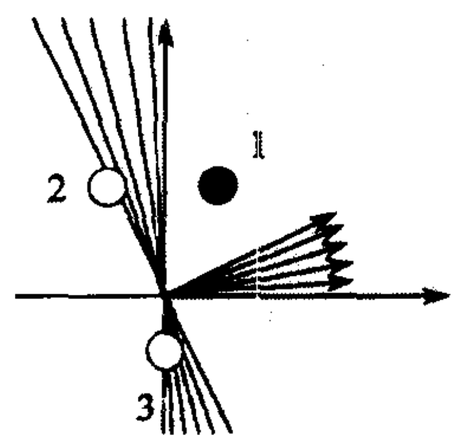



图 4-11

2. 学习规则的构造

在训练开始时，为网络的参数赋一些初始值。由于这里要训练的是一个两输入/单输出的无偏置值网络，所以仅需对其两个权值的进行初始化。这里将 ${}_1\mathbf{w}$ 的两个元素设置为如下两个随机生成的数：

$$ {}_1\mathbf{w}^T = [1.0 \quad -0.8] \tag{4.21} $$

现在将输入向量提供给网络。开始用 $\mathbf{p}_1$ 送入：

$$ \begin{aligned} a &= hardlim({}_1\mathbf{w}^T\mathbf{p}_1) = hardlim\left([1.0 \quad -0.8]\begin{bmatrix}1\\2\end{bmatrix}\right) \\ &= hardlim(-0.6) = 0 \end{aligned} \tag{4.22} $$

网络没有返回正确的值。该网络当前的实际输出为 0，而相应的目标值 $t_1$ 却为 1。

参考图 4-12 可以看出判决边界初始的权值向量导致了对向量 $\mathbf{p}_1$ 错误分类的判决边界。我们需要调整权值向量，使它更多地指向 $\mathbf{p}_1$，以便在后面更有可能得到正确的分类结果。

一种调整方法是令 ${}_1\mathbf{w}$ 等于 $\mathbf{p}_1$。这种简单的处理方法的确能够保证问题可以得到正确的分类结果。然而非常容易构造出一个并不能通过这种简单处理方法求解的问题。图 4-13 就给出了这样一个实例，在图中，如果令权值向量直接指向两个输出值为 1 的输入向量中的一个，那么权值向量并不是问题的正确解。如果每次都令 ${}_1\mathbf{w} = \mathbf{p}$，那么这两个输入向量中必有一个被错误划分，于是网络权值的求解过程将前后振荡，永远得不到正确的解。

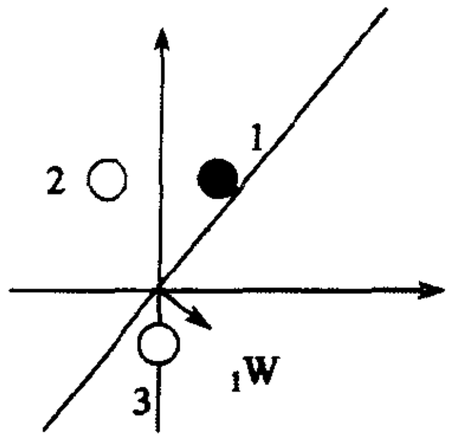



图 4-12

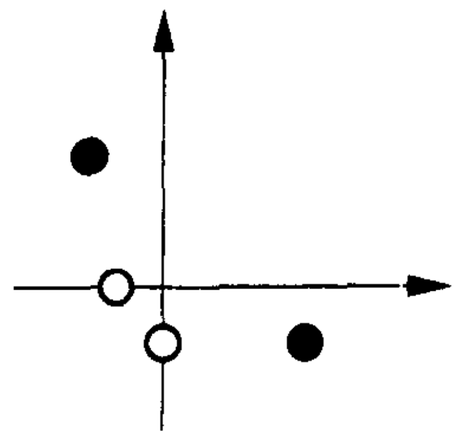


图 4-13

另一种调整方法是将 $\mathbf{p}_1$ 加到 $_1\mathbf{w}$ 上。这样会使 $_1\mathbf{w}$ 的指向更加偏向 $\mathbf{p}_1$。重复这一操作，将使 $_1\mathbf{w}$ 的指向逐步达到 $\mathbf{p}_1$ 的方向。这一规则可以表述为：

$$\text{如果 } t = 1, \text{且 } a = 0, \text{则} {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p} \tag{4.23}$$

在上述问题中应用这个规则，将会得到新的 $_1\mathbf{w}$ 值：

$$ {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}_1 = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix} \tag{4.24}$$

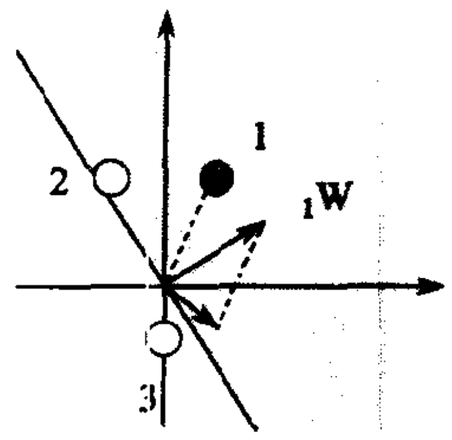



图 4-14

4-10

此操作如图 4-14 所示。

现在考虑另一个输入向量，并继续对权值进行调整。不断重复这一过程，直到所有输入向量被正确分类。

设下一个输入向量是 $\mathbf{p}_2$。当它被送入该网络后，有

$$\begin{aligned} a &= hardlim({}_1\mathbf{w}^T\mathbf{p}_2) = hardlim\left([2.0 \quad 1.2]\begin{bmatrix} -1 \\ 2 \end{bmatrix}\right) \\ &= hardlim(0.4) = 1 \end{aligned} \tag{4.25}$$

$\mathbf{p}_2$ 的目标值 $t_2$ 等于 0，而该网络的实际输出 $a$ 是 1。所以一个属于类 0 的向量被错误划分为类 1 了。

既然现在的目的是将 $_1\mathbf{w}$ 从输入向量所指的方向移开，因此可以将式(4.23)中的加法变为减法

$$\text{如果 } t = 0, \text{且 } a = 1, \text{则} {}_1\mathbf{w}^{neu} = {}_1\mathbf{w}^{old} - \mathbf{p} \tag{4.26}$$

如果在测试问题中应用该规则，可求出

$$ {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}_2 = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix} - \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix} \tag{4.27}$$

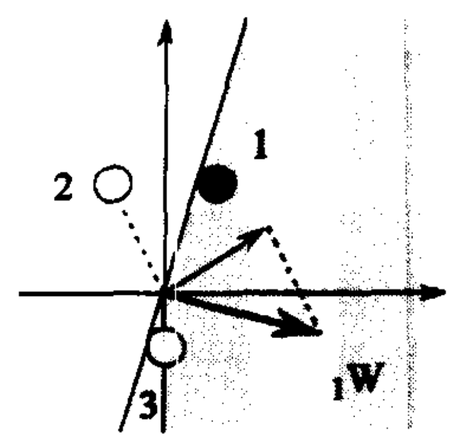



图 4-15

结果如图 4-15 所示。

现在将第三个输入向量 $\mathbf{p}_3$ 送入该网络：

$$\begin{aligned} a &= hardlim({}_1\mathbf{w}^T\mathbf{p}_3) = hardlim\left([3.0 \quad -0.8]\begin{bmatrix} 0 \\ -1 \end{bmatrix}\right) \\ &= hardlim(0.8) = 1 \end{aligned} \tag{4.28}$$

可以看出，这里 $_1\mathbf{w}$ 所形成的判定边界也错误划分了 $\mathbf{p}_3$。在这种情况下，前面已经有了相应的处理规则。所以，按照式(4.26)对 $_1\mathbf{w}$ 进行修正：

$$ {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}_3 = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 3.0 \\ 0.2 \end{bmatrix} \tag{4.29}$$

4-11

如图 4-16 表明该感知机最终可以对上述三个输入向量进行正确的分类。如果将上述任意输入向量送入神经元，感知机将输出输入向量的正确分类。

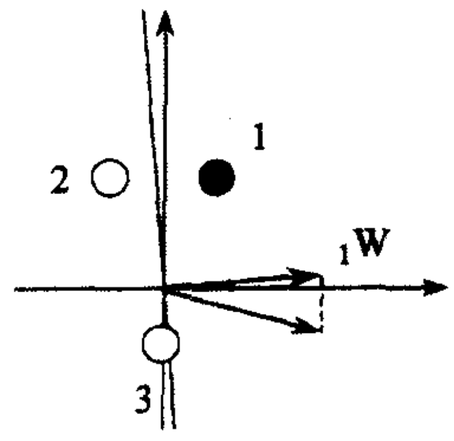



图 4-16

据此，可以得到第三条也是最后一条规则：如果感知机能够正确工作，则不用改变权值向量：

$$\text{如果 } t = a, \text{则} {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} \tag{4.30}$$

下面是涵盖了实际输出值和目标输出值所有可能组合的三条规则：

$$\begin{aligned} &\text{如果 } t = 1, \text{且 } a = 0, \text{则} {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p} \\ &\text{如果 } t = 0, \text{且 } a = 1, \text{则} {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p} \\ &\text{如果 } t = a, \text{则} {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} \end{aligned} \tag{4.31}$$

3. 统一的学习规则

式(4.31)中的三条规则可以统一表示为一个表达式。首先将感知机的误差定义为一个新的变量 $e$：

$$e = t - a \tag{4.32}$$

现在可将式(4.31)中的三条规则重写为：

$$\begin{aligned} &\text{如果 } e = 1, \text{则} {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p} \\ &\text{如果 } e = -1, \text{则} {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p} \\ &\text{如果 } e = 0, \text{则} {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} \end{aligned} \tag{4.33}$$

仔细观察式(4.33)中的前两条规则，不难发现 $\mathbf{p}$ 的符号和误差 $e$ 的符号一致。另外，在第三条规则中，由于 $e=0$，所以 $\mathbf{p}$ 没有出现。所以可以将上述三条规则统一成一个表达式：

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + e\mathbf{p} = {}_1\mathbf{w}^{old} + (t - a)\mathbf{p} \tag{4.34}$$

此规则可扩展到偏置值的训练过程中：可以将偏置值看作是一个输入总是为 1 的权值即可。于是可以将式(4.34)中的 $\mathbf{p}$ 用偏置值的输入 1 替换，得到感知机的偏置值学习规则：

4-12

$$b^{new} = b^{old} + e \tag{4.35}$$

4. 多神经元感知机的训练

由式(4.34)和式(4.35)给出的感知机规则，修改单神经元感知机的权值向量。我们能把这个规则按照如下方法推广到如图 4-1 所示的多神经元感知机。权值矩阵的第 $i$ 行用下式进行修改：

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + e_i\mathbf{p} \tag{4.36}$$

而偏置向量的第 $i$ 个元素则按下式进行修改：

$$b_i^{new} = b_i^{old} + e_i \tag{4.37}$$

**感知机规则** 感知机的学习规则可以方便地用矩阵符号表示为：

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{e}\mathbf{p}^T \tag{4.38}$$

和

$$\mathbf{b}^{new} = \mathbf{b}^{old} + \mathbf{e} \tag{4.39}$$

为了验证感知机的学习规则，再次考虑第 3 章中的苹果/橘子识别问题。其输入/输出原型向量为：

$$\left\{\mathbf{p}_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, t_1 = [0]\right\} \quad \left\{\mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, t_2 = [1]\right\} \tag{4.40}$$

(请注意：这里橘子模式 $\mathbf{p}_1$ 的目标输出用 0 表示，而不是用第 3 章中所用的 -1 表示。这是因为本章使用的是 *hardlim* 传输函数，而不是 *hardlims* 传输函数。)

通常，将权值和偏置值初始化为较小的随机数。假设这里的初始权值矩阵和偏置值分别为：

$$\mathbf{W} = [0.5 \quad -1 \quad -0.5], \quad b = 0.5 \tag{4.41}$$

第一步将第一个输入向量 $\mathbf{p}_1$ 送入网络：

$$a = hardlim(\mathbf{W}\mathbf{p}_1 + b) = hardlim\left([0.5 \quad -1 \quad -0.5]\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + 0.5\right) \tag{4.42}$$

4-13

$$= hardlim(2.5) = 1$$

然后计算误差：

$$e = t_1 - a = 0 - 1 = 1 \tag{4.43}$$

权值更新为

$$\begin{aligned} \mathbf{W}^{new} &= \mathbf{W}^{old} + \mathbf{e}\mathbf{p}^T = [0.5 \quad -1 \quad -0.5] + (-1)[1 \quad -1 \quad -1] \\ &= [-0.5 \quad 0 \quad 0.5] \end{aligned} \tag{4.44}$$

偏置值更新为

$$b^{new} = b^{old} + e = 0.5 + (-1) = -0.5 \tag{4.45}$$

至此完成了第一次迭代。

该感知机学习规则的第二次迭代为：

$$a = hardlim(\mathbf{W}\mathbf{p}_2 + b) = hardlim\left([-0.5 \quad 0 \quad 0.5]\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + (-0.5)\right) \tag{4.46}$$

$$= hardlim(-0.5) = 0$$

$$e = t_2 - a = 1 - 0 = 1 \tag{4.47}$$

$$\begin{aligned} \mathbf{W}^{new} &= \mathbf{W}^{old} + \mathbf{e}\mathbf{p}^T \\ &= [-0.5 \quad 0 \quad 0.5] + (1)[1 \quad 1 \quad -1] \\ &= [0.5 \quad 1 \quad -0.5] \end{aligned} \tag{4.48}$$

$$b^{new} = b^{old} + e = -0.5 + 1 = 0.5 \tag{4.49}$$

第三次迭代重新从第一个输入向量开始：

$$a = hardlim(\mathbf{W}\mathbf{p}_1 + b) = hardlim\left([0.5 \quad 1 \quad -0.5]\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + 0.5\right) \tag{4.50}$$

$$= hardlim(0.5) = 1$$

$$e = t_1 - a = 0 - 1 = -1 \tag{4.51}$$

$$\begin{aligned} \mathbf{W}^{new} &= \mathbf{W}^{old} + \mathbf{e}\mathbf{p}^T = [0.5 \quad 1 \quad -0.5] + (-1)[1 \quad -1 \quad -1] \\ &= [-0.5 \quad 2 \quad 0.5] \end{aligned} \tag{4.52}$$

4-14

$$b^{new} = b^{old} + e = 0.5 + (-1) = -0.5 \tag{4.53}$$

如果继续迭代下去，将会发现两个输入向量都能被正确分类。算法已收敛到了一个解上。请注意：最后得到的判定边界和第 3 章中所得到的判定边界并不一样，虽然两个判定边界都可以正确区分这两个输入向量。

验证感知机学习规则可使用 *Neural Network Design Demonstration* **Perceptron Rule**(**nnd4pr**)。

### 4.2.4 收敛性证明

虽然感知机的学习规则非常简单，但它十分有效。实际上可以证明：只要权值的解存在，该规则总能收敛到实现期望分类的权值上。本节将给出如图 4-17 所示的单神经元感知机的学习规则的收敛性证明。

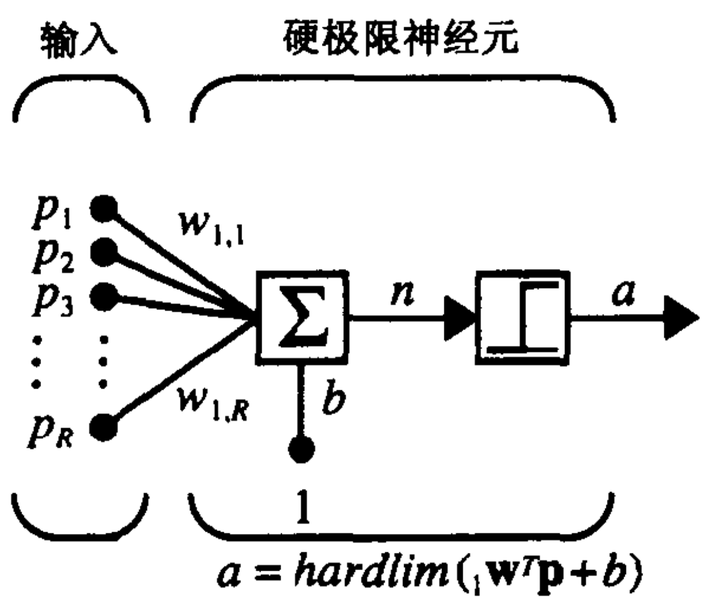



图 4-17 单神经元感知机

这个感知机的输出可由下式得到：

$$a = hardlim({}_1\mathbf{w}^T\mathbf{p} + b) \tag{4.54}$$

网络提供了正确反映网络行为的下述实例：

$$\{\mathbf{p}_1, t_1\}, \{\mathbf{p}_2, t_2\}, \cdots, \{\mathbf{p}_Q, t_Q\} \tag{4.55}$$

其中每个目标输 $t_q$ 取值 0 或 1。

1. 记号

为了便于描述证明过程，首先引入几个新的记号。这里将权值矩阵和偏置值组合为一个

4-15 向量：

$$\mathbf{x} = \begin{bmatrix} {}_1\mathbf{w} \\ b \end{bmatrix} \tag{4.56}$$

同样，在输入向量中也增加一个参数 1，以表示偏置输入：

$$\mathbf{z}_q = \begin{bmatrix} \mathbf{p}_q \\ 1 \end{bmatrix} \tag{4.57}$$

现在可将神经元的净输入表示为：

$$n = {}_1\mathbf{w}^T\mathbf{p} + b = \mathbf{x}^T\mathbf{z} \tag{4.58}$$

那么，感知机的学习规则(式(4.34)和式(4.35))可以写成

$$\mathbf{x}^{new} = \mathbf{x}^{old} + e\mathbf{z} \tag{4.59}$$

误差 $e$ 可以取 1，$-1$ 或 0。如果 $e=0$，那么权值不变；如果 $e=1$，则将输入向量和权值向量相加；如果 $e=-1$，那么权值向量减去输入向量。如果只考虑权值向量发生改变的哪些迭代，则该学习规则变为

$$\mathbf{x}(k) = \mathbf{x}(k-1) + \mathbf{z}'(k-1) \tag{4.60}$$

其中 $\mathbf{z}'(k-1)$是如下集合中的一个元素：

$$\{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_Q, -\mathbf{z}_1, -\mathbf{z}_2, \cdots, -\mathbf{z}_Q\} \tag{4.61}$$

现假设存在对所有 $Q$ 个输入向量进行正确分类的权值向量，并将这一解记为 $\mathbf{x}^*$。对该权值向量，假设

$$\text{如果 } t_q = 1, \text{那么 } \mathbf{x}^{*T}\mathbf{z}_q > \delta > 0 \tag{4.62}$$

以及

$$\text{如果 } t_q = 0, \text{那么 } \mathbf{x}^{*T}\mathbf{z}_q < -\delta < 0 \tag{4.63}$$

2. 证明

下面开始证明感知机收敛定理。为此必须找出算法每一阶段权值向量长度的上界和下界。 4-16

假设算法的初始权值向量为 0，也即 $\mathbf{x}(0)=\mathbf{0}$(这并不影响到参数的普遍性)。那么，迭代 $k$ 次($k$ 次改变权值向量)后，由式(4.60)得到

$$\mathbf{x}(k) = \mathbf{z}'(0) + \mathbf{z}'(1) + \cdots + \mathbf{z}'(k-1) \tag{4.64}$$

求迭代 $k$ 次后的权值向量和最终的权值向量解 $\mathbf{x}^*$ 之间的内积，可得

$$\mathbf{x}^{*T}\mathbf{x}(k) = \mathbf{x}^{*T}\mathbf{z}'(0) + \mathbf{x}^{*T}\mathbf{z}'(1) + \cdots + \mathbf{x}^{*T}\mathbf{z}'(k-1) \tag{4.65}$$

由式(4.61)~(4.63)可知

$$\mathbf{x}^{*T}\mathbf{z}'(\mathbf{i}) > \delta \tag{4.66}$$

所以

$$\mathbf{x}^{*T}\mathbf{x}(k) > k\delta \tag{4.67}$$

由柯西-施瓦兹不等式(见[Brog91])可得

$$(\mathbf{x}^{*T}\mathbf{x}(k))^2 \leqslant \|\mathbf{x}^*\|^2 \|\mathbf{x}(k)\|^2 \tag{4.68}$$

其中

$$\|\mathbf{x}\|^2 = \mathbf{x}^T\mathbf{x} \tag{4.69}$$

如果将式(4.67)和式(4.68)相结合，则可以得到迭代 $k$ 次后权值向量长度平方的下界为：

$$\|\mathbf{x}(k)\|^2 \geqslant \frac{(\mathbf{x}^{*T}\mathbf{x}(k))^2}{\|\mathbf{x}^*\|^2} > \frac{(k\delta)^2}{\|\mathbf{x}^*\|^2} \tag{4.70}$$

下面求权值向量长度的上界。从第 $k$ 次迭代时权值向量长度的改变量入手：

$$\begin{aligned}\|\mathbf{x}(k)\|^2 &= \mathbf{x}^T(k)\mathbf{x}(k)\\ &= [\mathbf{x}(k-1) + \mathbf{z}'(k-1)]^T[\mathbf{x}(k-1) + \mathbf{z}'(k-1)]\\ &= \mathbf{x}^T(k-1)\mathbf{x}(k-1) + 2\mathbf{x}^T(k-1)\mathbf{z}'(k-1) + \mathbf{z}'^T(k-1)\mathbf{z}'(k-1)\end{aligned} \tag{4.71}$$

注意 4-17

$$\mathbf{x}^T(k-1)\mathbf{z}'(k-1) \leqslant 0 \tag{4.72}$$

因为权值向量只有在前一输入向量被错误分类时才会进行更新。因此式(4.71)可以简化为

$$\|\mathbf{x}(k)\|^2 \leq \|\mathbf{x}(k-1)\|^2 + \|\mathbf{z}'(k-1)\|^2 \tag{4.73}$$

对 $\|\mathbf{x}(k-1)\|^2$，$\|\mathbf{x}(k-2)\|^2$，…重复上述过程，可得

$$\|\mathbf{x}(k)\|^2 \leq \|\mathbf{z}'(0)\|^2 + \cdots + \|\mathbf{z}'(k-1)\|^2 \tag{4.74}$$

令 $\Pi = max\{\|\mathbf{z}'(i)\|^2\}$，该上界可简化成

$$\|\mathbf{x}(k)\|^2 \leq k\Pi \tag{4.75}$$

至此，已求出了 $k$ 次迭代时权值向量长度平方的上界(式(4.75))和下界(式(4.76))。将其合并，求得

$$k\Pi \geq \|\mathbf{x}(k)\|^2 > \frac{(k\delta)^2}{\|\mathbf{x}^*\|^2} \text{ 或 } k < \frac{\Pi\|\mathbf{x}^*\|^2}{\delta^2} \tag{4.76}$$

由于 $k$ 有上界意味着权值的改变次数是有限的，所以感知机的学习规则将在有限次迭代后收敛。

迭代的最大次数(权值向量的改变次数)与 $\delta^2$ 成反比关系。该参数是输入模式与判定边界的解靠近程度的一种测度。这意味着，如果输入向量越靠近判定边界，就越难将它们分开，就要迭代更多次才能使算法收敛。

请注意该证明是建立在下面三条关键假设基础之上的：

1) 问题的解存在，也即满足式(4.66)。

2) 仅在输入向量被错误分类时才改变权值，也即满足式(4.72)。

3) 输入向量长度的上界 Π 存在。

由于证明的一般性，所以感知机学习规则的许多变形同样也可以证明是收敛的(参考习题 E4.9)。

### 3. 局限性

4-18 只要问题的解存在，那么感知机学习规则就一定能够在有限步数内收敛到问题的一个解。这不禁又提出了一个新的重要问题：感知机能够求解哪些问题？前面已经说明单神经元感知机可将输入空间分为两个区域，区域之间的判定边界可以由下式定义：

$$_1\mathbf{w}^T\mathbf{p} + b = 0 \tag{4.77}$$

**线性可分性** 这是一个线性边界(超平面)，因而感知机可以对那些能够被线性边界分开的输入向量进行分类。这样的向量称为是线性可分的。前面 4.2.2 节逻辑与门实例就是一个二维线性可分的问题，第 3 章中的橘子/苹果识别问题则是一个三维线性可分的实例。

然而，许多问题并非是线性可分的。典型的实例就是 XOR 门，XOR 门的输入/目标对是

$$\left\{\mathbf{p}_1 = \begin{bmatrix}0\\0\end{bmatrix}, t_1 = 0\right\}\left\{\mathbf{p}_2 = \begin{bmatrix}0\\1\end{bmatrix}, t_2 = 1\right\}\left\{\mathbf{p}_3 = \begin{bmatrix}1\\0\end{bmatrix}, t_3 = 1\right\}\left\{\mathbf{p}_4 = \begin{bmatrix}1\\1\end{bmatrix}, t_4 = 0\right\}$$

此问题可以用图 4-18 中的最左边的图来表示，该图同时还给出了另外两个线性不可分问题。试在图 4-18 中所有目标为 0 的向量和所有目标为 1 的向量之间画一条直线。

基本的感知机是不能解决这样简单问题的。在某种程度上来说，这种情况导致了 20 世纪 70 年代人们对神经网络研究兴趣的减退。Rosenblatt 也曾研究过更加复杂的网络，他觉得复杂的网络能够克服基本感知机的局限性，但是他未能将感知机学习规则有效地扩展到这样复杂的网络中。第 11 章将介绍能够求解任意分类问题的多层感知机，以及能用于训练多层感知机的反传算法。

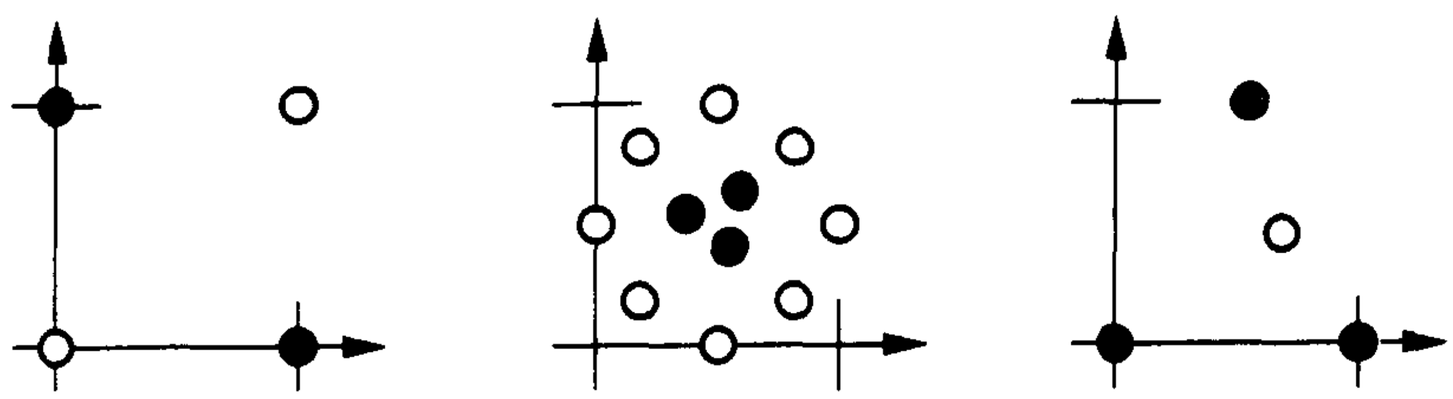
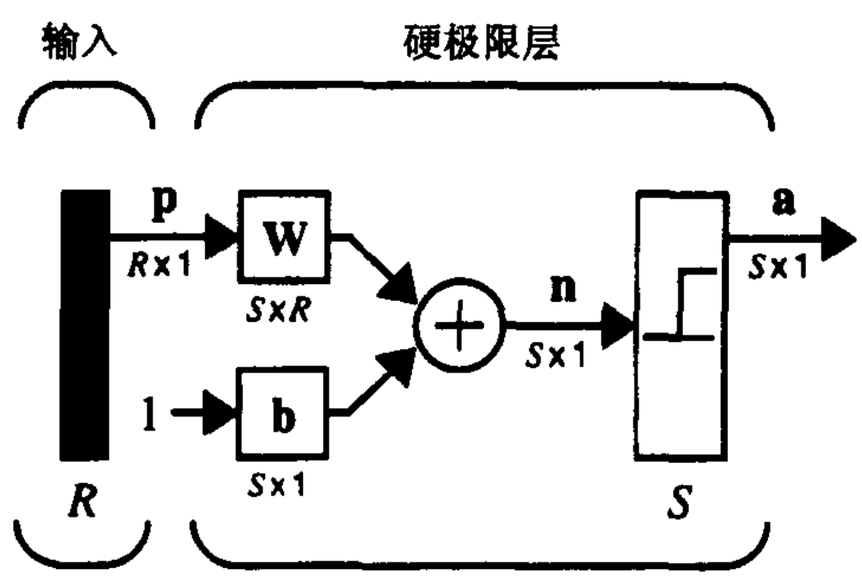


图 4-18　线性不可分问题

4-19

## 4.3　小结

**感知机的结构**




$$\mathbf{a} = \mathbf{hardlim}(\mathbf{Wp}+\mathbf{b})$$

$$\mathbf{a} = \mathbf{hardlim}(\mathbf{Wp} + \mathbf{b}) \qquad \mathbf{W} = \begin{bmatrix} {}_1\mathbf{w}^T \\ {}_2\mathbf{w}^T \\ \vdots \\ {}_S\mathbf{w}^T \end{bmatrix}$$

$$a_i = hardlim(n_i) = hardlim({}_i\mathbf{w}^T\mathbf{p} + b_i)$$

**判定边界**

$${}_i\mathbf{w}^T\mathbf{p} + b_i = 0$$

判定边界总与权值向量垂直。单层感知机只能对线性可分的向量进行分类。

**感知机学习规则**

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{e}\mathbf{p}^T$$
$$\mathbf{b}^{new} = \mathbf{b}^{old} + \mathbf{e}$$

其中 $\mathbf{e} = \mathbf{t} - \mathbf{a}$。

4-20

## 4.4　例题

**P4.1** 请画出图 4-19 中三个简单分类问题的判定边界。求相应于判定边界的权值和偏置值。

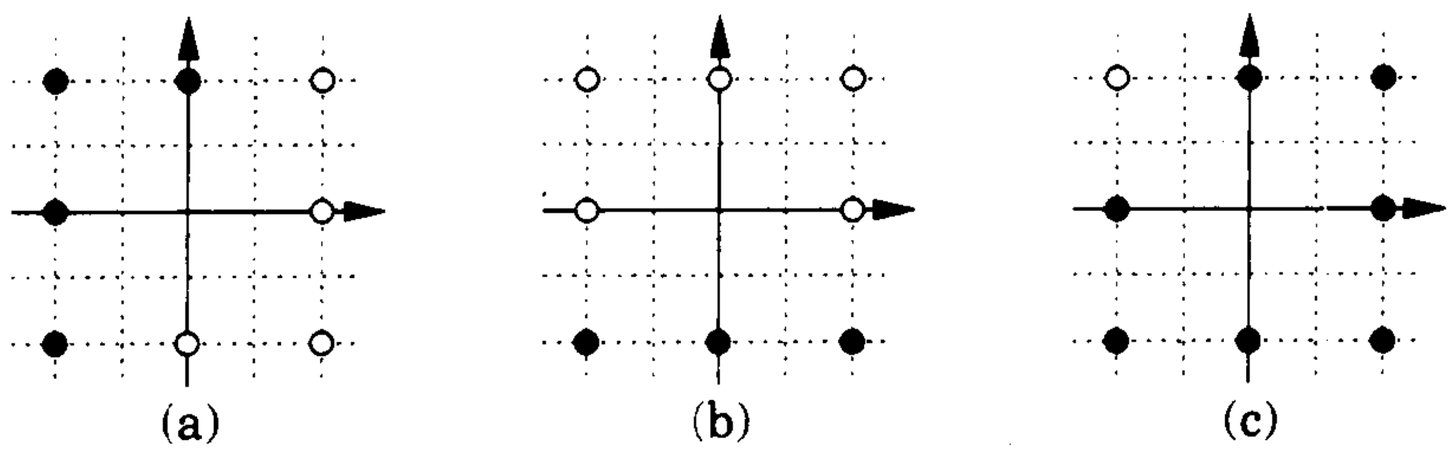
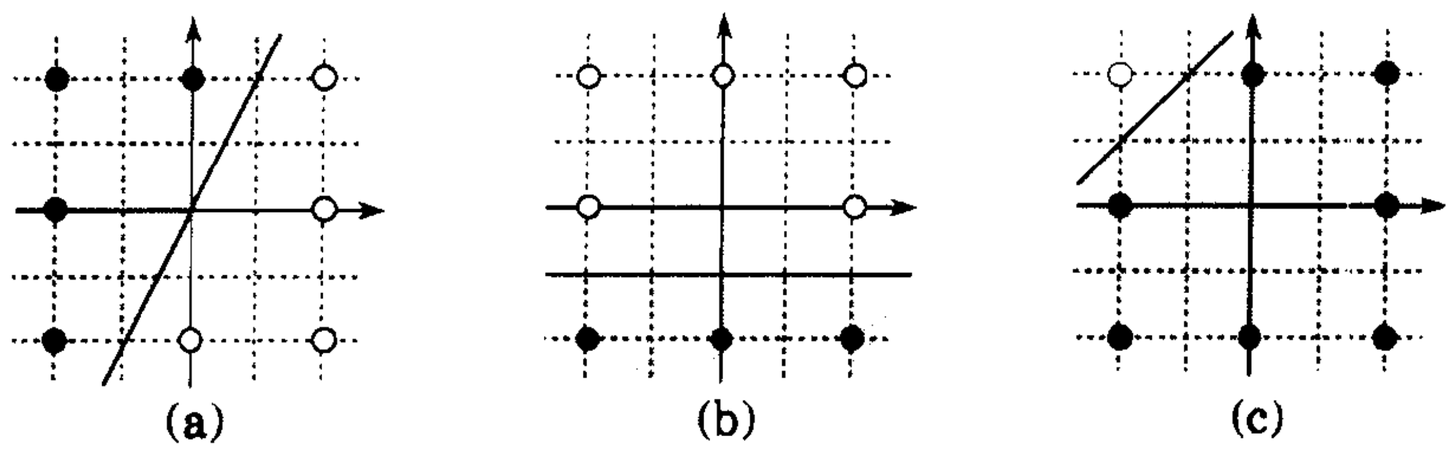


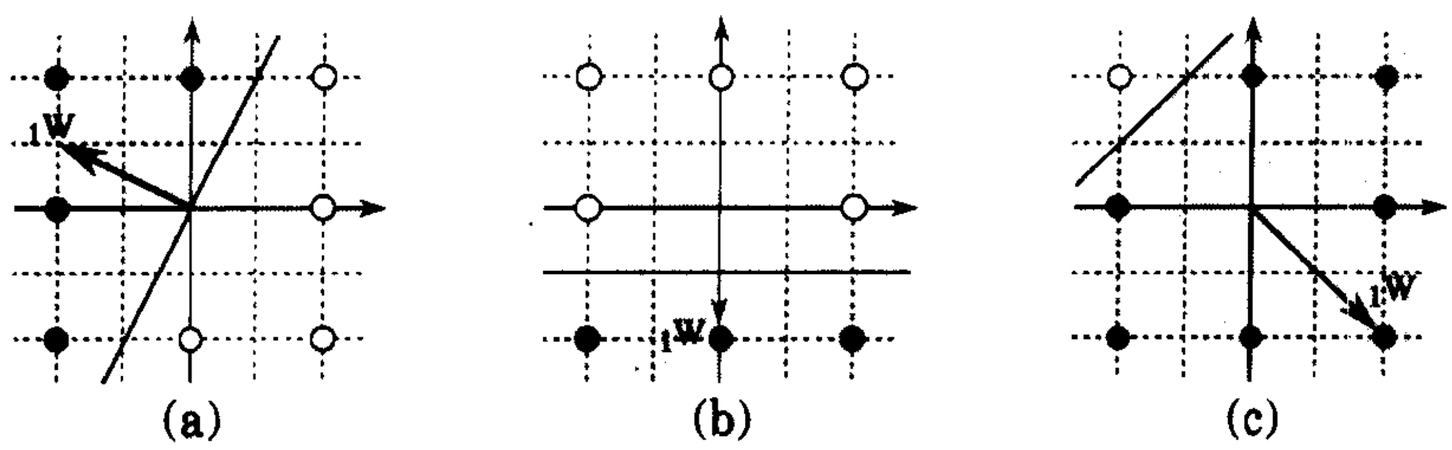

图 4-19 简单的分类问题

**解**

首先在黑色数据点集和空心数据点集之间画一条直线将它们分开。




下一步求解相应的权值和偏置值。权值向量必须与判定边界垂直，并指向类 1(黑色点)一方，而权值向量的长度则可任意选择。




下面是所选择的一组权值向量：

4-21 (a) ${}_1\mathbf{w}^T = [-2 \quad 1]$, (b) ${}_1\mathbf{w}^T = [0 \quad -2]$, (c) ${}_1\mathbf{w}^T = [2 \quad -2]$

为求解每个感知机的偏置值。可以选择判定边界上满足式(4.15)的点：

$$ {}_1\mathbf{w}^T\mathbf{p} + b = 0 $$

$$ b = -{}_1\mathbf{w}^T\mathbf{p} $$

据此可得如下三个偏置值：

$$ (a)\, b = -[-2 \quad 1]\begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0, \quad (b)\, b = -[0 \quad -2]\begin{bmatrix} 0 \\ -1 \end{bmatrix} = -2, $$

$$ (c)\, b = -[2 \quad -2]\begin{bmatrix} -2 \\ 1 \end{bmatrix} = 6 $$

现在可以利用问题所给出的数据点来验证这些解。下面用输入向量 $\mathbf{p} = [-2 \quad 2]^T$ 来验证第一个网络：

$$
\begin{aligned}
a &= hardlim({}_1\mathbf{w}^T\mathbf{p} + b) \\
&= hardlim\left([-2 \quad 1]\begin{bmatrix}-2\\2\end{bmatrix} + 0\right) \\
&= hardlim(6) \\
&= 1
\end{aligned}
$$

读者可以用 MATLAB 完成对新数据点的自动验证过程。这里用第一个网络对一个不在原问题中的数据点进行分类：

```
w = [-2  1];   b = 0;
a = hardlim(w*[1;1] + b)
a =
     0
```

**P4.2** 将下面所定义的分类问题转换为由一组不等式约束的权值和偏置值所定义的一个等价问题。

$$
\left\{\mathbf{p}_1 = \begin{bmatrix}0\\2\end{bmatrix}, t_1 = 1\right\}\left\{\mathbf{p}_2 = \begin{bmatrix}1\\0\end{bmatrix}, t_2 = 1\right\}\left\{\mathbf{p}_3 = \begin{bmatrix}0\\-2\end{bmatrix}, t_3 = 0\right\}\left\{\mathbf{p}_4 = \begin{bmatrix}2\\0\end{bmatrix}, t_4 = 0\right\}
$$

**解**

每个目标 $t_i$ 表明了相应于 $\mathbf{p}_i$ 的净输入是小于 0 还是大于等于 0。比如，由于 $t_1$ 是 1，则相应于 $\mathbf{p}_1$ 的净输入一定大于等于 0。因此，可以得到下列不等式： 4-22

$$
\begin{aligned}
\mathbf{W}\mathbf{p}_1 + b &\geqslant 0 \\
0w_{1,1} + 2w_{1,2} + b &\geqslant 0 \\
2w_{1,2} + b &\geqslant 0
\end{aligned}
$$

对输入/目标对 $\{\mathbf{p}_2, t_2\}$、$\{\mathbf{p}_3, t_3\}$ 和 $\{\mathbf{p}_4, t_4\}$ 应用上述过程，可以得到如下一组不等式：

$$
\begin{aligned}
&2w_{1,2} + b \geqslant 0 && \text{(i)} \\
&w_{1,1} + b \geqslant 0 && \text{(ii)} \\
&-2w_{1,2} + b < 0 && \text{(iii)} \\
&2w_{1,1} + b < 0 && \text{(iv)}
\end{aligned}
$$

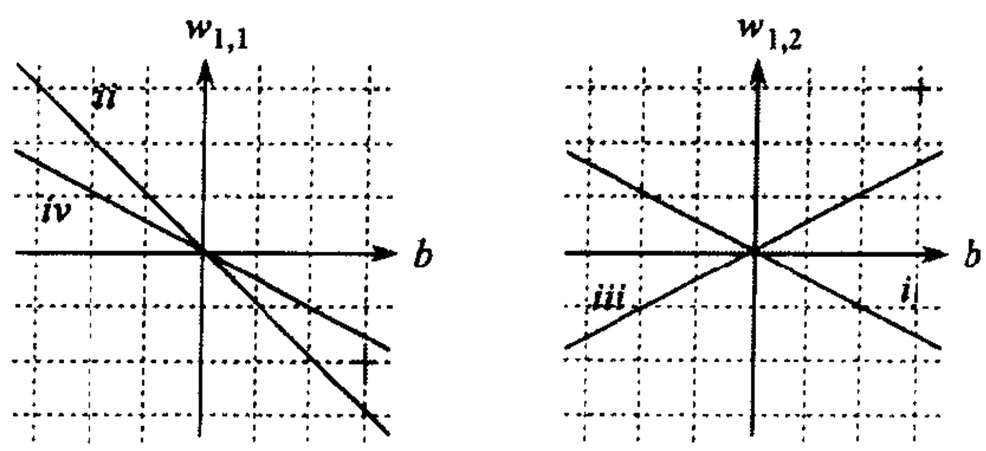
解不等式组比解方程组要难，难在通常情况下不等式组都有无数个解(就像是线性可分的分类问题通常有无数个线性判定边界一样)。

不过，由于此问题比较简单，所以可以通过图解由不等式组定义的解空间来求解。请注意 $w_{1,1}$ 仅出现在(ii)和(iv)中，而 $w_{1,2}$ 仅出现在(i)和(iii)中。所以，两组不等式可用如下两个图来表示：

任何落于暗灰色区域中的权值和偏置值都可作为此分类问题的解。其中一个解为：

$$
\mathbf{W} = [-2 \quad 3], \quad b = 3
$$

4-23




**P4.3** 考虑具有如下四类输入向量的分类问题。这四类输入向量分别是

$$第1类:\left\{\mathbf{p}_1=\begin{bmatrix}1\\1\end{bmatrix},\mathbf{p}_2=\begin{bmatrix}1\\2\end{bmatrix}\right\}\quad 第2类:\left\{\mathbf{p}_3=\begin{bmatrix}2\\-1\end{bmatrix},\mathbf{p}_4=\begin{bmatrix}2\\0\end{bmatrix}\right\}$$

$$第3类:\left\{\mathbf{p}_5=\begin{bmatrix}-1\\2\end{bmatrix},\mathbf{p}_6=\begin{bmatrix}-2\\1\end{bmatrix}\right\}\quad 第4类:\left\{\mathbf{p}_7=\begin{bmatrix}-1\\-1\end{bmatrix},\mathbf{p}_8=\begin{bmatrix}-2\\-2\end{bmatrix}\right\}$$

试设计一种感知机网络求解此问题。

**解**

由于 $S$ 个神经元的感知机可对 $2^S$ 个类别进行分类，所以求解此问题至少需要两个神经元。这种两神经元的感知机如图 4-20 所示。

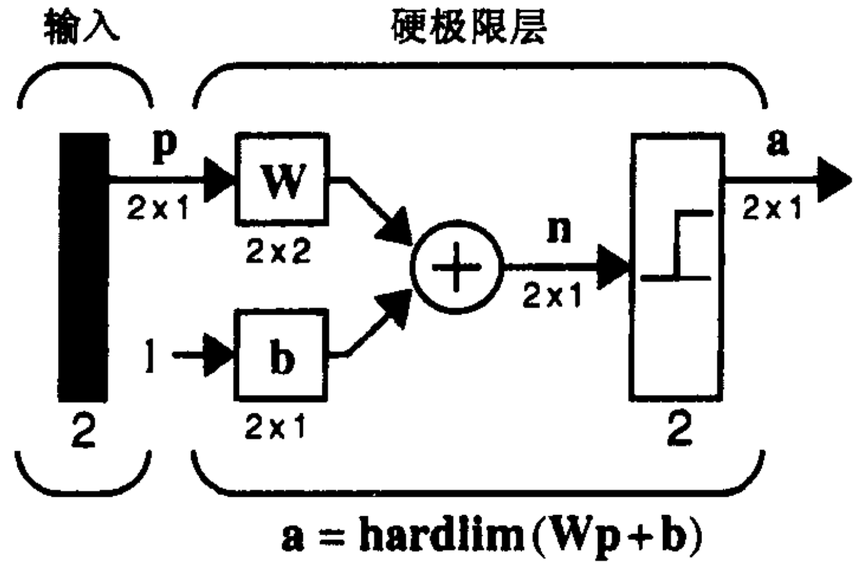



图 4-20 两神经元的感知机

我们从显示在图 4-21 的输入向量开始。图中用空心圆圈○表示第 1 类输入向量，用空心方框□表示第 2 类输入向量，用黑色圆圈●表示第 3 类输入向量，用黑色方框■表示第 4 类输入向量。

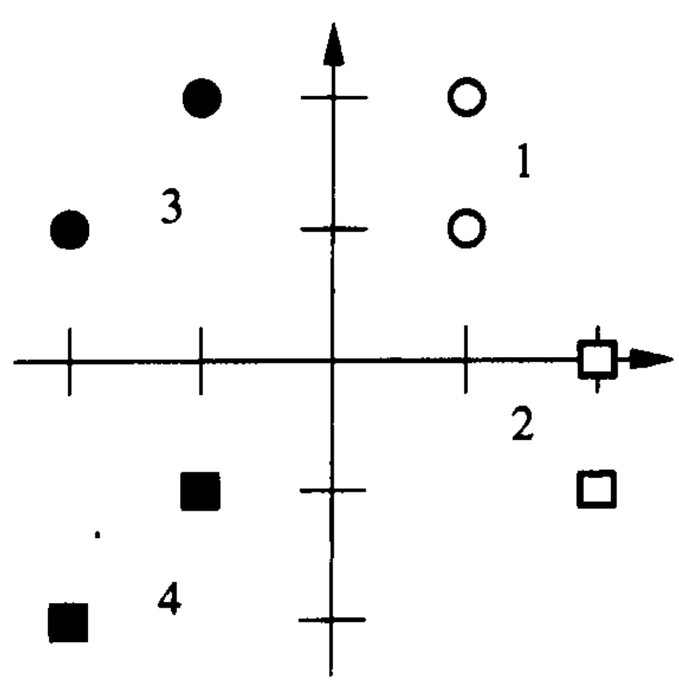



图 4-21 例题 P4.3 的输入向量

两神经元感知机可以生成两条判定边界。为了将输入空间分为四类，必须要有一条判定边界将四类输入分为两组，每组分别包含两类输入，而另一条判定边界必须能够将各类输入区分开(如图 4-22 所示)。从图 4-22 可知问题的模式是线性可分的。 4-24

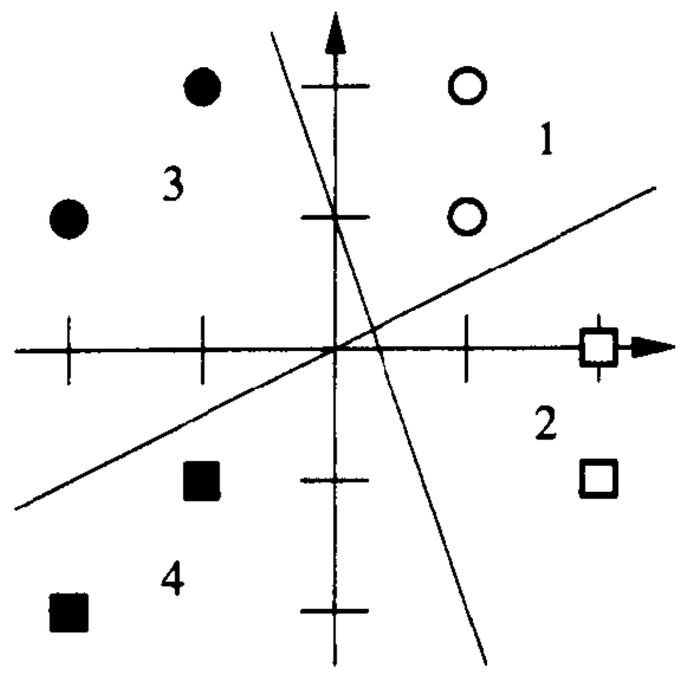



图 4-22　例题 P4.3 的试用判定边界

权值向量必须与判定边界垂直，且指向神经元输出为 1 的区域。下一步将确定每条边界的哪一边应该输出 1。其中一种选择如图 4-23 所示，图中阴影部分表示存在神经元输出为 1 的输入区域，而最暗的阴影表示两个神经元的输出都为 1 的输入区域。请注意，这个解对应的目标值分别为：

$$\text{第 1 类}:\left\{\mathbf{t}_1=\begin{bmatrix}0\\0\end{bmatrix},\mathbf{t}_2=\begin{bmatrix}0\\0\end{bmatrix}\right\}\quad \text{第 2 类}:\left\{\mathbf{t}_3=\begin{bmatrix}0\\1\end{bmatrix},\mathbf{t}_4=\begin{bmatrix}0\\1\end{bmatrix}\right\}$$

$$\text{第 3 类}:\left\{\mathbf{t}_5=\begin{bmatrix}1\\0\end{bmatrix},\mathbf{t}_6=\begin{bmatrix}1\\0\end{bmatrix}\right\}\quad \text{第 4 类}:\left\{\mathbf{t}_7=\begin{bmatrix}1\\1\end{bmatrix},\mathbf{t}_8=\begin{bmatrix}1\\1\end{bmatrix}\right\}$$

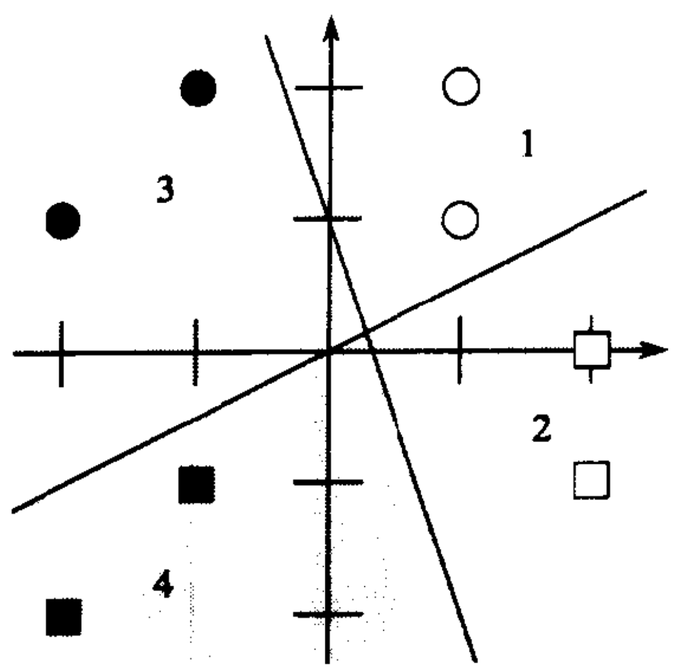



图 4-23　例题 P4.3 的判定区域

可以选择权值向量为 4-25

$$_1\mathbf{W}=\begin{bmatrix}-3\\-1\end{bmatrix},\quad {}_2\mathbf{W}=\begin{bmatrix}1\\-2\end{bmatrix}$$

请注意：权值向量的长度并不十分重要，重要的是它们的方向。它们必须和判定边界垂直。现在可以在判定边界上选择满足式(4.15)的一个点来计算偏置值：

$$b_1 = -{}_1\mathbf{w}^T\mathbf{p} = -\begin{bmatrix}-3 & -1\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} = 1$$

$$b_2 = -{}_2\mathbf{w}^T\mathbf{p} = -\begin{bmatrix}1 & -2\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix} = 0$$

按照矩阵的表示形式，有

$$\mathbf{W} = \begin{bmatrix}{}_1\mathbf{w}^T\\{}_2\mathbf{w}^T\end{bmatrix} = \begin{bmatrix}-3 & -1\\1 & -2\end{bmatrix},\quad \mathbf{b} = \begin{bmatrix}1\\0\end{bmatrix}$$

至此完成了所要求的设计。

**P4.4** 请用感知机学习规则求解如下分类问题。按顺序重复使用各个输入向量，直至最终求得问题的解，并在求出一个解后画出问题的图形。

4-26

$$\left\{\mathbf{p}_1 = \begin{bmatrix}2\\2\end{bmatrix}, t_1 = 0\right\}\left\{\mathbf{p}_2 = \begin{bmatrix}1\\-2\end{bmatrix}, t_2 = 1\right\}\left\{\mathbf{p}_3 = \begin{bmatrix}-2\\2\end{bmatrix}, t_3 = 0\right\}\left\{\mathbf{p}_4 = \begin{bmatrix}-1\\1\end{bmatrix}, t_4 = 1\right\}$$

请使用如下的初始权值和偏置值：

$$\mathbf{W}(0) = \begin{bmatrix}0 & 0\end{bmatrix},\quad b(0) = 0$$

**解**

首先利用初始的权值和偏置值计算与第一个输入向量 $\mathbf{p}_1$ 相应的感知机输出 $a$：

$$\begin{aligned}a &= hardlim(\mathbf{W}(0)\mathbf{p}_1 + b(0))\\ &= hardlim\left(\begin{bmatrix}0 & 0\end{bmatrix}\begin{bmatrix}2\\2\end{bmatrix} + 0\right) = hardlim(0) = 1\end{aligned}$$

感知机实际输出值 $a$ 不等于输入向量 $\mathbf{p}_1$ 的目标值 $t_1$，所以要按学习规则根据误差求解新的权值和偏置值。

$$e = t_1 - a = 0 - 1 = -1$$

$$\mathbf{W}(1) = \mathbf{W}(0) + e\mathbf{p}_1^T = \begin{bmatrix}0 & 0\end{bmatrix} + (-1)\begin{bmatrix}2 & 2\end{bmatrix} = \begin{bmatrix}-2 & -2\end{bmatrix}$$

$$b(1) = b(0) + e = 0 + (-1) = -1$$

然后应用修改后的权值和偏置值处理 $\mathbf{p}_2$：

$$\begin{aligned}a &= hardlim(\mathbf{W}(1)\mathbf{p}_2 + b(1))\\ &= hardlim\left(\begin{bmatrix}-2 & -2\end{bmatrix}\begin{bmatrix}1\\-2\end{bmatrix} - 1\right) = hardlim(1) = 1\end{aligned}$$

这次感知机的实际输出 $a$ 等于输入向量 $\mathbf{p}_2$ 的目标值 $t_2$。根据感知机的学习规则可知，不会改变权值和偏置值：

$$\mathbf{W}(2) = \mathbf{W}(1)$$
$$b(2) = b(1)$$

4-27

现在处理第三个输入向量：

$$\begin{aligned}a &= hardlim(\mathbf{W}(2)\mathbf{p}_3 + b(2))\\ &= hardlim\left(\begin{bmatrix}-2 & -2\end{bmatrix}\begin{bmatrix}-2\\-2\end{bmatrix} - 1\right) = hardlim(-1) = 0\end{aligned}$$

可以看出感知机的实际输出值等于输入向量 $\mathbf{p}_3$ 的目标值 $t_3$，同样不会修改权值和偏置值。

$$\mathbf{W}(3) = \mathbf{W}(2)$$
$$b(3) = b(2)$$

最后转到对输入向量 $\mathbf{p}_4$ 进行处理：

$$a = hardlim(\mathbf{W}(3)\mathbf{p}_4 + b(3))$$
$$= hardlim\left(\begin{bmatrix} -2 & -2 \end{bmatrix}\begin{bmatrix} -1 \\ 1 \end{bmatrix} - 1\right) = hardlim(-1) = 0$$

可以看出，感知机当前的实际输出值 $a$ 不等于输入向量 $\mathbf{p}_4$ 的目标值 $t_4$。所以，感知机的学习规则将对权值 $\mathbf{W}$ 和偏置值 $b$ 进行修改：

$$e = t_4 - a = 1 - 0 = 1$$
$$\mathbf{W}(4) = \mathbf{W}(3) + e\mathbf{p}_4^T = \begin{bmatrix} -2 & -2 \end{bmatrix} + (1)\begin{bmatrix} -1 & 1 \end{bmatrix} = \begin{bmatrix} -3 & -1 \end{bmatrix}$$
$$b(4) = b(3) + e = -1 + 1 = 0$$

现在必须再次检测第一个输入向量 $\mathbf{p}_1$。感知机这次的实际输出值 $a$ 等于第一个输入向量 $\mathbf{p}_1$ 的目标值 $t_1$。

$$a = hardlim(\mathbf{W}(4)\mathbf{p}_1 + b(4))$$
$$= hardlim\left(\begin{bmatrix} -3 & -1 \end{bmatrix}\begin{bmatrix} 2 \\ 2 \end{bmatrix} + 0\right) = hardlim(-8) = 0$$

所以不会改变权值和偏置值。

$$\mathbf{W}(5) = \mathbf{W}(4)$$
$$b(5) = b(4)$$

第二次输入向量 $\mathbf{p}_2$ 后，由于感知机的实际输出和所期望的目标输出之间存在误差，所以又需修改权值和偏置值：

4-28

$$a = hardlim(\mathbf{W}(5)\mathbf{p}_2 + b(5))$$
$$= hardlim\left(\begin{bmatrix} -3 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ -2 \end{bmatrix} + 0\right) = hardlim(-1) = 0$$

相应地，感知机各个参数新的取值为：

$$e = t_2 - a = 1 - 0 = 1$$
$$\mathbf{W}(6) = \mathbf{W}(5) + e\mathbf{p}_2^T = \begin{bmatrix} -3 & -1 \end{bmatrix} + (1)\begin{bmatrix} 1 & -2 \end{bmatrix} = \begin{bmatrix} -2 & -3 \end{bmatrix}$$
$$b(6) = b(5) + e = 0 + 1 = 1$$

重复上述过程，再一次经过每个输入向量，就能够产生没有误差的正确分类：

$$a = hardlim(\mathbf{W}(6)\mathbf{p}_3 + b(6)) = hardlim\left(\begin{bmatrix} -2 & -3 \end{bmatrix}\begin{bmatrix} -2 \\ 2 \end{bmatrix} + 1\right) = 0 = t_3$$

$$a = hardlim(\mathbf{W}(6)\mathbf{p}_4 + b(6)) = hardlim\left(\begin{bmatrix} -2 & -3 \end{bmatrix}\begin{bmatrix} -1 \\ 1 \end{bmatrix} + 1\right) = 1 = t_4$$

$$a = hardlim(\mathbf{W}(6)\mathbf{p}_1 + b(6)) = hardlim\left(\begin{bmatrix} -2 & -3 \end{bmatrix}\begin{bmatrix} 2 \\ 2 \end{bmatrix} + 1\right) = 0 = t_1$$

$$a = hardlim(\mathbf{W}(6)\mathbf{p}_2 + b(6)) = hardlim\left(\begin{bmatrix} -2 & -3 \end{bmatrix}\begin{bmatrix} 1 \\ -2 \end{bmatrix} + 1\right) = 1 = t_2$$

所以算法已经收敛，最终的解为：

$$\mathbf{W} = [-2 \quad -3] \quad b = 1$$

现在就可以用图形的方式表示训练数据和判定边界。判定边界由下式给定：

$$n = \mathbf{Wp} + b = w_{1,1}p_1 + w_{1,2}p_2 + b = -2p_1 - 3p_2 + 1 = 0$$

令 $p_1 = 0$，可以求得判定边界在坐标轴 $p_2$ 上的截距为：

$$p_2 = -\frac{b}{w_{1,2}} = -\frac{1}{-3} = \frac{1}{3} \quad (\text{如果 } p_1 = 0)$$

令 $p_2 = 0$，同样可以求得判定边界再坐标轴 $p_1$ 上的截距为：

4-29

$$p_1 = -\frac{b}{w_{1,1}} = -\frac{1}{-2} = \frac{1}{2} \quad (\text{如果 } p_2 = 0)$$

求解得到的判定边界如图 4-24 所示。

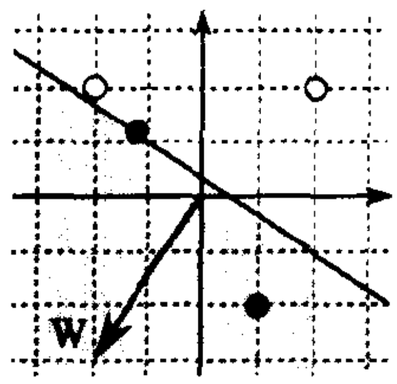



图 4-24 例题 P4.4 的判定边界

请注意：上述判定边界刚好穿过一个训练向量。根据问题的定义，这是完全可以接受的，因为求解中所用的硬极限函数当其输入为 0 时，函数值为 1，在例题中，该向量的目标值就是 1。

**P4.5** 继续考虑例题 P4.3 中的四类判定问题。利用感知机学习规则训练一种感知机网络来求解这个问题。

**解**

如果采用与例题 P4.3 中相同的目标向量，那么训练集为：

$$\left\{\mathbf{p}_1 = \begin{bmatrix}1\\1\end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix}0\\0\end{bmatrix}\right\}\left\{\mathbf{p}_2 = \begin{bmatrix}1\\2\end{bmatrix}, \mathbf{t}_2 = \begin{bmatrix}0\\0\end{bmatrix}\right\}$$

$$\left\{\mathbf{p}_3 = \begin{bmatrix}2\\-1\end{bmatrix}, \mathbf{t}_3 = \begin{bmatrix}0\\1\end{bmatrix}\right\}\left\{\mathbf{p}_4 = \begin{bmatrix}2\\0\end{bmatrix}, \mathbf{t}_4 = \begin{bmatrix}0\\1\end{bmatrix}\right\}$$

$$\left\{\mathbf{p}_5 = \begin{bmatrix}-1\\2\end{bmatrix}, \mathbf{t}_5 = \begin{bmatrix}1\\0\end{bmatrix}\right\}\left\{\mathbf{p}_6 = \begin{bmatrix}-2\\1\end{bmatrix}, \mathbf{t}_6 = \begin{bmatrix}1\\0\end{bmatrix}\right\}$$

$$\left\{\mathbf{p}_7 = \begin{bmatrix}-1\\-1\end{bmatrix}, \mathbf{t}_7 = \begin{bmatrix}1\\1\end{bmatrix}\right\}\left\{\mathbf{p}_8 = \begin{bmatrix}-2\\-2\end{bmatrix}, \mathbf{t}_8 = \begin{bmatrix}1\\1\end{bmatrix}\right\}$$

假设算法的初始权值和偏置值分别为：

$$\mathbf{W}(0) = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}, \quad \mathbf{b}(0) = \begin{bmatrix}1\\1\end{bmatrix}$$

第 1 次迭代结果为：

$$\mathbf{a} = hardlim(\mathbf{W}(0)\mathbf{p}_1 + \mathbf{b}(0)) = hardlim\left(\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} + \begin{bmatrix}1\\1\end{bmatrix}\right) = \begin{bmatrix}1\\1\end{bmatrix} \tag{4-30}$$

$$\mathbf{e} = \mathbf{t}_1 - \mathbf{a} = \begin{bmatrix}0\\0\end{bmatrix} - \begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}-1\\-1\end{bmatrix}$$

$$\mathbf{W}(1) = \mathbf{W}(0) + \mathbf{e}\mathbf{p}_1^T = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} + \begin{bmatrix}-1\\-1\end{bmatrix}\begin{bmatrix}1 & 1\end{bmatrix} = \begin{bmatrix}0 & -1\\-1 & 0\end{bmatrix}$$

$$\mathbf{b}(1) = \mathbf{b}(0) + \mathbf{e} = \begin{bmatrix}1\\1\end{bmatrix} + \begin{bmatrix}-1\\-1\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}$$

第 2 次迭代结果为：

$$\mathbf{a} = hardlim(\mathbf{W}(1)\mathbf{p}_2 + \mathbf{b}(1)) = hardlim\left(\begin{bmatrix}0 & -1\\-1 & 0\end{bmatrix}\begin{bmatrix}1\\2\end{bmatrix} + \begin{bmatrix}0\\0\end{bmatrix}\right) = \begin{bmatrix}0\\0\end{bmatrix}$$

$$\mathbf{e} = \mathbf{t}_2 - \mathbf{a} = \begin{bmatrix}0\\0\end{bmatrix} - \begin{bmatrix}0\\0\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}$$

$$\mathbf{W}(2) = \mathbf{W}(1) + \mathbf{e}\mathbf{p}_2^T = \begin{bmatrix}0 & -1\\-1 & 0\end{bmatrix} + \begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1 & 2\end{bmatrix} = \begin{bmatrix}0 & -1\\-1 & 0\end{bmatrix}$$

$$\mathbf{b}(2) = \mathbf{b}(1) + \mathbf{e} = \begin{bmatrix}0\\0\end{bmatrix} + \begin{bmatrix}0\\0\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}$$

第 3 次迭代结果为：

$$\mathbf{a} = hardlim(\mathbf{W}(2)\mathbf{p}_3 + \mathbf{b}(2)) = hardlim\left(\begin{bmatrix}0 & -1\\-1 & 0\end{bmatrix}\begin{bmatrix}2\\-1\end{bmatrix} + \begin{bmatrix}0\\0\end{bmatrix}\right) = \begin{bmatrix}1\\0\end{bmatrix}$$

$$\mathbf{e} = \mathbf{t}_3 - \mathbf{a} = \begin{bmatrix}0\\1\end{bmatrix} - \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}-1\\1\end{bmatrix}$$

$$\mathbf{W}(3) = \mathbf{W}(2) + \mathbf{e}\mathbf{p}_3^T = \begin{bmatrix}0 & -1\\-1 & 0\end{bmatrix} + \begin{bmatrix}-1\\1\end{bmatrix}\begin{bmatrix}2 & -1\end{bmatrix} = \begin{bmatrix}-2 & 0\\1 & -1\end{bmatrix} \tag{4-31}$$

$$\mathbf{b}(3) = \mathbf{b}(2) + \mathbf{e} = \begin{bmatrix}0\\0\end{bmatrix} + \begin{bmatrix}-1\\1\end{bmatrix} = \begin{bmatrix}-1\\1\end{bmatrix}$$

从第 4 次迭代到第 8 次迭代的过程中，权值矩阵和偏置值向量均没有作任何修改：

$$\mathbf{W}(8) = \mathbf{W}(7) = \mathbf{W}(6) = \mathbf{W}(5) = \mathbf{W}(4) = \mathbf{W}(3)$$

$$\mathbf{b}(8) = \mathbf{b}(7) = \mathbf{b}(6) = \mathbf{b}(5) = \mathbf{b}(4) = \mathbf{b}(3)$$

第 9 次迭代结果为：

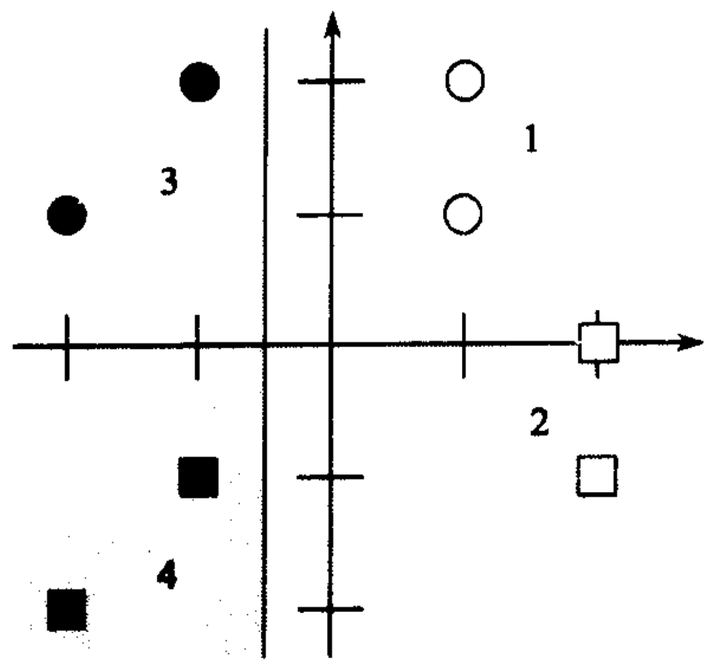
$$\mathbf{a} = hardlim(\mathbf{W}(8)\mathbf{p}_1 + \mathbf{b}(8)) = hardlim\left(\begin{bmatrix}-2 & 0\\1 & -1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} + \begin{bmatrix}-1\\1\end{bmatrix}\right) = \begin{bmatrix}0\\1\end{bmatrix}$$

$$\mathbf{e} = \mathbf{t}_1 - \mathbf{a} = \begin{bmatrix}0\\0\end{bmatrix} - \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\-1\end{bmatrix}$$

$$\mathbf{W}(9) = \mathbf{W}(8) + \mathbf{e}\mathbf{p}_1^T = \begin{bmatrix}-2 & 0\\1 & -1\end{bmatrix} + \begin{bmatrix}0\\-1\end{bmatrix}\begin{bmatrix}1 & 1\end{bmatrix} = \begin{bmatrix}-2 & 0\\0 & -2\end{bmatrix}$$

$$\mathbf{b}(9) = \mathbf{b}(8) + \mathbf{e} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

自此算法已经收敛，因为所有输入模式将被正确分类。最终的判定边界如图 4-25 所示。请读者将这个结果与例题 P4.3 中设计的网络相比较。




4-32

图 4-25 例题 P4.5 最终的判定边界

## 4.5 结束语

本章介绍了我们的第一个学习规则——感知机规则。感知机学习规则属于有监督学习类型，其中学习规则用一组正确反映网络行为的实例的方式提供。当每个输入送入网络后，该规则调整网络参数，使网络的实际输出逐步接近相应输入的目标值。

虽然感知机的学习规则非常简单，但是它的功能十分强大。前面已经证明：只要问题的解存在，那么学习规则总能收敛到正确的解上。感知机的弱点并不在于它的学习规则，而是在于其简单的网络结构。标准的感知机模型只能分类线性可分的向量。本书的第 11 章将会把感知机结构扩展到多层感知机，以求解任意的分类问题。将在第 11 章介绍的反传学习规则可以用于训练这些网络。

第 3 章和第 4 章使用了线性代数的许多概念，如内积、投影、距离(范数)等。在后面各章，读者将会发现良好的线性代数基础对理解神经网络模型是非常必要的。第 5 章和第 6 章将回顾一些对学习神经网络较为重要的线性代数的关键概念，目的是为深入理解神经网络奠定良好的基础知识。

4-33

### 参考文献

[BaSu83] A. Barto, R. Sutton and C. Anderson, "Neuron - like adaptive elements can solve difficult learning control problems", *IEEE Transaction on Systems, Man and Cybernetics*, Vol. 13, No. 5, pp. 834 - 846, 1983.

这是一篇介绍用增强学习算法训练神经网络平衡逆向振荡的经典论文。

[Brog91] W. L. Brogan, *Modern Control Theory*, 3rd Ed., Englewood Cliffs, NJ: Prentice - Hall, 1991.

这是一本关于线性系统的好书。书的前半部分主要介绍线性代数知识。书中有几节

对求解线性微分方程以及线性和非线性系统的稳定性问题作了很好的讨论。书中还有许多例题和习题。

[McPi43] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115 - 133, 1943.

这篇文章介绍了神经元的第一个数学模型，神经元根据输入信号的加权求和与阈值的比较结果确定它是否激活。

[MiPa69] M. Minsky and Papert, *Perceptrons*, Cambridge, MA: MIT Press, 1969.

这是一本具有标志性意义的著作，其中包含了第一次对感知机能够学习什么这一问题的严密研究。指出对待感知机的正确态度应该是既要解释感知机的局限性，而且要找到克服这些局限性的方向。不幸的是，该书悲观地认为感知机的局限性说明了神经网络领域是一条死胡同。尽管这不是真实的情况，但它还是在此后若干年内严重影响了神经网络的研究和投资。

[Rose58] F. Rosenblatt, "The Perceptron: A probabilistic Model for information storage and organization in the brain", *Psychological Review*, Vol. 65, pp. 386 - 408, 1958.

本文提出了第一种实用的人工神经网络——感知机。 4-34

[Rose61] F. Rosenblatt, *Principles of Neurodynamics*, Washington DC: Spartan Press, 1961.

这是首批关于神经计算的书之一。

[WhSo92] D. White and D. Sofge(Eds.), *Handbook of Intelligent Control*, New York: Van Nostrand Reinhold, 1992.

该书收集了当时一些关于控制系统中的神经网络和模糊逻辑的研究和应用方面的论文。 4-35

## 习题

**E4.1** 考虑下面定义的分类问题：

$$\left\{\mathbf{p}_1=\begin{bmatrix}1\\-1\end{bmatrix},t_1=1\right\}\left\{\mathbf{p}_2=\begin{bmatrix}0\\0\end{bmatrix},t_2=1\right\}\left\{\mathbf{p}_3=\begin{bmatrix}1\\-1\end{bmatrix},t_3=1\right\}$$

$$\left\{\mathbf{p}_4=\begin{bmatrix}1\\0\end{bmatrix},t_4=0\right\}\left\{\mathbf{p}_5=\begin{bmatrix}0\\1\end{bmatrix},t_5=0\right\}$$

(i) 画出能求解此问题的单神经元感知机结构图，并指出需要多少个输入？

(ii) 画出输入数据点的分布图，并根据目标值对其进行标记。用(i)中所给出的网络能够求解这个问题吗？为什么？

**E4.2** 考虑下面定义的分类问题：

$$\left\{\mathbf{p}_1=\begin{bmatrix}-1\\1\end{bmatrix},t_1=1\right\}\left\{\mathbf{p}_2=\begin{bmatrix}-1\\-1\end{bmatrix},t_2=1\right\}$$

$$\left\{\mathbf{p}_3=\begin{bmatrix}0\\0\end{bmatrix},t_3=0\right\}\left\{\mathbf{p}_4=\begin{bmatrix}1\\0\end{bmatrix},t_4=0\right\}$$

(i) 设计一个求解这个问题的单神经元感知机。选择与判定边界垂直的权值向

量以图形方式设计出网络。

(ii) 用全部 4 个输入向量验证求解结果。

(iii) 用求解结果对下面 4 个输入向量分类。可以手工计算，也可以用 MATLAB 计算：

$$\mathbf{p}_5 = \begin{bmatrix} -2 \\ 0 \end{bmatrix} \quad \mathbf{p}_6 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \mathbf{p}_7 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \mathbf{p}_8 = \begin{bmatrix} -1 \\ -2 \end{bmatrix}$$

(iv) (iii)中哪个向量总是用同一方式分类而同 $\mathbf{W}$，$b$ 的选择无关？哪些向量依赖于 $\mathbf{W}$ 和 $b$ 的选择？为什么？

**E4.3** 用解不等式的方法求习题 E4.2 的解(参考例题 P4.2)，并用新的求解结果重做习题 E4.2d 的(ii)和(iii)题。(由于不能以成对的方式将权值和偏置值分离开来，所以这里的求解过程要比例题 P4.2 复杂。)

4-36

**E4.4** 对下列初始参数，应用感知机学习规则求解习题 E4.2 的分类问题，并用新的求解结果重做习题 E4.2 的(i)，(ii)和(iii)题。

$$\mathbf{W}(0) = [0 \quad 0], \quad b(0) = 0$$

**E4.5** 用数学方法(而不是图形方式)证明下面问题对于两输入/单神经元感知机而言是不可解的。

$$\left\{\mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, t_1 = 1\right\}\left\{\mathbf{p}_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, t_2 = 0\right\}$$

$$\left\{\mathbf{p}_3 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, t_3 = 1\right\}\left\{\mathbf{p}_4 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_4 = 0\right\}$$

(提示：将输入/目标分类要求以限制权值和偏置值的不等式的方式写出来。)

**E4.6** 有时在感知机网络中也会采用对称硬极限传输函数 *hardlims*(如图 4-26 所示)，而不采用硬极限传输函数 *hardlim*。此时目标值也将变成在集合[-1，1]中取值，而不是在集合[0，1]中取值。

(i) 写出分别将有序集[0，1]的数映射到有序集[-1，1]的简单表达式，以及执行逆映射的表达式。

(ii) 考虑两个权值和偏置值都相同的单神经元感知机。第一个网络采用在集合[0，1]中取值的硬极限函数 *hardlim*，而第二个网络采用对称硬极限函数 *hardlims*。如果提交给两个网络的输入都是 $\mathbf{p}$，并按照感知机的学习规则更新输入，那么它们的权值还将会一样吗？

(iii) 如果对两个神经元的权值的改变不一样，那么它们有什么不同？为什么？

(iv) 对采用硬极限传输函数的标准感知机设定初始权值和偏置值，试为采用对称硬极限传输函数的感知机构造一个初始化方法，使得两个感知机能够在训练同样的数据时，响应也一样。

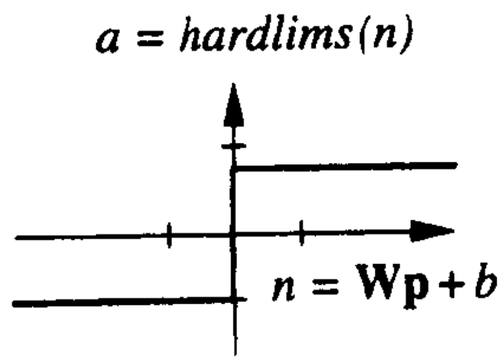



图 4-26

**E4.7** 下面定义的有序集是通过测量 Fuzzy Wuzzy 动物玩具厂的玩具兔和玩具熊的重量和耳朵的长度获得的。目标值表明了相应输入向量表示的是兔子(0)还是熊(1)。输入向量的第一个元素是玩具的重量，第二个元素是玩具耳朵的长度。

$$\left\{\mathbf{p}_1 = \begin{bmatrix}1\\4\end{bmatrix}, t_1 = 0\right\}\left\{\mathbf{p}_2 = \begin{bmatrix}1\\5\end{bmatrix}, t_2 = 0\right\}$$ 4-37

$$\left\{\mathbf{p}_3 = \begin{bmatrix}2\\4\end{bmatrix}, t_3 = 0\right\}\left\{\mathbf{p}_4 = \begin{bmatrix}2\\5\end{bmatrix}, t_4 = 0\right\}$$

$$\left\{\mathbf{p}_5 = \begin{bmatrix}3\\1\end{bmatrix}, t_5 = 1\right\}\left\{\mathbf{p}_6 = \begin{bmatrix}3\\2\end{bmatrix}, t_6 = 1\right\}$$

$$\left\{\mathbf{p}_7 = \begin{bmatrix}4\\1\end{bmatrix}, t_7 = 1\right\}\left\{\mathbf{p}_8 = \begin{bmatrix}4\\2\end{bmatrix}, t_8 = 1\right\}$$

(i) 用 MATLAB 对一个网络进行初始化和训练，以求解这个“实际”问题。

(ii) 用 MATLAB 和输入向量来验证所求的权值和偏置值。

(iii) 改变输入向量，使任何解的判定边界都不会通过一个原始输入向量(即保证求解过程只会得到鲁棒性判定边界)。然后重新训练该网络。

**E4.8** 请重新考虑例题 P4.3 和 P4.5 中给出的四种类别的分类问题。假设将输入向量 $\mathbf{p}_3$ 改为

$$\mathbf{p}_3 = \begin{bmatrix}2\\2\end{bmatrix}$$

(i) 这时问题仍然是线性可分的吗？请用图来表示答案。

(ii) 利用 MATLAB 工具，对一个网络进行初始化和训练以求解这个问题。请解释求解结果。

(iii) 如果将 $\mathbf{p}_3$ 改为 $\mathbf{p}_3 = \begin{bmatrix}2\\1.5\end{bmatrix}$，这时问题是线性可分的吗？

(iv) 根据(iii)题中给出的 $\mathbf{p}_3$，利用 MATLAB 工具，对一个网络进行初始化和训练以求解这个问题。请解释求解结果。

**E4.9** 下面是一种变形的感知机学习规则：

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha\mathbf{e}\mathbf{p}^T$$
$$\mathbf{b}^{new} = \mathbf{b}^{old} + \alpha\mathbf{e}$$

其中称 $\alpha$ 为学习速率。证明这个算法的收敛性。证明中需要对学习速率作出限制吗？试作出解释。 4-38

# 第5章　信号和权值向量空间

## 5.1　目标

从第 3 章和第 4 章可以看出：将神经网络的输入、输出以及权值矩阵的行作为向量看待是非常有好处的。这一章将详细研究这些向量空间，并且复习一些对分析神经网络十分有用的向量空间性质。这里首先将从一般的定义开始，并将这些定义应用于特定的神经网络问题中。本章和第 6 章所讨论的概念将被广泛应用于本书其他各章。这些概念是深入理解神经网络工作原理的关键。

5-1

## 5.2　理论和实例

线性代数是理解神经网络所必需的数学知识的核心。读者在第 3 章和第 4 章中看到了神经网络输入/输出向量表示的应用。而且，不难发现将权值矩阵的行看作是和输入向量处于同一向量空间中的向量也是十分有用的。

在第 3 章的 Hamming 网络中，前馈层权值矩阵的行等于标准向量。实际上，前馈层的任务就是计算标准向量和输入向量之间的内积。在单神经元感知机网络中，也可看到判定边界总是和其权值矩阵(一个行向量)垂直。

本章将复习在神经网络中有关向量空间的一些基本概念(比如内积、正交性等)。这里将从向量空间的一般定义开始，给出神经网络应用中常用的一些向量基本性质。

在开始前要说明一下向量的记号。到目前为止所讨论的向量都是实数的有序 $n$ 元组(列)，且用小写的黑正体字母表示，例如，

$$\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_n]^T \tag{5.1}$$

这些都是$\Re^n$ 中的向量。$\Re^n$ 是标准的 $n$ 维欧基里德空间。这一章将讨论比欧基里德空间 $\Re^n$ 更一般的向量空间，这些更一般的向量将用手写体表示，比如 $\mathcal{X}$。同时本章还将说明这些一般向量通常是如何用一列数来表示的。

### 5.2.1　线性向量空间

到底什么是向量空间？这里要给出一个非常一般的定义。这个定义看起来比较抽象，我们将给出很多具体的实例。应用这一普遍的定义不仅可以解决一大类问题，而且还可以使读者更加深入地理解这一概念。

**定义**　一个线性向量空间 $X$ 是一组定义在标量域 $F$ 上且满足如下条件的元素集合(向量)：

5-2

1) 一个称为向量加的操作定义为：如果 $\mathcal{X} \in X$($\mathcal{X}$ 是 $X$ 的一个元素)和 $\mathcal{Y} \in X$，那么 $\mathcal{X} + \mathcal{Y} \in X$。
2) $\mathcal{X} + \mathcal{Y} = \mathcal{Y} + \mathcal{X}$。
3) $(\mathcal{X} + \mathcal{Y}) + \mathcal{Z} = \mathcal{X} + (\mathcal{Y} + \mathcal{Z})$。

4) 存在惟一一个称为零向量的向量 $0 \in X$，对于所有的 $x \in X$，有：$x + 0 = x$。
5) 对于每一个向量 $x \in X$，在 $X$ 中只有惟一一个被称为 $-x$ 的向量，满足 $x + (-x) = 0$ 。
6) 一个称为向量乘的操作定义为：对所有 $a \in F$ 的标量，以及所有的向量 $x \in X$，有 $ax \in X$。
7) 对于任意的 $x \in X$ 和标量 1，有 $1x = x$。
8) 对于任意两个标量 $a \in F$ 和 $b \in F$，以及任意的 $x \in X$，有 $a(bx) = (ab)x$。
9) $(a + b)x = ax + bx$。
10) $a(x + y) = ax + ay$。

为了说明上述条件，这里将给出一些例子，并且确定它们是否为向量空间。首先考虑二维的欧基里德空间$\Re^2$，如图 5-1 所示。显然它是一个向量空间，并且对于向量加和标量乘操作的标准定义而言，全部满足上述 10 个条件。

$\Re^2$ 的子集又将如何？$\Re^2$ 的什么子集仍然是向量空间(子空间)？考虑图 5-2 中方框内的区域 $X$。它可以全部满足上述 10 个条件吗？显然该区域连条件 1 都不能满足。如图 5-2 所示，向量 $x$ 和 $y$ 在 $X$ 的区域内，但是 $x + y$ 却可能不在 $X$ 的区域内。从这个例子可以看出，任何限定边界的集合都不可能是向量空间。

那么$\Re^2$ 存在是向量空间的任何子集吗？考虑图 5-3 中的直线 $X$(假设该线两端均为无限长)，那么这条线是向量空间吗？将这个问题留给读者，请读者证明此直线的确满足上述所有 10 个条件。那么是否所有这种无限长的直线都满足上述 10 个条件？实际上，所有经过坐标轴原点的直线都满足上述 10 个条件。但是，如果直线不经过坐标轴的原点，那么至少这种直线不能满足第 4 个条件。

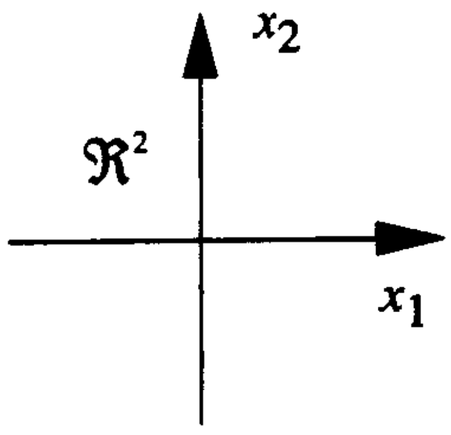



图 5-1

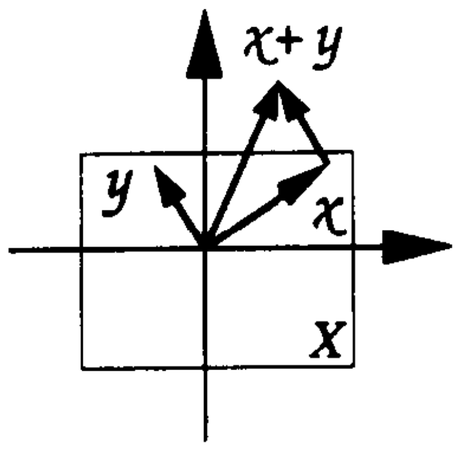



图 5-2

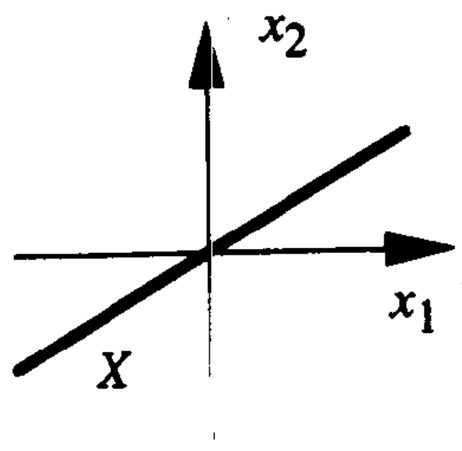



图 5-3

除了标准的欧基里德空间之外，还有许多其他的集合同样满足向量空间的 10 个条件。例如考虑最高阶数小于或等于 2 的多项式集合 $P^2$。此集合的两个元素是 5-3

$$\begin{aligned} x &= 2 + t + 4t^2 \\ y &= 1 + 5t \end{aligned} \tag{5.2}$$

如果读者已经习惯于将向量看作是一列数字，那么这两个元素的确是奇怪的向量。但是请记住：一个集合只要满足上述 10 个条件，就可以被认为是一个向量空间。那么集合 $P^2$ 是否也完全满足上述条件呢？如果将两个阶数小于或等于 2 的多项式相加，其结果仍然是一个阶数小于或等于 2 的多项式。因此，集合 $P^2$ 满足上述第 1 个条件。另外，将一个标量和一个多项式相乘，是不会改变该多项式的阶数的，所以集合 $P^2$ 满足上述第 6 个条件。显然，验证集合 $P^2$ 满足上述 10 个条件并不是一件困难的事，集合 $P^2$ 的确是一个向量空间。

假设 $C_{[0,1]}$是定义在[0, 1] 区间上的所有连续函数的集合，该集合的两个元素是

$$\begin{aligned} x &= \sin(t) \\ y &= e^{-2t} \end{aligned} \tag{5.3}$$

集合的另一个元素如图 5-4 所示。

由于两个连续函数的和仍然是一个连续函数，一个标量乘以一个连续函数仍然是一个连续函数，所以集合 $C_{[0,1]}$也是一个向量空间。这个集合与前面讨论过的向量空间不同，它是无限维的。本章后面将定义维的含义。

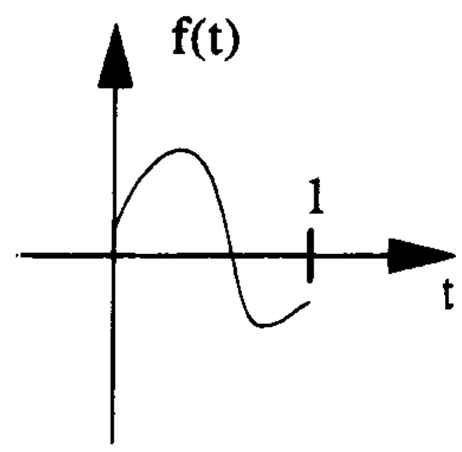



图 5-4

### 5.2.2 线性无关

前面已经给出了向量空间的定义，从现在开始将研究向量的一些性质。这里要研究的第一个性质就是向量的线性无关性和线性相关性。

如果对 $n$ 个向量$\{x_1, x_2, \cdots, x_n\}$而言，存在 $n$ 个标量$a_1, a_2, \cdots, a_n$(这 $n$ 个标量中至少有一个是非零的)，满足

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = 0 \tag{5.4}$$

那么$\{x_i\}$是线性相关的。

**线性无关** 与之相反，如果 $a_1x_1 + a_2x_2 + \cdots + a_nx_n = 0$ ，当且仅当每个 $a_i$ 均等于零，那么称$\{x_i\}$是一组线性无关的向量。

5-4

注意这些定义实际上等价于：如果一个向量集合是无关的，那么这个集合中的任何向量都不能表示成该集合中其他向量的线性组合。

作为一个线性无关的实例，考虑第 3 章中的模式识别问题。两个标准模式(橘子和苹果)由如下两个向量表示：

$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} \tag{5.5}$$

令 $a_1\mathbf{p}_1 + a_2\mathbf{p}_2 = 0$，则有

$$\begin{bmatrix} a_1 + a_2 \\ -a_1 + a_2 \\ -a_1 + (-a_2) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.6}$$

但式(5.6)只有当 $a_1 = a_2 = 0$ 时成立。所以 $\mathbf{p}_1$ 与 $\mathbf{p}_2$ 线性无关。

现在考虑阶数小于等于 2 的多项式空间 $P^2$ 中的向量。设该空间的中的三个向量分别是

$$x_1 = 1 + t + t^2, x_2 = 2 + 2t + t^2, x_3 = 1 + t \tag{5.7}$$

如果令 $a_1 = 1$，$a_2 = -1$，$a_3 = 1$，那么

$$a_1x_1 + a_2x_2 + a_3x_3 = 0 \tag{5.8}$$

所以，这三个向量线性相关。

### 5.2.3 生成空间

下面将定义对向量空间的维数(大小)进行定义。为此，首先给出一个生成集合的概念。

假设 $X$ 是一个线性空间，且$\{\mathcal{U}_1, \mathcal{U}_2, \cdots, \mathcal{U}_m\}$是 $X$ 中一般向量的子集。该子集能够生成 $X$，当且仅当对每一个 $\mathcal{X} \in X$，都存在一组标量 $x_1, x_2, \cdots, x_m$，满足 $\mathcal{X} = x_1\mathcal{U}_1 + x_2\mathcal{U}_2 + \cdots + x_m\mathcal{U}_m$。也就是说，如果空间中的每个向量都能写成该子集中向量的线性组合，那么这个子集就能够生成一个空间。

**基集**　一个向量空间的维数是由生成该空间所需要的最少向量个数决定的。由此导出了基集的概念。$X$ 的基集是由生成 $X$ 的线性无关的向量所组成的集合。任何基集包含了生成空间所需要的最少个数的向量。因此 $X$ 的维数就等于基集中元素的个数。任何向量空间都可以有多个基集，但每一个基集都必须包含相同数目的元素(请参考[Stra80]中的有关证明)。 5-5

以线性空间 $P^2$ 为例，该空间的一个可能的基是：

$$\mathcal{U}_1 = 1, \mathcal{U}_2 = t, \mathcal{U}_3 = t^2 \tag{5.9}$$

显然任何一个阶数小于或等于 2 的多项式都可以通过这三个向量的线性组合表示。但请注意，$P^2$ 中的任意三个线性无关的向量都可以组成该空间的一个基。比如该空间的基也可以是：

$$\mathcal{U}_1 = 1, \mathcal{U}_2 = 1 + t, \mathcal{U}_3 = 1 + t + t^2 \tag{5.10}$$

### 5.2.4　内积

从第 3 章和第 4 章对神经网络的讨论中可以发现，内积是许多神经网络操作的基础。这里将介绍内积的一般定义，并给出相关的一些实例。

**内积**　任何满足如下列条件的关于 $\mathcal{X}$ 和 $\mathcal{Y}$ 的标量函数都可以定义为一个内积$(\mathcal{X}, \mathcal{Y})$：

1) $(\mathcal{X}, \mathcal{Y}) = (\mathcal{Y}, \mathcal{X})$；

2) $(\mathcal{X}, a\mathcal{Y}_1 + b\mathcal{Y}_2) = a(\mathcal{X}, \mathcal{Y}_1) + b(\mathcal{X}, \mathcal{Y}_2)$；

3) $(\mathcal{X}, \mathcal{X}) \geqslant 0$，当且仅当 $\mathcal{X}$ 是零向量时$(\mathcal{X}, \mathcal{X}) = 0$。

对于$\mathfrak{R}^n$ 中的向量而言，其标准内积为

$$\mathbf{x}^T\mathbf{y} = x_1y_1 + x_2y_2 + \cdots + x_ny_n \tag{5.11}$$

但是这并不是惟一可能的内积形式。比如，对定义在[0, 1]区间内所有连续函数的集合 $C_{[0,1]}$而言，下面给出的标量函数(式 5.12)就是它的一种内积形式(请参见例题 P5.6)。

$$(\mathcal{X}, \mathcal{Y}) = \int_0^1 \mathcal{X}(t)\mathcal{Y}(t)\,dt \tag{5.12}$$ 5-6

### 5.2.5　范数

**范数**　我们要定义的下一个操作是范数，它是一个基于向量长度概念的操作。如果一个标量函数 $\|\mathcal{X}\|$ 满足以下一些性质，则称其为范数：

1) $\|\mathcal{X}\| \geqslant 0$；

2) $\|\mathcal{X}\| = 0$，当且仅当 $\mathcal{X} = 0$；

3) 对所有的标量 $a$ 有 $\|a\mathcal{X}\| = |a|\|\mathcal{X}\|$；

4) $\|\mathcal{X} + \mathcal{Y}\| \leqslant \|\mathcal{X}\| + \|\mathcal{Y}\|$。

实际上，有很多函数都可以满足上述条件。一个普通的范数是基于内积按如下方式定义

的：

$$\|x\| = (x, x)^{1/2} \tag{5.13}$$

对于欧基里德空间$\Re^n$而言，其内积的定义为：

$$\|\mathbf{x}\| = (\mathbf{x}^T\mathbf{x})^{1/2}\sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} \tag{5.14}$$

在神经网络应用中，通常要将输入向量归一化，也即每个输入向量的$\|\mathbf{p}_i\| = 1$。

**角度** 利用上述范数和内积的定义，可以对维数大于2的向量空间的角度概念进行推广。这里可以将向量$x$和$y$之间的角度$\theta$定义为：

$$\cos\theta = \frac{(x, y)}{\|x\|\|y\|} \tag{5.15}$$

### 5.2.6 正交性

既然前面定义了内积操作，那么现在就可以介绍正交性这一重要的概念了。

**正交性** 如果两个向量$x$，$y \in X$，满足$(x, y) = 0$，那么说这两个向量是正交的。

正交性是神经网络中的一个重要概念。在第7章中读者将会看到，当一个模式识别问题的模式向量是归一化的和正交的，那么利用Hebb规则对一个线性联想器神经网络进行训练，可以得到很好的识别效果。

5-7 除了有正交的向量之外，还可以有正交的向量空间。如果向量$x \in X$正交于子空间$X_1$中的每一个向量，则$x$正交于子空间$X_1$，通常将其记为$x \perp X_1$。如果子空间$X_1$中的每一个向量都正交于子空间$X_2$中的每一个向量，则子空间$X_1$正交与子空间$X_2$，对此用$X_1 \perp X_2$来表示。

图5-5给出了第3章感知机实例中(参见图3-4)所用到的两个正交空间。$p_1$，$p_3$平面是$\Re^3$的子空间，该平面与$p_2$轴($\Re^3$的另外一个子空间)正交。$p_1$，$p_3$平面是感知机网络的判定边界。在例题P5.1中，读者将会看到：当偏置值为零时，感知机的判定边界是一个向量空间。

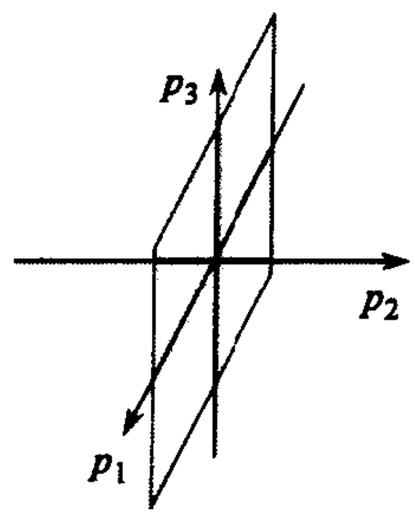


图5-5

Gram-Schmidt 正交化方法

线性无关和正交性是相互联系的。可以将线性无关向量集合转换为一个正交向量集合，而且两者所生成的向量空间是相同的。这个标准的转换过程被称为Gram-Schmidt正交化方法。

假设有$n$个线性无关的向量$y_1$，$y_2$，…，$y_n$，如果希望通过这些向量得到$n$个正交向量$v_1$，$v_2$，…，$v_n$，那么可以首先选择第一个线性无关向量作为第一个正交向量：

$$v_1 = y_1 \tag{5.16}$$

为了得到第二个正交向量，可以将$y_2$减去处于$v_1$方向上的分量。据此，可以得到下式：

$$v_2 = y_2 - av_1 \tag{5.17}$$

其中$a$必须选择合适的值，使$v_1$正交于$v_2$，也即：

$$(v_1, v_2) = (v_1, y_2 - av_1) = (v_1, y_2) - a(v_1, v_1) = 0 \tag{5.18}$$

或

$$a = \frac{(v_1, y_2)}{(v_1, v_1)} \tag{5.19}$$

**投影**　因此，为了得到 $\mathcal{y}_2$ 在 $\mathcal{v}_1$ 方向上的分量 $a\mathcal{v}_1$，需要求这两个向量的内积。也称 $a\mathcal{v}_1$ 是 $\mathcal{y}_2$ 在向量 $\mathcal{v}_1$ 上的投影。

如果继续这一过程，那么第 $k$ 步是

$$\mathcal{v}_k = \mathcal{y}_k - \sum_{i=1}^{k-1} \frac{(\mathcal{v}_i, \mathcal{y}_k)}{(\mathcal{v}_i, \mathcal{v}_i)} \mathcal{v}_i \tag{5.20}$$

5-8

为了具体说明这个过程，请考虑下面在空间 $\Re^2$ 中的线性无关向量：

$$\mathbf{y}_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \quad \mathbf{y}_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{5.21}$$

第一个正交向量为：

$$\mathbf{v}_1 = \mathbf{y}_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \tag{5.22}$$

第二个正交向量的计算如下所示：

$$\mathbf{v}_2 = \mathbf{y}_2 - \frac{\mathbf{v}_1^T \mathbf{y}_2}{\mathbf{v}_1^T \mathbf{v}_1} \mathbf{v}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} - \frac{[2 \quad 1]\begin{bmatrix} 1 \\ 2 \end{bmatrix}}{[2 \quad 1]\begin{bmatrix} 2 \\ 1 \end{bmatrix}} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} - \begin{bmatrix} 1.6 \\ 0.8 \end{bmatrix} = \begin{bmatrix} -0.6 \\ 1.2 \end{bmatrix} \tag{5.23}$$

这一过程可以用图 5-6 来表示。

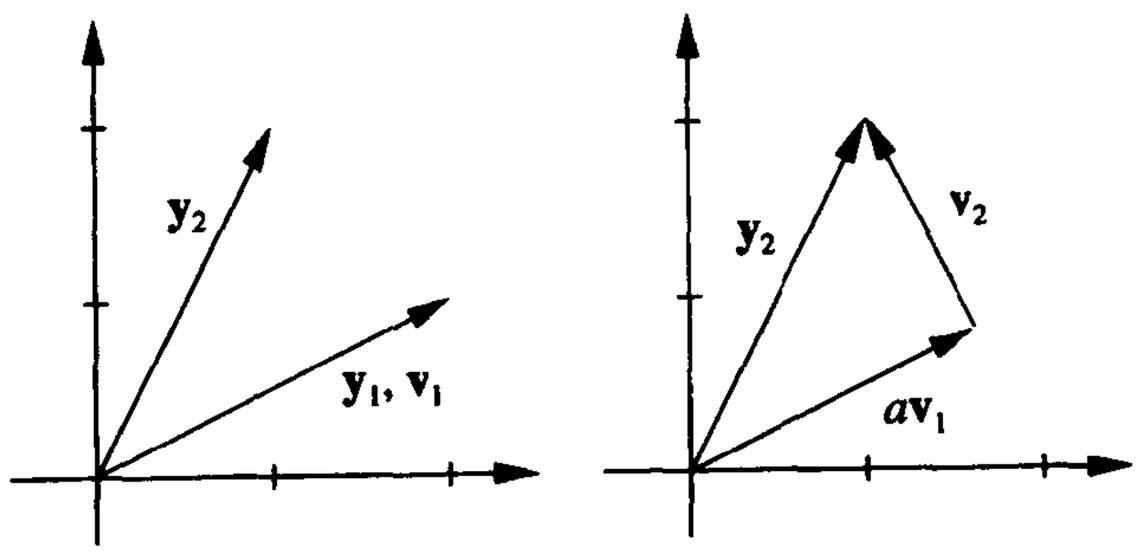



图 5-6　Gram-Schmidt 正交化实例

**标准正交向量**　另外，还可以将向量 $\mathbf{v}_1$，$\mathbf{v}_2$ 分别除以其范数，从而得到一个标准正交向量集。

验证正交化过程可使用 *Neural Network Design Demonstration* **Gram-Schmidt** (**nnd5gs**)。

### 5.2.7　向量展开式

请注意：前面用手写体字符($\mathcal{x}$)表示一般的向量，用黑体($\mathbf{x}$)来表示 $\Re^n$ 中的向量，而 $\Re^n$ 中的向量也可以用一列数的形式来表示。本节将说明有限维空间中的一般向量也可以表示为一列数的形式，并且这些一般向量在某些方面和 $\Re^n$ 中的向量是等价的。 5-9

**向量展开式**　如果向量空间 $X$ 的基集为 $\{\mathcal{v}_1, \mathcal{v}_2, \cdots, \mathcal{v}_n\}$，那么任意 $\mathcal{x} \in X$ 有如下

惟一的向量展开式：

$$\mathcal{X} = \sum_{i=1}^{n} x_i \mathcal{V}_i = x_1\mathcal{V}_1 + x_2\mathcal{V}_2 + \cdots + x_n\mathcal{V}_n \tag{5.24}$$

所以，有限维向量空间中的任意向量都可以用一列数来表示：

$$\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_n]^T \tag{5.25}$$

这里的 $\mathbf{x}$ 表示一般的向量 $\mathcal{X}$。当然为了解释 $\mathbf{x}$ 的含义，还需要知道基集是什么。对同一个 $\mathcal{X}$ 而言。如果基集发生了变化，那么 $\mathbf{x}$ 也随着发生变化。下一小节将对此进行更详细的讨论。

如果基集中的向量是正交的(即$(\mathcal{V}_i, \mathcal{V}_j)=0$，$i \neq j$)，那么可以非常容易计算出上述展开式中的系数，只要在式(5.24)两边求与 $\mathcal{V}_j$ 的内积即可：

$$(\mathcal{V}_j, \mathcal{X}) = (\mathcal{V}_j, \sum_{i=1}^{n} x_i\mathcal{V}_i) = \sum_{i=1}^{n} x_i(\mathcal{V}_j, \mathcal{V}_i) = x_j(\mathcal{V}_j, \mathcal{V}_j) \tag{5.26}$$

所以，上述展开式中的系数由下式给出：

$$x_j = \frac{(\mathcal{V}_j, \mathcal{X})}{(\mathcal{V}_j, \mathcal{V}_j)} \tag{5.27}$$

当基集中的向量不正交时，计算上述展开式中的系数要相对复杂一些。在下一小节中将会介绍这种情况。

### 互逆基向量

**互逆基向量** 如果需要向量展开式，而基集又不是正交的，那么就必须引入由下列等式所定义的互逆基底：

5-10

$$(r_i, \mathcal{V}_j) = \begin{cases} 0, i \neq j \\ 1, i = j \end{cases} \tag{5.28}$$

其中基向量为$\{\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_n\}$，而互逆基向量为$\{r_1, r_2, \cdots, r_n\}$。

如果互逆基向量已经表示为一列数的形式(通过向量展开式)，并且采用了标准内积

$$(r_i, \mathcal{V}_j) = \mathbf{r}_i^T\mathbf{v}_j \tag{5.29}$$

那么，式(5.28)可以用矩阵的形式表示为：

$$\mathbf{R}^T\mathbf{B} = \mathbf{I} \tag{5.30}$$

其中

$$\mathbf{B} = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \cdots \quad \mathbf{v}_n] \tag{5.31}$$

$$\mathbf{R} = [\mathbf{r}_1 \quad \mathbf{r}_2 \quad \cdots \quad \mathbf{r}_n] \tag{5.32}$$

所以，可以从下式求得 $\mathbf{R}$：

$$\mathbf{R}^T = \mathbf{B}^{-1} \tag{5.33}$$

最后可根据 $\mathbf{R}$ 的列求得互逆基向量。

现在请考虑向量展开式

$$\mathcal{X} = x_1\mathcal{V}_1 + x_2\mathcal{V}_2 + \cdots + x_n\mathcal{V}_n \tag{5.34}$$

求式(5.34)的两边和 $r_1$ 之间的内积：

$$(r_1, \mathcal{X}) = x_1(r_1, \mathcal{V}_1) + x_2(r_1, \mathcal{V}_2) + \cdots + x_n(r_1, \mathcal{V}_n) \tag{5.35}$$

根据定义得

$$(r_1, \mathscr{V}_2) = (r_1, \mathscr{V}_3) = \cdots = (r_1, \mathscr{V}_n) = 0$$
$$(r_1, \mathscr{V}_1) = 1 \tag{5.36}$$

所以，上述展开式中的第一个系数是

$$x_1 = (r_1, \mathscr{X}) \tag{5.37}$$

一般情况下，展开式中的第 $j$ 个系数为

$$x_j = (r_j, \mathscr{X}) \tag{5.38}$$

5-11

现在请考虑如下实例，设有两个基向量：

$$\mathbf{v}_1^S = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \qquad \mathbf{v}_2^S = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \tag{5.39}$$

其中，上标 $S$ 表示这两个列向量是按 $\mathfrak{R}^2$ 中的标准基向量展开的结果。$\mathfrak{R}^2$ 中的标准基向量如图 5-7 所示，分别为图中的向量 $s_1$ 和 $s_2$。

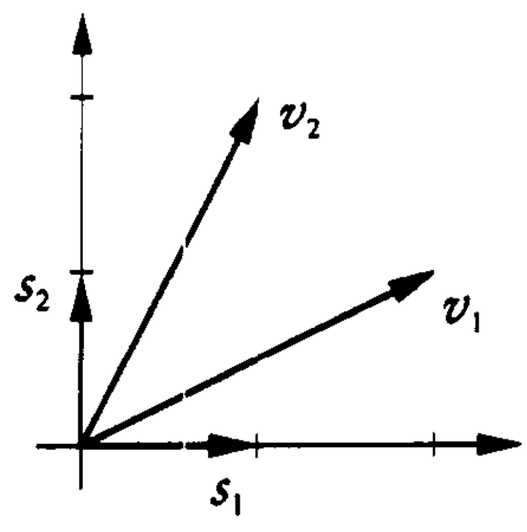


图 5-7

现假设要用这两个基向量对下面的向量进行展开：

$$\mathbf{x}^S = \begin{bmatrix} 0 \\ \frac{3}{2} \end{bmatrix} \tag{5.40}$$

由于要按照两个不同的基集对向量进行展开，所以这里必须要注意各个数学符号都要明确地标注。

展开该向量的第一步是找到互逆基向量：

$$\mathbf{R}^T = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}^{-1} = \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} \end{bmatrix}, \mathbf{r}_1 = \begin{bmatrix} \frac{2}{3} \\ -\frac{1}{3} \end{bmatrix}, \mathbf{r}_2 = \begin{bmatrix} -\frac{1}{3} \\ \frac{2}{3} \end{bmatrix} \tag{5.41}$$

下面求展开式中的系数：

$$x_1^v = \mathbf{r}_1^T \mathbf{x}^S = \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} \end{bmatrix} \begin{bmatrix} 0 \\ \frac{3}{2} \end{bmatrix} = -\frac{1}{2}$$
$$x_2^v = \mathbf{r}_2^T \mathbf{x}^S = \begin{bmatrix} -\frac{1}{3} & \frac{2}{3} \end{bmatrix} \begin{bmatrix} 0 \\ \frac{3}{2} \end{bmatrix} = 1 \tag{5.42}$$

或按矩阵形式写成

$$\mathbf{x}^v = \mathbf{R}^T \mathbf{x}^S = \mathbf{B}^{-1} \mathbf{x}^S = \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} \end{bmatrix} \begin{bmatrix} 0 \\ \frac{3}{2} \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} \\ 1 \end{bmatrix} \tag{5.43}$$

5-12

于是有(如图 5-8 所示)：

$$\mathscr{X} = -\frac{1}{2}\mathscr{V}_1 + 1\mathscr{V}_2 \tag{5.44}$$

注意现在有两种 $\mathscr{X}$ 的展开式，分别由 $\mathbf{x}^S$ 和 $\mathbf{x}^v$ 表示，即是

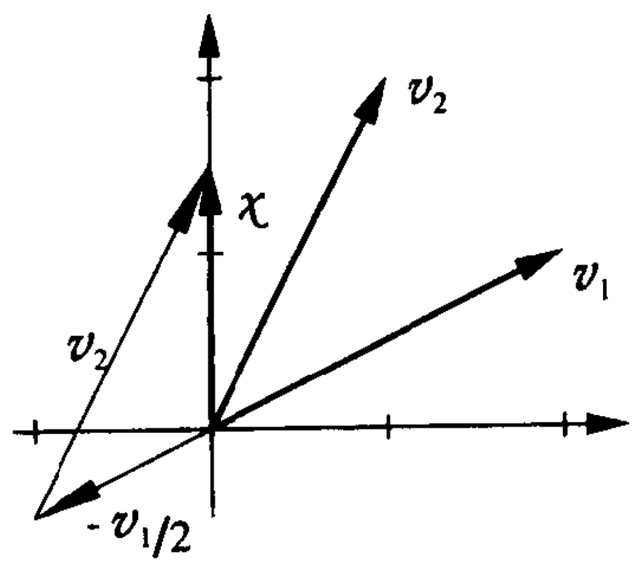



图 5-8 向量展开

$$x = 0s_1 + \frac{3}{2}s_2 = -\frac{1}{2}v_1 + 1v_2 \tag{5.45}$$

由此可以看出，当要用一列数字表示一个一般向量时，必须知道其向量展开式所采用的基集是什么。在本书中如果没有特殊说明，那么假设所采用的都是标准基集。

式(5.43)说明了 $x$ 的两种不同表示方式之间的关系：$\mathbf{x}^v = \mathbf{B}^{-1}\mathbf{x}^S$。这一操作也称为基变换。在后面几章某些神经网络性能分析中，基变换非常重要。

5-13 验证向量展开过程可使用 *Neural Network Design Demonstration* **Reciprocal Basis** (**nnd5rb**)。

## 5.3 小结

### 线性向量空间

**定义** 一个线性向量空间 $X$ 是一组定义在标量域 $F$ 上且满足如下条件的元素(向量)集合：

1) 定义一个称为向量加的操作：如果 $x \in X$($x$ 是 $X$ 的一个元素)，且 $y \in X$，那么 $x + y \in X$。
2) $x + y = y + x$。
3) $(x + y) + z = x + (y + z)$。
4) 存在惟一一个称为零向量的向量 $0 \in X$，对于所有的 $x \in X$，有 $x + 0 = x$。
5) 对于每一个向量 $x \in X$，在 $X$ 中只有惟一一个被称为 $-x$ 的向量，满足 $x + (-x) = 0$。
6) 定义一个称为乘的操作：对所有标量 $a \in F$ 以及所有向量 $x \in X$，有 $ax \in X$。
7) 对于任意的 $x \in X$ 和标量 1，有 $1x = x$。
8) 对于任意两个标量 $a \in F$ 和 $b \in F$，以及任意的 $x \in X$，有 $a(bx) = (ab)x$。
9) $(a + b)x = ax + bx$。
10) $a(x + y) = ax + ay$。

### 线性无关

如果对 $n$ 个向量 $\{x_1, x_2, \cdots, x_n\}$ 而言，存在 $n$ 个标量 $a_1, a_2, \cdots, a_n$(其中至少有

一个不是零)，满足

$$a_1\mathcal{X}_1 + a_2\mathcal{X}_2 + \cdots + a_n\mathcal{X}_n = 0$$

那么$\{\mathcal{X}_i\}$是线性无关的。 5-14

**生成空间**

假设 $X$ 是一个线性向量空间，且$\{\mathcal{U}_1, \mathcal{U}_2, \cdots, \mathcal{U}_m\}$是 $X$ 中的向量的子集。该子集能够生成 $X$，当且仅当对每一个 $\mathcal{X}\in X$，都存在一组标量 $x_1$，$x_2$，…，$x_m$，满足 $\mathcal{X} = x_1\mathcal{U}_1 + x_2\mathcal{U}_2 + \cdots + x_m\mathcal{U}_m$。

**内积**

任何满足下列条件的关于 $\mathcal{X}$ 和 $\mathcal{Y}$ 的标量函数都可以定义为一个内积$(\mathcal{X},\mathcal{Y})$：

1) $(\mathcal{X},\mathcal{Y}) = (\mathcal{Y},\mathcal{X})$；

2) $(\mathcal{X}, a\mathcal{Y}_1 + b\mathcal{Y}_2) = a(\mathcal{X},\mathcal{Y}_1) + b(\mathcal{X},\mathcal{Y}_2)$；

3) $(\mathcal{X},\mathcal{X}) \geqslant 0$，当且仅当 $\mathcal{X}$ 是零向量时$(\mathcal{X},\mathcal{X}) = 0$。

**范数**

如果一个标量函数 $\|\mathcal{X}\|$ 满足以下一些性质，则将其称之为范数：

1) $\|\mathcal{X}\| \geqslant 0$；

2) $\|\mathcal{X}\| = 0$，当且仅当 $\mathcal{X} = 0$；

3) 对所有的标量 $a$ 有 $\|a\mathcal{X}\| = |a|\|\mathcal{X}\|$；

4) $\|\mathcal{X} + \mathcal{Y}\| \leqslant \|\mathcal{X}\| + \|\mathcal{Y}\|$。

**向量之间的夹角**

向量 $\mathcal{X}$ 和 $\mathcal{Y}$ 之间的角度 $\theta$ 定义为：

$$\cos\theta = \frac{(\mathcal{X},\mathcal{Y})}{\|\mathcal{X}\|\|\mathcal{Y}\|}$$

**正交性**

如果两个向量 $\mathcal{X},\mathcal{Y}\in X$ 满足$(\mathcal{X},\mathcal{Y}) = 0$，那么说这两个向量是正交的。

**Gram-Schmidt 正交化方法**

假设有 $n$ 个线性无关的向量 $\mathcal{Y}_1$，$\mathcal{Y}_2$，…，$\mathcal{Y}_n$。根据这些向量得到 $n$ 个正交向量 $\mathcal{V}_1$，$\mathcal{V}_2$，…，$\mathcal{V}_n$：

$$\mathcal{V}_1 = \mathcal{Y}_1$$

$$\mathcal{V}_k = \mathcal{Y}_k - \sum_{i=1}^{k-1}\frac{(\mathcal{V}_i,\mathcal{Y}_k)}{(\mathcal{V}_i,\mathcal{V}_i)}\mathcal{V}_i$$ 5-15

其中$\frac{(\mathcal{V}_i,\ \mathcal{Y}_k)}{(\mathcal{V}_i,\ \mathcal{V}_i)}\mathcal{V}_i$ 是 $\mathcal{Y}_k$ 在 $\mathcal{V}_i$ 上的投影。

**向量展开式**

$$\mathcal{X} = \sum_{i=1}^{n} x_i\mathcal{V}_i = x_1\mathcal{V}_1 + x_2\mathcal{V}_2 + \cdots + x_n\mathcal{V}_n$$

对正交向量而言，

$$x_j = \frac{(\mathcal{V}_j, \mathcal{X})}{(\mathcal{V}_j, \mathcal{V}_j)}$$

**互逆基向量**

$$(r_i, \mathcal{V}_j) = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}$$

$$x_j = (r_j, \mathcal{X})$$

为了计算互逆基向量，可采用如下方法：

$$\mathbf{B} = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \cdots \quad \mathbf{v}_n]$$

$$\mathbf{R} = [\mathbf{r}_1 \quad \mathbf{r}_2 \quad \cdots \quad \mathbf{r}_n]$$

$$\mathbf{R}^T = \mathbf{B}^{-1}$$

也可用矩阵形式表示为

5-16

$$\mathbf{x}^v = \mathbf{B}^{-1}\mathbf{x}^S$$

## 5.4 例题

**P5.1** 考虑如图 5-9 中所示的单神经元感知机网络。第 3 章中给出了该网络的判定边界为(参见式(3.6))：$\mathbf{Wp} + b = 0$。试证明：若 $b = 0$，那么判定边界是一个向量空间。

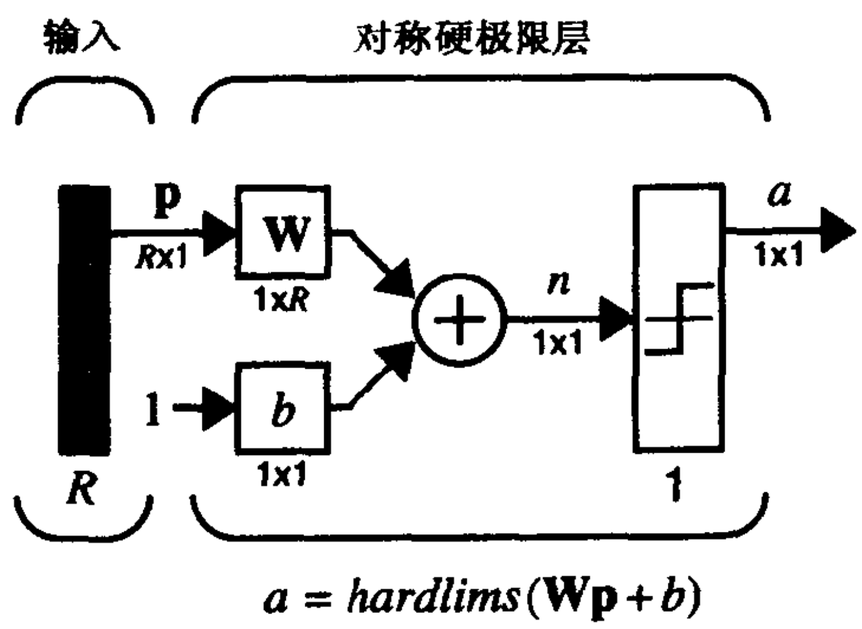



图 5-9 单神经元感知机

**解**

该判定边界如果是一个向量空间就必须满足本章开始所给出的 10 个条件。条件 1 要求两个向量空间之和仍然是一个向量空间。令 $\mathbf{p}_1$ 和 $\mathbf{p}_2$ 分别是判定边界上的两个向量，它们一定满足：

$$\mathbf{Wp}_1 = 0, \mathbf{Wp}_2 = 0$$

将上述两个等式相加，有

$$\mathbf{W}(\mathbf{p}_1 + \mathbf{p}_2) = 0$$

由此可以看出这两个向量之和也在判定边界上。

显然，判定边界能够满足条件 2 和条件 3。条件 4 要求零向量在判定边界上。由于 $\mathbf{W0} = 0$，所以零向量在判定边界上。条件 5 则意味着：如果 $\mathbf{p}$ 在判定边界上，那么 $-\mathbf{p}$ 也必须在判定边界上。如果 $\mathbf{p}$ 在判定边界上，那么

$$\mathbf{Wp} = 0$$

在该式两边同时乘以 -1，可得

$$\mathbf{W}(-\mathbf{p}) = 0$$

所以判定边界也满足条件 5。 5-17

如果对判定边界上的任意 $\mathbf{p}$，$a\mathbf{p}$ 也在判定边界上，那么判定边界将满足条件 6。和条件 5 验证一样，将前面等式两边同时乘以 $a$，有

$$\mathbf{W}(a\mathbf{p}) = 0$$

据此可知判定边界也满足条件 6。

显然，条件 7 到条件 10 对判定边界而言也是满足的。所以该感知机的判定边界是一个向量空间。

**P5.2** 证明非负连续函数($f(t) \geqslant 0$)集 $Y$ 不是一个向量空间。

**证**

这个集合违反了向量空间所需要的几个条件。比如，该集合不存在负向量，从而它不能满足条件 5。同样，考虑条件 6，由于函数 $f(t) = |t|$ 是集合 $Y$ 的一个元素，令 $a = -2$，则有

$$af(2) = -2|2| = -4 < 0$$

因此，$af(t)$ 不是集合 $Y$ 的元素，使条件 6 不能满足。

**P5.3** 下面哪一组向量是线性无关的？请找出每个集合生成空间的维数。

(i) $\begin{bmatrix}1\\1\\1\end{bmatrix}$　$\begin{bmatrix}1\\0\\1\end{bmatrix}$　$\begin{bmatrix}1\\2\\1\end{bmatrix}$

(ii) $\sin t$　$\cos t$　$2\cos(t+\pi/4)$

(iii) $\begin{bmatrix}1\\1\\1\\1\end{bmatrix}$　$\begin{bmatrix}1\\0\\1\\1\end{bmatrix}$　$\begin{bmatrix}1\\2\\1\\1\end{bmatrix}$

**解**

(i) 求解这个问题有几种方法。首先假设这些向量是相关的，那么有

$$a_1\begin{bmatrix}1\\1\\1\end{bmatrix} + a_2\begin{bmatrix}1\\0\\1\end{bmatrix} + a_3\begin{bmatrix}1\\2\\1\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix}$$ 5-18

如果能够求得上式中的系数，且这些系数不全为 0，那么这些向量就是相关的。通过观察不难发现，$a_1 = 2$，$a_2 = -1$，$a_3 = -1$ 能够使上式成立，所以这些向量是相关的。

当在 $\mathfrak{R}^n$ 中有 $n$ 个向量时，求解此问题的另一种方法是以矩阵的形式将上式写成

$$\begin{bmatrix}1 & 1 & 1\\1 & 0 & 2\\1 & 1 & 1\end{bmatrix}\begin{bmatrix}a_1\\a_2\\a_3\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix}$$

如果上式中的矩阵存在逆，那么该等式的解要求所有的系数都是零。在这种情况下，这

些向量是线性无关的。如果矩阵是一个奇异矩阵(不存在逆)，那么必然可以求得满足该等式的一个非 0 系数集合。在这种情况下，这些向量是线性相关的。所以，可以以这些向量为列构造一个矩阵。如果该矩阵的行列式为 0(奇异矩阵)，那么这些向量就是相关的。否则，它们是线性无关的。将矩阵的第一列用 Laplace 展开式[Brog91]展开，有

$$\begin{vmatrix} 1 & 1 & 1 \\ 1 & 0 & 2 \\ 1 & 1 & 1 \end{vmatrix} = 1\begin{vmatrix} 0 & 2 \\ 1 & 1 \end{vmatrix} + (-1)\begin{vmatrix} 1 & 1 \\ 1 & 1 \end{vmatrix} + 1\begin{vmatrix} 1 & 1 \\ 0 & 2 \end{vmatrix} = -2+0+2=0$$

所以这些向量是相关的。

另外，由于可以证明这三个向量中的任意两个向量都是线性无关的，所以由这三个向量张成的向量空间的维数为 2。

(ii) 根据一些三角等式，有

$$\cos\left(t+\frac{\pi}{4}\right) = \frac{-1}{\sqrt{2}}\sin t + \frac{1}{\sqrt{2}}\cos t$$

所以，这些向量也是相关的。由于 $\sin t$ 和 $\cos t$ 的任何线性组合都不等于 0，所以这些向量所生成的空间的维数是 2。

(iii) 这与(i)题相似，只是向量个数比这些向量的原始空间中向量个数要少(只有$\Re^4$空间中的 3 个向量)。在这种情况下，由这 3 个向量所构成的矩阵不再是一个方阵，所以不能计算其行列式的值。不过可以采用称为 Gram 的方法[Brog91]，这种方法可以求出一个矩阵的行列式，矩阵的第 $i$ 行第 $j$ 列的元素是向量 $i$ 和向量 $j$ 的内积。这些向量是线性相关的，当且仅当 Gram 矩阵的行列式为零。这里的 Gram 行列式为：

5-19

$$G = \begin{vmatrix} (\mathbf{x}_1,\mathbf{x}_1) & (\mathbf{x}_1,\mathbf{x}_2) & (\mathbf{x}_1,\mathbf{x}_3) \\ (\mathbf{x}_2,\mathbf{x}_1) & (\mathbf{x}_2,\mathbf{x}_2) & (\mathbf{x}_2,\mathbf{x}_3) \\ (\mathbf{x}_3,\mathbf{x}_1) & (\mathbf{x}_3,\mathbf{x}_2) & (\mathbf{x}_3,\mathbf{x}_3) \end{vmatrix}$$

其中

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad \mathbf{x}_2 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \quad \mathbf{x}_3 = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 1 \end{bmatrix}$$

所以

$$G = \begin{vmatrix} 4 & 3 & 5 \\ 3 & 3 & 3 \\ 5 & 3 & 7 \end{vmatrix} = 4\begin{vmatrix} 3 & 3 \\ 3 & 7 \end{vmatrix} + (-3)\begin{vmatrix} 3 & 5 \\ 3 & 7 \end{vmatrix} + 5\begin{vmatrix} 3 & 5 \\ 3 & 3 \end{vmatrix} = 48-18-30=0$$

同样，也可以按如下方法证明这些向量是线性相关的：

$$2\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} - 1\begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} - 1\begin{bmatrix} 1 \\ 2 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

另外，这些向量生成空间的维数一定小于 3。可以证明 $\mathbf{x}_1$ 和 $\mathbf{x}_2$ 是线性无关的，因为

$$G = \begin{vmatrix} 4 & 3 \\ 3 & 3 \end{vmatrix} = 4 \neq 0$$

所以这些向量生成空间的维数为 2。

**P5.4** 在第 3 章和第 4 章曾经讨论过单层感知机只适用于识别一组线性可分的模式(参见图 3-3 中的线性边界)。那么请问，如果两个模式是线性可分的，它们一定是线性无关的吗？

**答**

不是。这是两个没有任何关联的概念。比如，考虑如图 5-10 所示的两输入感知机。假设现在希望区分如下两个向量：

5-20

$$\mathbf{p}_1 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \qquad \mathbf{p}_2 = \begin{bmatrix} 1.5 \\ 1.5 \end{bmatrix}$$

如果将权值和偏置值分别设定为 $w_{11} = 1$，$w_{12} = 1$ 和 $b = -2$，那么其判定边界如图 5-11 所示。显然，这两个向量是线性可分的。但是，由于 $\mathbf{p}_2 = 3\mathbf{p}_1$，它们之间并不是线性无关的。

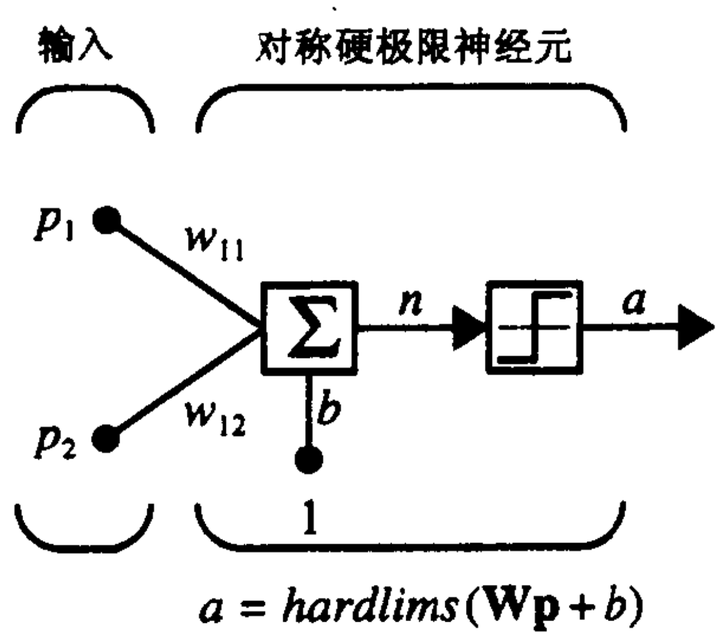



图 5-10　两输入感知机

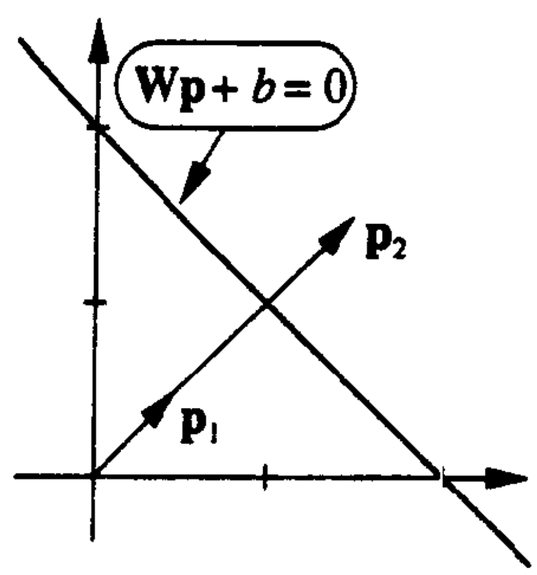



图 5-11　判定边界

**P5.5** 用 Gram-Schmidt 正交化方法，求如下基向量的正交集。

$$\mathbf{y}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad \mathbf{y}_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{y}_3 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

**解**

第 1 步：

$$\mathbf{v}_1 = \mathbf{y}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

第 2 步：

5-21

$$\mathbf{v}_2 = \mathbf{y}_2 - \frac{\mathbf{v}_1^T \mathbf{y}_2}{\mathbf{v}_1^T \mathbf{v}_1}\mathbf{v}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - \frac{\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}}{\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} = \begin{bmatrix} 2/3 \\ -1/3 \\ -1/3 \end{bmatrix}$$

第 3 步：

$$\mathbf{v}_3 = \mathbf{y}_3 - \frac{\mathbf{v}_1^T\mathbf{y}_3}{\mathbf{v}_1^T\mathbf{v}_1}\mathbf{v}_1 - \frac{\mathbf{v}_2^T\mathbf{y}_3}{\mathbf{v}_2^T\mathbf{v}_2}\mathbf{v}_2$$

$$\mathbf{v}_3 = \begin{bmatrix}0\\1\\0\end{bmatrix} - \frac{\begin{bmatrix}1 & 1 & 1\end{bmatrix}\begin{bmatrix}0\\1\\0\end{bmatrix}}{\begin{bmatrix}1 & 1 & 1\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix}}\begin{bmatrix}1\\1\\1\end{bmatrix} - \frac{\begin{bmatrix}2/3 & -1/3 & -1/3\end{bmatrix}\begin{bmatrix}0\\1\\0\end{bmatrix}}{\begin{bmatrix}2/3 & -1/3 & -1/3\end{bmatrix}\begin{bmatrix}2/3\\-1/3\\-1/3\end{bmatrix}}\begin{bmatrix}2/3\\-1/3\\-1/3\end{bmatrix}$$

$$\mathbf{v}_3 = \begin{bmatrix}0\\1\\0\end{bmatrix} - \begin{bmatrix}1/3\\1/3\\1/3\end{bmatrix} - \begin{bmatrix}-1/3\\1/6\\1/6\end{bmatrix} = \begin{bmatrix}0\\1/2\\-1/2\end{bmatrix}$$

**P5.6** 考虑由定义在区间[ - 1, 1]上的所有多项式所构成的向量空间。试证明$(\mathscr{x}, \mathscr{y}) = \int_{-1}^{1} \mathscr{x}(t)\mathscr{y}(t)\,dt$ 是一个有效的内积。

**证**

内积必须具有如下三个性质：

1) $(\mathscr{x},\mathscr{y}) = (\mathscr{y},\mathscr{x})$

$$(\mathscr{x},\mathscr{y}) = \int_{-1}^{1}\mathscr{x}(t)\mathscr{y}(t)\mathrm{d}t = \int_{-1}^{1}\mathscr{y}(t)\mathscr{x}(t)\mathrm{d}t = (\mathscr{y},\mathscr{x})$$

5-22 2) $(\mathscr{x}, a\mathscr{y}_1 + b\mathscr{y}_2) = a(\mathscr{x},\mathscr{y}_1) + b(\mathscr{x},\mathscr{y}_2)$：

$$\begin{aligned}(\mathscr{x}, a\mathscr{y}_1 + b\mathscr{y}_2) &= \int_{-1}^{1}\mathscr{x}(t)(a\mathscr{y}_1(t) + b\mathscr{y}_2(t))\mathrm{d}t\\ &= a\int_{-1}^{1}\mathscr{x}(t)\mathscr{y}_1(t)\mathrm{d}t + b\int_{-1}^{1}\mathscr{x}(t)\mathscr{y}_2(t)\mathrm{d}t\\ &= a(\mathscr{x},\mathscr{y}_1) + b(\mathscr{x},\mathscr{y}_2)\end{aligned}$$

3) $(\mathscr{x},\mathscr{x}) \geqslant 0$，其中等式成立，当且仅当 $\mathscr{x}$ 为零向量：

$$(\mathscr{x},\mathscr{x}) = \int_{-1}^{1}\mathscr{x}(t)\mathscr{x}(t)\mathrm{d}t = \int_{-1}^{1}\mathscr{x}^2(t)\mathrm{d}t \geqslant 0$$

当且仅当 $\mathscr{x}$ 为零向量(在 $-1 \leqslant t \leqslant 1$ 区间内，$\mathscr{x}(t) = 0$)时，上面等式成立。

**P5.7** 假设在前一例题中所定义的向量空间在区间[ - 1, 1]上定义的多项式集合有两个向量 $1 + t$ 和 $1 - t$。计算基于这两个向量的一个正交向量集合。

**解**

第 1 步：

$$\mathscr{v}_1 = \mathscr{y}_1 = 1 + t$$

第2步：

$$\mathcal{V}_2 = \mathcal{Y}_2 - \frac{(\mathcal{V}_1, \mathcal{Y}_2)}{(\mathcal{V}_1, \mathcal{V}_1)}\mathcal{V}_1$$

其中

$$(\mathcal{V}_1, \mathcal{Y}_2) = \int_{-1}^{1}(1+t)(1-t)\,dt = \left(t - \frac{t^3}{3}\right)\Big|_{-1}^{1} = \left(\frac{2}{3}\right) - \left(-\frac{2}{3}\right) = \frac{4}{3}$$

$$(\mathcal{V}_1, \mathcal{V}_1) = \int_{-1}^{1}(1+t)^2\,dt = \frac{(1+t)^3}{3}\Big|_{-1}^{1} = \left(\frac{8}{3}\right) - (0) = \frac{8}{3}$$

所以

$$\mathcal{V}_2 = (1-t) - \frac{4/3}{8/3}(1+t) = \frac{1}{2} - \frac{3}{2}t$$

5-23

**P5.8** 将 $\mathbf{x} = [6 \quad 9 \quad 9]^T$ 用如下基向量集展开：

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad \mathbf{v}_2 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \quad \mathbf{v}_3 = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}$$

**解**

第1步，计算互逆基向量：

$$\mathbf{B} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 2 \end{bmatrix}, \quad \mathbf{B}^{-1} = \begin{bmatrix} \frac{5}{3} & -\frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} \\ -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} \end{bmatrix}$$

取 $\mathbf{B}^{-1}$ 的行：

$$\mathbf{r}_1 = \begin{bmatrix} 5/3 \\ -1/3 \\ -1/3 \end{bmatrix}, \quad \mathbf{r}_2 = \begin{bmatrix} -1/3 \\ -1/3 \\ 2/3 \end{bmatrix}, \quad \mathbf{r}_3 = \begin{bmatrix} -1/3 \\ 2/3 \\ -1/3 \end{bmatrix}$$

计算展开式的系数：

$$x_1^v = \mathbf{r}_1^T\mathbf{x} = \begin{bmatrix} \frac{5}{3} & \frac{-1}{3} & \frac{-1}{3} \end{bmatrix}\begin{bmatrix} 6 \\ 9 \\ 9 \end{bmatrix} = 4$$

$$x_2^v = \mathbf{r}_2^T\mathbf{x} = \begin{bmatrix} \frac{-1}{3} & \frac{-1}{3} & \frac{2}{3} \end{bmatrix}\begin{bmatrix} 6 \\ 9 \\ 9 \end{bmatrix} = 1$$

$$x_3^v = \mathbf{r}_3^T\mathbf{x} = \begin{bmatrix} \frac{-1}{3} & \frac{2}{3} & \frac{-1}{3} \end{bmatrix}\begin{bmatrix} 6 \\ 9 \\ 9 \end{bmatrix} = 1$$

5-24

最后展开式写成

$$\mathbf{x} = x_1^v\mathbf{v}_1 + x_2^v\mathbf{v}_2 + x_3^v\mathbf{v}_3 = 4\begin{bmatrix}1\\1\\1\end{bmatrix} + 1\begin{bmatrix}1\\2\\3\end{bmatrix} + 1\begin{bmatrix}1\\3\\2\end{bmatrix}$$

也可用矩阵形式将上述过程表示为：

$$\mathbf{x}^v = \mathbf{B}^{-1}\mathbf{x} = \begin{bmatrix}\frac{5}{3} & -\frac{1}{3} & -\frac{1}{3}\\ -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3}\\ -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3}\end{bmatrix}\begin{bmatrix}6\\9\\9\end{bmatrix} = \begin{bmatrix}4\\1\\1\end{bmatrix}$$

请注意：$\mathbf{x}$ 和 $\mathbf{x}^v$ 表示的是同一个向量，但是它们分别是按照不同的基集展开的(如果不
5-25 特别说明，就假定 $\mathbf{x}$ 采用的是标准基集)。

## 5.5 结束语

本章给出了一些有关向量空间的基本概念和相关知识，它们是理解神经网络工作原理的关键。向量空间所涵盖的知识很多，我们并不试图涉及它的各个方面，而只是给出一些和神经网络密切相关的概念。这里讨论的问题几乎以后各章都要重新提到。

下一章将继续研究与神经网络密切相关的线性代数的主题，那里将主要研究线性变换和
5-26 矩阵。

### 参考文献

[Brog91] W. L. Brogan, *Modern Control Theory*, 3rd Ed., Englewood Cliffs, NJ: Prentice-Hall, 1991.

这是一部关于线性系统的好书。该书前半部分主要讨论线性代数知识。这本书有一些讨论求解线性微分方程组以及线性和非线性系统稳定性的很好章节。另外，书中还有许多例题。

[Stra76] G. Strang, *Linear Algebra and Its Applications*, New York: Academic Press, 1980.

这是 Strang 写的一本有关线性代数的优秀基础教材。本书中给出了许多线性代数的
5-27 应用实例。

### 习题

**E5.1** 再次考虑例题 P5.1。证明：若 $b \neq 0$，那么判定边界不是一个向量空间。

**E5.2** 在例题 P5.1 中，向量空间的维数是多少？

**E5.3** 考虑所有满足条件 $f(0)=0$ 的连续函数集合。证明：这些连续函数集合是一个向量空间。

**E5.4** 证明：所有的 $2\times 2$ 矩阵的集合是一个向量空间。

**E5.5** 在下列向量集合中，哪些是线性无关的？请求出每个向量集合所生成的向量空间的维数。(可用 MATLAB 中的函数 rank 对(i)和(iv)的答案进行验证。)

(i) $\begin{bmatrix}1\\2\\3\end{bmatrix}$　$\begin{bmatrix}1\\0\\1\end{bmatrix}$　$\begin{bmatrix}1\\2\\1\end{bmatrix}$

(ii) $\sin t$　$\cos t$　$\cos(2t)$

(iii) $1+t$　$1-t$

(iv) $\begin{bmatrix}1\\2\\2\\1\end{bmatrix}$　$\begin{bmatrix}1\\0\\0\\1\end{bmatrix}$　$\begin{bmatrix}3\\4\\4\\3\end{bmatrix}$

**E5.6** 请回顾一下第 3 章中的苹果和橘子的模式识别问题。试计算每个原型模式(苹果和橘子)向量和测试输入模式(椭圆形橘子)向量之间的夹角。验证向量夹角表示形式的直观意义。

$$\mathbf{P}_1=\begin{bmatrix}1\\-1\\-1\end{bmatrix}(\text{橘子}),\quad \mathbf{P}_2=\begin{bmatrix}1\\1\\-1\end{bmatrix}(\text{苹果}),\quad \mathbf{P}=\begin{bmatrix}-1\\-1\\-1\end{bmatrix}$$

5-28

**E5.7** 用 Gram-Schmidt 正交化方法，应用下列基向量求出一个正交向量集合。(请用 MATLAB 验证所得的答案。)

$$\mathbf{y}_1=\begin{bmatrix}1\\0\\0\end{bmatrix},\quad \mathbf{y}_2=\begin{bmatrix}1\\1\\0\end{bmatrix},\quad \mathbf{y}_3=\begin{bmatrix}1\\1\\1\end{bmatrix}$$

**E5.8** 考虑区间[0, 1]上的所有分段连续函数所构成的向量空间。图 5-12 定义的基集 $\{f_1, f_2, f_3\}$ 包含这个向量空间中的三个向量。

(i) 证明这个集合是线性无关的。

(ii) 试用 Gram-Schmidt 方法生成正交集合。内积的定义为

$$(f,g)=\int_0^1 f(t)g(t)\mathrm{d}t$$

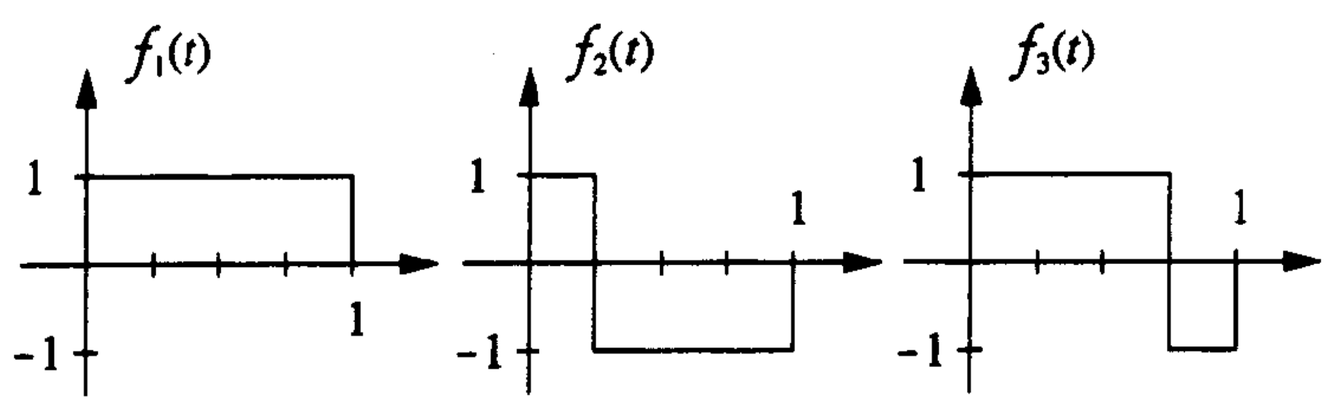



图 5-12　习题 E5.8 的基集

**E5.9** 试按下面的基集展开 $\mathbf{x}=[1\quad 2\quad 2]^T$。(请用 MATLAB 验证所得的答案。)

$$\mathbf{v}_1=\begin{bmatrix}-1\\1\\0\end{bmatrix},\quad \mathbf{v}_2=\begin{bmatrix}1\\1\\-2\end{bmatrix},\quad \mathbf{v}_3=\begin{bmatrix}1\\1\\0\end{bmatrix}$$

**E5.10** 试求使 $\|\mathcal{X}-a\mathcal{Y}\|$ 达到最小值的 $a$(采用 $\|\mathcal{X}\|=(\mathcal{X},\mathcal{X})^{1/2}$)。证明：对于 $a$ 的
5-29 这一取值，向量 $\mathcal{Z}=\mathcal{X}-a\mathcal{Y}$ 和向量 $\mathcal{Y}$ 正交，且 $\|\mathcal{X}-a\mathcal{Y}\|^2+\|a\mathcal{Y}\|^2=\|\mathcal{X}\|^2$(向量 $a\mathcal{Y}$ 是 $\mathcal{X}$ 在 $\mathcal{Y}$ 上的投影)。假设 $\mathcal{X}$ 和 $\mathcal{Y}$ 是二维向量，请画图解释这
5-30 一概念是如何与 Gram-Schmidt 正交化方法相关的。

# 第6章　神经网络中的线性变换

## 6.1　目的

本章将接着第 5 章继续论述神经网络分析所需要的数学基础。第 5 章复习了有关向量空间的内容，本章将探讨在神经网络中所采用的线性变换。

正如读者在前面几章中所看到的，输入向量和权值矩阵相乘是神经网络执行的一个关键操作。该操作是线性变换的一个具体实例。这一章希望研究一般的线性变换及其基本特点。本章将涉及诸如特征值、特征向量和基变换等基本概念，这些概念对读者理解一些诸如性能学习(包括 Widrow-Hoff 规则和反传学习算法)以及 Hopfield 网络的收敛特性等神经网络关键课题是十分重要的。 6-1

## 6.2　理论和实例

我们知道，第 3 章所讨论的 Hopfield 网络(如图 6-1)是根据下式同步对网络的输出进行修改的：

$$\mathbf{a}(t+1) = satlin(\mathbf{W}\mathbf{a}(t) + \mathbf{b}) \tag{6.1}$$

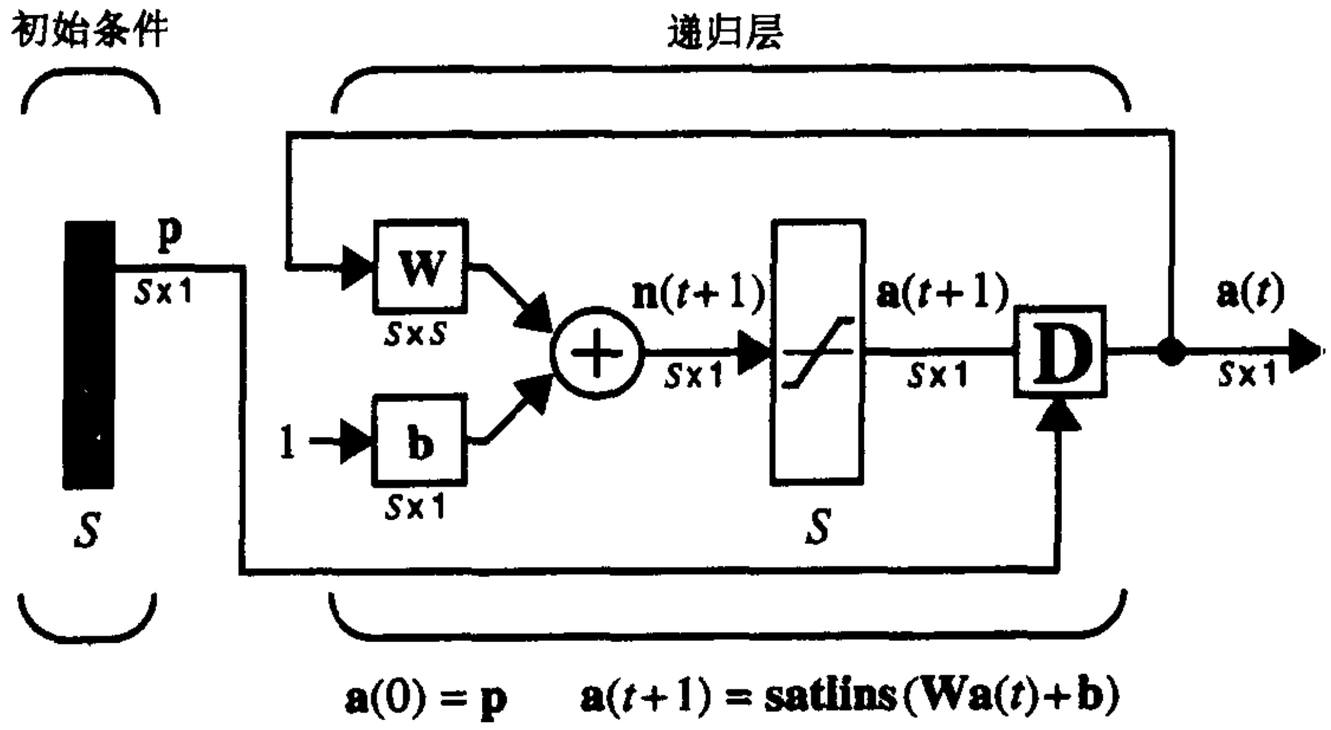



图 6-1　Hopfield 网络

请注意：在上式表示的迭代过程中，每次迭代操作均是网络的输出和权值矩阵 **W** 相乘。那么，这种重复操作的有什么作用呢？能否确定网络的输出是最终收敛到一个稳态值，还是趋于无穷，抑或是振荡不止呢？这一章将给出在本书所讨论的神经网络中解答诸如这类问题的数学基础。

### 6.2.1　线性变换

这里首先从一些基本定义开始论述。

**变换**　一个变换由如下三部分组成：

1）一个被称为定义域的元素集合 $X=\{x_i\}$；
2）一个被称为值域的元素集合 $Y=\{y_i\}$；
6-2 3）一个将每个 $x_i\in X$ 和一个元素 $y_i\in Y$ 相联系的规则。

**线性变换** 一个变换 $\mathscr{A}$ 是线性的，如果
1）对所有的 $x_1$，$x_2\in X$，$\mathscr{A}(x_1+x_2)=\mathscr{A}(x_1)+\mathscr{A}(x_2)$；
2）对所有的 $x\in X$ 和 $a\in R$，$\mathscr{A}(ax)=a\mathscr{A}(x)$。

假设某个变换 $\mathscr{A}$ 是在二维空间 $\mathfrak{R}^2$ 中将一个向量旋转 $\theta$ 角（如图 6-2 所示）。图 6-3 和图 6-4 表示该旋转变换满足线性变换定义中的条件 1，即如果希望将两个向量的和向量旋转一个角度，可以首先对这两个向量分别进行旋转，然后再对其求和。图 6-5 表示旋转变换满足线性变换定义中的条件 2，即如果希望将一个向量的伸缩向量进行旋转，可以首先旋转该向量，然后再对其伸缩。由此可以看出，旋转变换是一个线性变换。

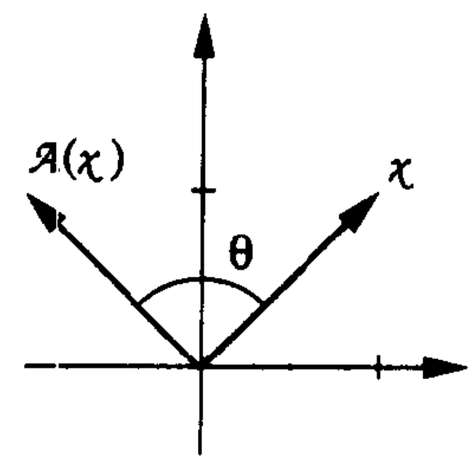


图 6-2 旋转变换

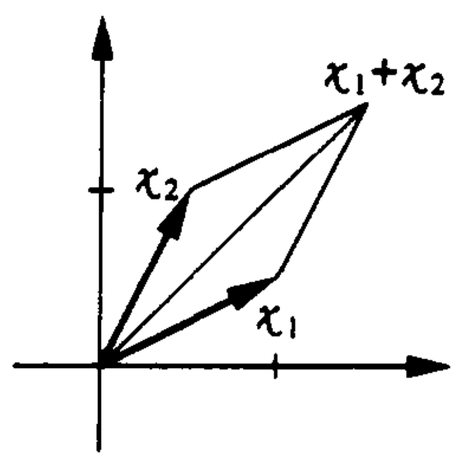


图 6-3 两个向量之和的旋转

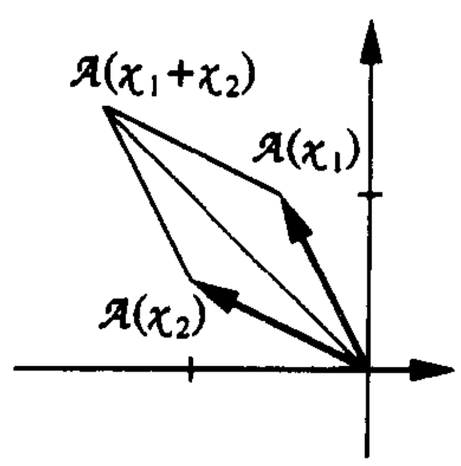


图 6-4 两个向量旋转后的和

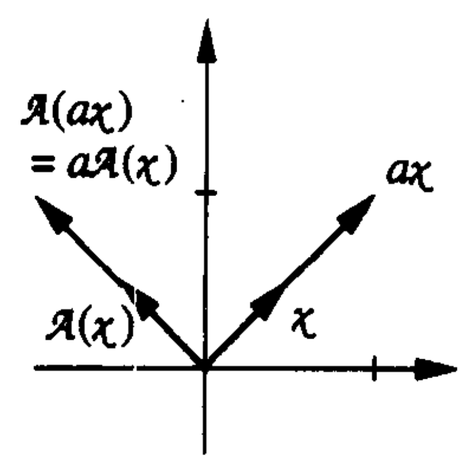


图 6-5 伸缩向量的变换

### 6.2.2 矩阵表示

正如在本章的开始所提到的，矩阵相乘是线性变换的一个实例。同样，可以证明两个有限维向量空间之间的任何线性变换都可以用一个矩阵来表示（这和上一章所阐述的在有限维的向量空间中的任何一个向量可以用一个数列来表示是一样的），为了说明这一点，本章将用到上一章所给出的多数概念。

设 $\{v_1, v_2, \cdots, v_n\}$ 是向量空间 $X$ 的一个基底，$\{u_1, u_2, \cdots, u_m\}$ 是向量空间 $Y$ 的一个基。即是对任意两个向量 $x\in X$ 和 $y\in Y$，有

$$x=\sum_{i=1}^{n} x_i v_i \text{ 和 } y=\sum_{i=1}^{m} y_i u_i \tag{6.2}$$

设 $\mathscr{A}$ 是一个定义域为 $X$ 值域为 $Y$ 的线性变换（$\mathscr{A}$：$X\to Y$）。那么

$$\mathscr{A}(x)=y \tag{6.3}$$

可以写成

$$\mathscr{A}\left(\sum_{j=1}^{n} x_j v_j\right)=\sum_{i=1}^{m} y_i u_i \tag{6.4}$$

因为 $\mathscr{A}$ 是一个线性算子，所以式(6.4)可写成

6-3

$$\sum_{j=1}^{n} x_j \mathscr{A}(v_j)=\sum_{i=1}^{m} y_i u_i \tag{6.5}$$

因为向量 $\mathcal{A}(\mathcal{V}_i)$是值域 $Y$ 中的一个元素，所以这些向量可以用 $Y$ 的基向量的线性组合形式写成

$$\mathcal{A}(\mathcal{V}_j) = \sum_{i=1}^{m} a_{ij}\mathcal{U}_i \tag{6.6}$$

(注意：上面展开式中的系数 $a_{ij}$并不是随意选取的。)如果将式(6.6)代入式(6.5)，可得

$$\sum_{j=1}^{n} x_j \sum_{i=1}^{m} a_{ij}\mathcal{U}_i = \sum_{i=1}^{m} y_i\mathcal{U}_i \tag{6.7}$$

交换式(6.7)中求和的顺序，有

$$\sum_{i=1}^{m} \mathcal{U}_i \sum_{j=1}^{n} a_{ij}x_j = \sum_{i=1}^{m} y_i\mathcal{U}_i \tag{6.8}$$

重新组织式(6.8)，可得

$$\sum_{i=1}^{m} \mathcal{U}_i\left(\sum_{j=1}^{n} a_{ij}x_j - y_i\right) = 0 \tag{6.9}$$

因为所有的 $\mathcal{U}_i$ 形成的是一个基集，所以它们必须是相互独立的。这也意味着式(6.9)中每个和 $\mathcal{U}_i$ 相乘的系数必须等于 0(参见式(5.4))，所以

$$\sum_{j=1}^{n} a_{ij}x_j = y_i \tag{6.10}$$

此式正好是下面形式的矩阵乘：

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{6.11}$$

上面这些结果表明：对于两个有限维向量空间之间的任意线性变换都存在与其相应的矩阵表示。当该矩阵和定义域向量 $\mathcal{X}$ 的展式相乘，可以得到一个变换向量 $\mathcal{Y}$ 的展式。

6-4

请记住：与一般向量的数列表示形式并不是惟一的类似(参见第 5 章)，一个变换的矩阵表示也不是惟一的。如果改变定义域或值域的基集，那么变换的矩阵表示也会随之改变。在后面各章将用到变换的这一矩阵表示特性。

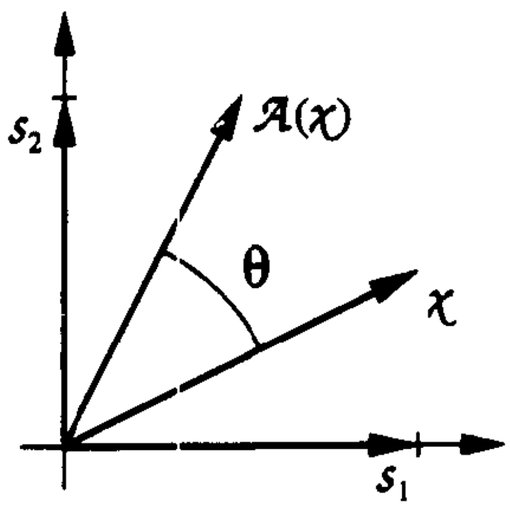



图 6-6

下面将以旋转变换为例，来讨论变换的矩阵表示，看看如何找到该变换的矩阵表示。实际上，其关键步骤已经在式(6.6)中给出。我们必须对定义域中的每个基向量进行变换，然后将其按照值域中的基向量形式展开。这里的定义域和值域相同($X = Y = \Re^2$)。为简单起见，对其采用标准基 $\mathcal{U}_i = \mathcal{V}_i = s_i$(如图 6-6 所示)。

第 1 步是对第一个基向量进行变换，并且以基向量的形式展开变换后的向量。如果将向量 $s_1$ 逆时针旋转一个角度 θ，可得

$$\mathcal{A}(s_1) = \cos(\theta)s_1 + \sin(\theta)s_2 = \sum_{i=1}^{2} a_{i1}s_i = a_{11}s_1 + a_{21}s_2 \tag{6.12}$$

如图 6-7 所示。可以看到展式中的两个系数就是矩阵表示中的第一列。

第 2 步是对第二个基向量进行变换。如果将向量 $s_2$ 逆时针旋转一个角度 θ，可得

$$\mathcal{A}(s_2) = -\sin(\theta)s_1 + \cos(\theta)s_2 = \sum_{i=1}^{2} a_{i2}s_i = a_{12}s_1 + a_{22}s_2 \tag{6.13}$$

如图 6-8 所示。从展式中可以得到矩阵表示中的第二列。所以，完整的矩阵表示可以由下式给出：

$$\mathbf{A} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{6.14}$$

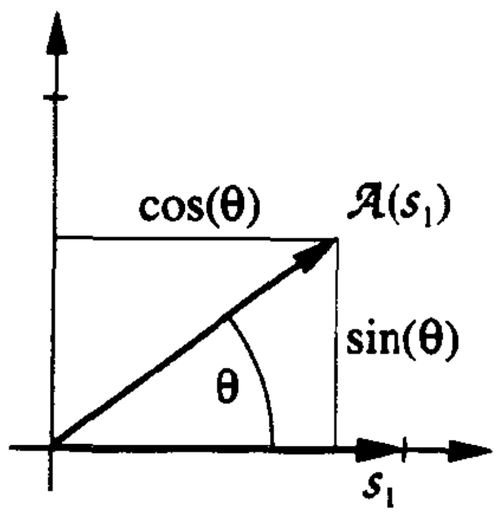



图 6-7

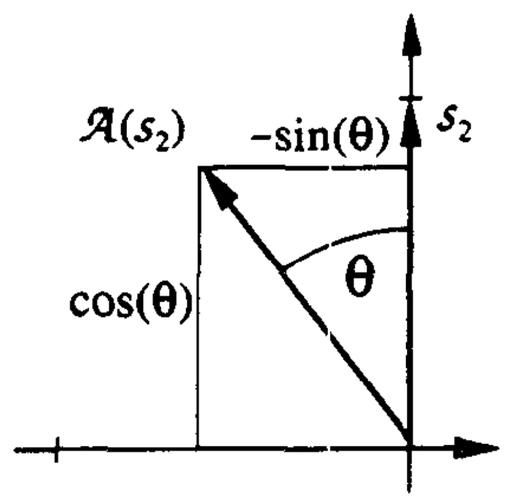



图 6-8

读者可以自行验证式(6.14)的矩阵表示形式。如果将一个向量和该矩阵相乘，那么向量将会被旋转一个角度 θ。

总起来说，可以利用式(6.6)得到一个变换的矩阵表示形式。首先需要对定义域中的每个基向量进行变换，然后以值域的基向量形式对变换后的向量进行展开，由每个展式的系数就可以得到变换矩阵中的一列。

6-5 *为了以图形方式研究上述生成矩阵表示的过程，请运行 Neural Network Design Demonstration Linear Transformations*(**nnd6lt**)。

### 6.2.3 基变换

前一节我们注意到一个线性变换的矩阵表示并不是惟一的。矩阵的表示依赖于变换的定义域和值域所采用的基集。在这一节，将说明变换的矩阵表示是如何随基集改变而改变的。

考虑一个线性变换：$\mathcal{A}$：$X \rightarrow Y$。设$\{\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_n\}$是向量空间 $X$ 的一个基，$\{\mathcal{U}_1, \mathcal{U}_2, \cdots, \mathcal{U}_m\}$是向量空间 $Y$ 的一个基。所以，任何向量 $\mathcal{X} \in X$ 均可以写成

$$\mathcal{X} = \sum_{i=1}^{n} x_i \mathcal{V}_i \tag{6.15}$$

而任何向量 $\mathcal{Y} \in Y$ 可以写成

$$\mathcal{Y} = \sum_{i=1}^{m} y_i \mathcal{U}_i \tag{6.16}$$

所以，如果

$$\mathcal{A}(\mathcal{X}) = \mathcal{Y} \tag{6.17}$$

那么，变换 $\mathcal{A}$ 的矩阵表示形式是

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \tag{6.18}$$

或

$$\mathbf{Ax} = \mathbf{y} \tag{6.19}$$

现在假设对 $X$ 和 $Y$ 使用不同的基集。设$\{t_1, t_2, \cdots, t_n\}$是 $X$ 的新基集，$\{w_1, w_2, \cdots, w_m\}$是 $Y$ 的新基集。那么，向量 $x \in X$ 可以写成

$$x = \sum_{i=1}^{n} x'_i t_i \tag{6.20}$$

6-6

向量 $y \in Y$ 可以写成

$$y = \sum_{i=1}^{m} y'_i w_i \tag{6.21}$$

这将得到如下新的矩阵表示：

$$\begin{bmatrix} a'_{11} & a'_{12} & \cdots & a'_{1n} \\ a'_{21} & a'_{22} & \cdots & a'_{2n} \\ \vdots & \vdots & & \vdots \\ a'_{m1} & a'_{m2} & \cdots & a'_{mn} \end{bmatrix} \begin{bmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_n \end{bmatrix} = \begin{bmatrix} y'_1 \\ y'_2 \\ \vdots \\ y'_m \end{bmatrix} \tag{6.22}$$

或

$$\mathbf{A}'(\mathbf{x}') = \mathbf{y}' \tag{6.23}$$

那么，$\mathbf{A}$ 和 $\mathbf{A}'$之间的关系是什么呢？要解答这个问题，必须找出两个基集之间的关系。首先，由于每个 $t_i$ 是 $X$ 的一个元素，那么可以按照 $X$ 原先基集的形式展开：

$$t_i = \sum_{j=1}^{n} t_{ji} v_j \tag{6.24}$$

其次，因为每个 $w_i$ 是 $Y$ 的一个元素，所以也可以按照 $Y$ 原先基集的形式展开：

$$w_i = \sum_{j=1}^{m} w_{ji} u_j \tag{6.25}$$

所以，基向量可以写为如下的列向量表示形式：

$$\mathbf{t}_i = \begin{bmatrix} t_{1i} \\ t_{2i} \\ \vdots \\ t_{ni} \end{bmatrix} \qquad \mathbf{w}_i = \begin{bmatrix} w_{1i} \\ w_{2i} \\ \vdots \\ w_{mi} \end{bmatrix} \tag{6.26}$$

定义一个列为 $\mathbf{t}_i$ 的矩阵：

$$\mathbf{B}_t = [\mathbf{t}_1 \quad \mathbf{t}_2 \quad \cdots \quad \mathbf{t}_n] \tag{6.27}$$

现在可以按照矩阵的形式将式(6.20)写为：

$$\mathbf{x} = x'_1\mathbf{t}_1 + x'_2\mathbf{t}_2 + \cdots + x'_n\mathbf{t}_n = \mathbf{B}_t\mathbf{x}' \tag{6.28}$$

6-7

这个等式说明了向量 $x$ 的两种不同表示之间的关系。

现在，定义一个列为 $\mathbf{w}_i$ 的矩阵：

$$\mathbf{B}_w = [\mathbf{w}_1 \quad \mathbf{w}_2 \quad \cdots \quad \mathbf{w}_m] \tag{6.29}$$

据此，也可以按照矩阵的形式将式(6.21)写成

$$\mathbf{y} = \mathbf{B}_w \mathbf{y}' \tag{6.30}$$

这个等式说明了向量 $y$ 的两种不同表示之间的关系。

现在将式(6.28)和式(6.30)代入式(6.19)，可得

$$\mathbf{A}\mathbf{B}_t\mathbf{X}' = \mathbf{B}_w\mathbf{y}' \tag{6.31}$$

如果我们用 $\mathbf{B}_w^{-1}$ 乘以式(6.31)的两边，有

$$[\mathbf{B}_w^{-1}\mathbf{A}\mathbf{B}_t]\mathbf{x}' = \mathbf{y}' \tag{6.32}$$

**基变换** 比较式(6.32)和式(6.23)可以得到如下基变换的操作：

$$\mathbf{A}' = [\mathbf{B}_w^{-1}\mathbf{A}\mathbf{B}_t] \tag{6.33}$$

**相似变换** 这个重要结果描述了一个给定线性变换的任何两个矩阵表示之间的关系，该变换称为相似变换(similarity transform)[Brog91]。此式在以下各章中十分有用。如果选择比较合适的基向量，那么就可以获得一个充分反映线性变换特点的矩阵表示。这个问题将在下一节讨论。

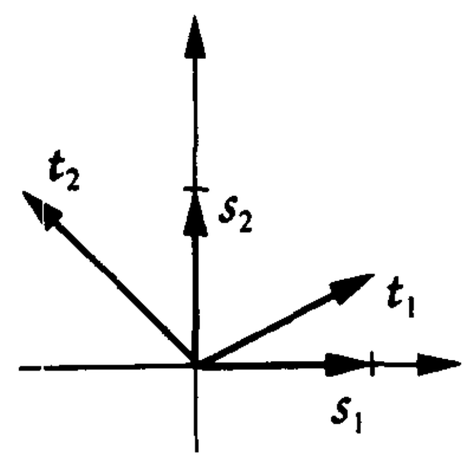


图 6-9 基变换实例

作为一个基集变换的实例，让我们重新看看上节所给出的向量旋转实例。在该实例中，利用标准的基集 $\{s_1, s_2\}$ 得到了一个矩阵表示。现在利用基 $\{t_1, t_2\}$ 找到一个新的矩阵表示(如图 6-9 所示)。注意：在该实例中，定义域和值域采用的是同一个基集。

第一步是根据式(6.24)和式(6.25)，按照标准基集的形式对 $t_1$ 和 $t_2$ 进行展开。观察图 6-9 可知：

6-8

$$t_1 = s_1 + 0.5s_2 \tag{6.34}$$

$$t_2 = -s_1 + s_2 \tag{6.35}$$

所以，可以将 $t_1$ 和 $t_2$ 写成

$$\mathbf{t}_1 = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} \qquad \mathbf{t}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \tag{6.36}$$

现在，可以得到矩阵

$$\mathbf{B}_t = [\mathbf{t}_1, \mathbf{t}_2] = \begin{bmatrix} 1 & -1 \\ 0.5 & 1 \end{bmatrix} \tag{6.37}$$

同时，由于这里对变换的定义域和值域都是采用相同的基集，所以

$$\mathbf{B}_w = \mathbf{B}_t = \begin{bmatrix} 1 & -1 \\ 0.5 & 1 \end{bmatrix} \tag{6.38}$$

现在，可以根据式(6.33)计算新的矩阵表示：

$$\begin{aligned} \mathbf{A}' = [\mathbf{B}_w^{-1}\mathbf{A}\mathbf{B}_t] &= \begin{bmatrix} 2/3 & 2/3 \\ -1/3 & 2/3 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 0.5 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1/3\sin\theta + \cos\theta & -4/3\sin\theta \\ 5/6\sin\theta & -1/3\sin\theta + \cos\theta \end{bmatrix} \end{aligned} \tag{6.39}$$

作为特例，不妨选取 $\theta = 30°$，于是有

$$\mathbf{A}' = \begin{bmatrix} 1.033 & -0.667 \\ 0.417 & 0.699 \end{bmatrix} \tag{6.40}$$

和

$$\mathbf{A} = \begin{bmatrix} 0.866 & -0.5 \\ 0.5 & 0.866 \end{bmatrix} \tag{6.41}$$

为了检验这些矩阵是否正确，假设和 $\mathbf{x}' = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ 相对应的测试向量是：

$$\mathbf{x} = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} \tag{6.42}$$

6-9

(注意：$\mathbf{x}$ 和 $\mathbf{x}'$ 表示的向量是第二个基集中的一个元素 $t_1$。)那么，变换后的测试向量是

$$\mathbf{y} = \mathbf{A}\mathbf{x} = \begin{bmatrix} 0.866 & -0.5 \\ 0.5 & 0.866 \end{bmatrix} \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.616 \\ 0.933 \end{bmatrix} \tag{6.43}$$

该向量应该和如下向量相对应：

$$\mathbf{y}' = \mathbf{A}'\mathbf{x}' = \begin{bmatrix} 1.033 & -0.667 \\ 0.416 & 0.699 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1.033 \\ 0.416 \end{bmatrix} \tag{6.44}$$

那么，如何确定 $\mathbf{y}$ 和 $\mathbf{y}'$ 的确是相对应呢？它们是以不同基集的形式来表示同一个向量 $\mathcal{Y}$，$\mathbf{y}$ 采用的基是 $\{s_1, s_2\}$，$\mathbf{y}'$ 采用的基是 $\{t_1, t_2\}$。在第 5 章中，利用互逆的基向量将一个变换转换成另一个变换(请见式(5.43))。利用此概念，可得

$$\begin{aligned} \mathbf{y}' = \mathbf{B}^{-1}\mathbf{y} &= \begin{bmatrix} 1 & -1 \\ 0.5 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0.616 \\ 0.933 \end{bmatrix} \\ &= \begin{bmatrix} 2/3 & 2/3 \\ -1/3 & 2/3 \end{bmatrix} \begin{bmatrix} 0.616 \\ 0.933 \end{bmatrix} = \begin{bmatrix} 1.033 \\ 0.416 \end{bmatrix} \end{aligned} \tag{6.45}$$

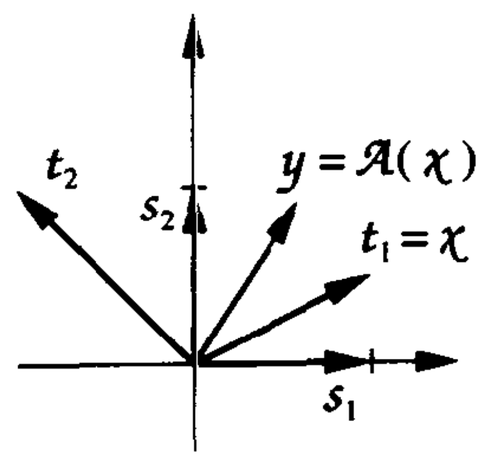


图 6-10

此式正好验证了前面的结果。这些向量表示在图 6-10 中。从图中可以看出，由式(6.43)和式(6.44)得到 $\mathbf{y}$ 和 $\mathbf{y}'$ 两种表示形式是合理的。

### 6.2.4　特征值和特征向量

本节将对线性变换的特征值和特征向量这两个关键性质进行讨论。这些性质的知识将使我们能回答有关神经网络性能的一些关键问题，比如在本章开始所提到的 Hopfield 网络的稳定性。

**特征值　特征向量**　这里首先给出特征值和特征向量的定义。考虑一个线性变换：$\mathcal{A}$：$X \to X$(定义域和值域相同)。分别称满足下式的那些不等于 0 的向量 $\mathcal{Z} \in X$ 和标量 $\lambda$ 分别是特征向量和特征值：

$$\mathcal{A}(\mathcal{Z}) = \lambda \mathcal{Z} \tag{6.46}$$

请注意，特征向量的表示问题，因为如果 $\mathcal{Z}$ 满足式(6.46)，那么 $a\mathcal{Z}$ 同样也满足该式，由此可知，特征向量实际上并不是一个真正的向量，而是一个向量空间。

所以，给定变换的一个特征向量表示一个方向，当对任何取该方向的向量进行变换时，它们都将继续指向相同的方向，仅仅是
6-10 按照特征值对向量的长度进行缩放。举例来说，再次考虑前几节中提到的旋转实例(如图 6-11 所示)。现在要问：是不是任何向量被旋转 30°之后，它们还是指向相同的方向？显然不是，这是因为变换没有实数特征值的情况。在后面将会看到，如果允许复数形式的特征值，那么该变换存在两个特征值。

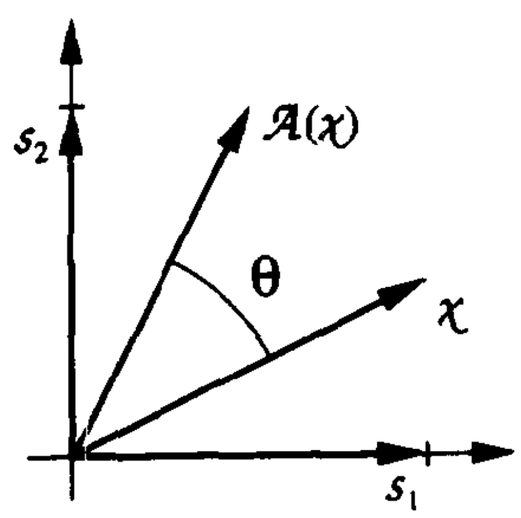


图 6-11 向量旋转实例

那么，又如何计算特征值和特征向量呢？假设现在选择了 $n$ 维向量空间 $X$ 的一个基，那么式(6.46)的矩阵表示可以写成

$$\mathbf{Az} = \lambda\mathbf{z} \tag{6.47}$$

或

$$[\mathbf{A} - \lambda\mathbf{I}]\mathbf{z} = 0 \tag{6.48}$$

这表示$[\mathbf{A} - \lambda\mathbf{I}]$的列之间是线性相关的，由此可以知该矩阵的行列式必为 0：

$$|[\mathbf{A} - \lambda\mathbf{I}]| = 0 \tag{6.49}$$

这个行列式是一个 $n$ 阶多项式，所以式(6.49)通常有 $n$ 个根，其中一些根可能是复数，也可能有些根是重复的。

现在，重新看看前面的旋转实例。如果采用标准基集，那么变换的矩阵是

$$\mathbf{A} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{6.50}$$

可以将式(6.49)写成

$$\left|\begin{bmatrix} \cos\theta - \lambda & -\sin\theta \\ \sin\theta & \cos\theta - \lambda \end{bmatrix}\right| = 0 \tag{6.51}$$

或

$$\lambda^2 - 2\lambda\cos\theta + ((\cos\theta)^2 + (\sin\theta)^2) = \lambda^2 - 2\lambda\cos\theta + 1 = 0 \tag{6.52}$$

该等式的根是

$$\lambda_1 = \cos\theta + j\sin\theta, \lambda_2 = \cos\theta - j\sin\theta \tag{6.53}$$

所以，正如前面所预计的，该变换没有实数形式的特征值(如果 $\sin\theta \neq 0$)。这也说明，
6-11 如果任何实向量被变换之后，它将指向一个新的方向。

考虑另外一个矩阵：

$$\mathbf{A} = \begin{bmatrix} -1 & 1 \\ 0 & -2 \end{bmatrix} \tag{6.54}$$

为了找到其特征值，必须求解

$$\left|\begin{bmatrix} -1-\lambda & 1 \\ 0 & -2-\lambda \end{bmatrix}\right| = 0 \tag{6.55}$$

或

$$\lambda^2 + 3\lambda + 2 = (\lambda + 1)(\lambda + 2) = 0 \tag{6.56}$$

求解式(6.56)，得到特征值

$$\lambda_1 = -1, \quad \lambda_2 = -2 \tag{6.57}$$

为了找到其特征向量，必须对式(6.48)求解，这里就是求解

$$\begin{bmatrix} -1-\lambda & 1 \\ 0 & -2-\lambda \end{bmatrix}\mathbf{z} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{6.58}$$

分别用 $\lambda_1$ 和 $\lambda_2$ 对该式进行两次求解。首先将 $\lambda_1$ 代入式(6.58)可得

$$\begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix}\mathbf{z}_1 = \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} z_{11} \\ z_{21} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{6.59}$$

或

$$z_{21} = 0, \text{对 } z_{11} \text{ 没有任何限制} \tag{6.60}$$

所以第一个特征向量是

$$\mathbf{z}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{6.61}$$

或者是该向量的任意标量倍。将 $\lambda_2$ 代入式(6.58)，可得 6-12

$$\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}\mathbf{z}_1 = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} z_{12} \\ z_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{6.62}$$

或

$$z_{22} = -z_{12} \tag{6.63}$$

所以第二个特征向量是

$$\mathbf{z}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \tag{6.64}$$

或者是该向量任意的标量倍数。

下面两式验证了上述结果的正确性：

$$\mathbf{A}\mathbf{z}_1 = \begin{bmatrix} -1 & 1 \\ 0 & -2 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \end{bmatrix} = (-1)\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \lambda_1\mathbf{z}_1 \tag{6.65}$$

$$\mathbf{A}\mathbf{z}_2 = \begin{bmatrix} -1 & 1 \\ 0 & -2 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \end{bmatrix} = (-2)\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \lambda_2\mathbf{z}_2 \tag{6.66}$$

为了测验一下对特征向量的理解，可以运行 *Neural Network Design Demonstration Eigenvector Game* (**nnd6eg**)。

**对角化**

如果某个变换有 $n$ 个不同的特征值，则可以保证得到该变换 $n$ 个线性无关的特征向量[Brog91]。因此特征向量组成变换的向量空间的一个基集。现在用特征向量作为基向量来求出前面变换(式(6.54))的矩阵。从式(6.33)可得

$$\mathbf{A}' = [\mathbf{B}^{-1}\mathbf{A}\mathbf{B}] = \begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} -1 & 1 \\ 0 & -2 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -2 \end{bmatrix} \tag{6.67}$$

**对角化** 注意，这是一个对角矩阵，特征值处于对角线上。实际上，这并不是一个巧

合，一旦变换有不同的特征值，那么就能通过将特征向量作为基向量的方法将该变换的矩阵表示对角化。可以将这种对角化过程总结如下：

设

6-13

$$\mathbf{B} = [\mathbf{z}_1 \quad \mathbf{z}_2 \quad \cdots \quad \mathbf{z}_n] \tag{6.68}$$

其中$\{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n\}$是一个矩阵 $\mathbf{A}$ 的特征向量。然后求

$$[\mathbf{B}^{-1}\mathbf{A}\mathbf{B}] = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} \tag{6.69}$$

其中$\{\lambda_1, \lambda_2, \cdots, \lambda_n\}$是矩阵 $\mathbf{A}$ 的特征值。

6-14

这个结果对后面各章分析若干神经网络的性能是十分有用的。

## 6.3 小结

### 变换

一个变换由如下三部分组成：

1）一个被称为定义域的元素集合 $X = \{\mathcal{X}_i\}$；

2）一个被称为值域的元素集合 $Y = \{\mathcal{Y}_i\}$；

3）一个将每个元素 $\mathcal{X}_i \in X$ 映射到元素 $\mathcal{Y}_i \in Y$ 的规则。

### 线性变换

一个变换 $\mathcal{A}$ 是线性的，如果：

1）对所有的 $\mathcal{X}_1$ 和 $\mathcal{X}_2 \in X$，$\mathcal{A}(\mathcal{X}_1 + \mathcal{X}_2) = \mathcal{A}(\mathcal{X}_1) + \mathcal{A}(\mathcal{X}_2)$；

2）对所有的 $\mathcal{X} \in X$ 和 $a \in R$，$\mathcal{A}(a\mathcal{X}) = a\mathcal{A}(\mathcal{X})$。

### 矩阵表示

设$\{\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_n\}$是向量空间 $X$ 的一个基，$\{\mathcal{U}_1, \mathcal{U}_2, \cdots, \mathcal{U}_m\}$是向量空间 $Y$ 的一个基。$\mathcal{A}$ 是一个定义域为 $X$ 和值域为 $Y$ 的线性变换：

$$\mathcal{A}(\mathcal{X}) = \mathcal{Y}$$

那么变换的矩阵表示中的系数可以由下式获得：

$$\mathcal{A}(\mathcal{V}_j) = \sum_{i=1}^{m} a_{ij}\mathcal{U}_i$$

### 基变换

$$\mathbf{B}_t = [\mathbf{t}_1 \quad \mathbf{t}_2 \quad \cdots \quad \mathbf{t}_n]$$

$$\mathbf{B}_w = [\mathbf{w}_1 \quad \mathbf{w}_2 \quad \cdots \quad \mathbf{w}_m]$$

6-15

$$\mathbf{A}' = [\mathbf{B}_w^{-1}\mathbf{A}\mathbf{B}_t]$$

**特征值和特征向量**

$$\mathbf{Az} = \lambda\mathbf{z}$$
$$|[\mathbf{A} - \lambda\mathbf{I}]| = 0$$

**对角化**

$$\mathbf{B} = [\mathbf{z}_1\ \mathbf{z}_2 \cdots \mathbf{z}_n]$$

其中$\{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n\}$是一个方阵 **A** 的特征向量。

$$[\mathbf{B}^{-1}\mathbf{AB}] = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}$$

6-16

## 6.4 例题

**P6.1** 考虑图 6-12 中具有线性传输函数的单层网络，请问从输入向量到输出向量之间的变换是线性变换吗？

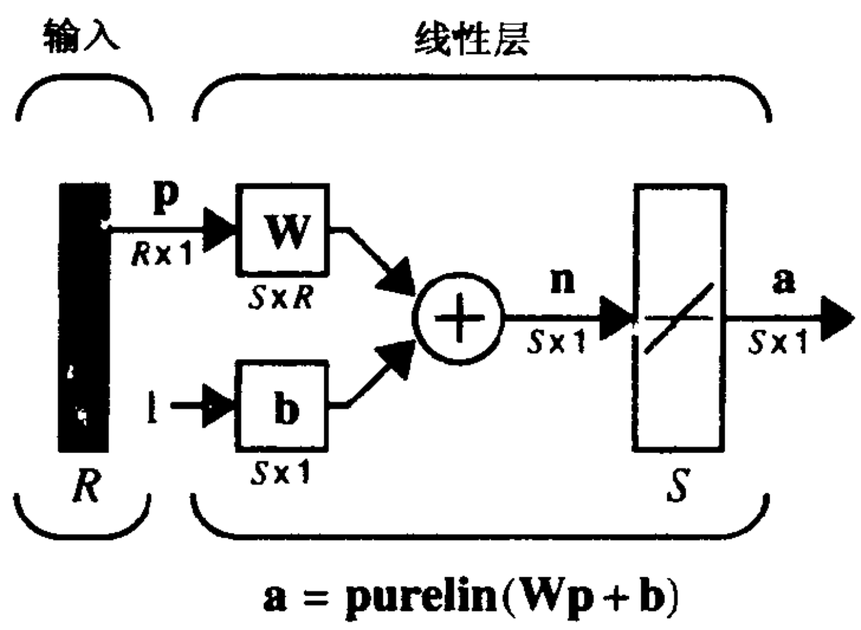



图 6-12　单个神经元感知机

**解**

网络的映射公式是

$$\mathbf{a} = \mathscr{A}(\mathbf{p}) = \mathbf{Wp} + \mathbf{b}$$

为了使这个变换是线性的，它必须满足：

1）$\mathscr{A}(\mathbf{p}_1 + \mathbf{p}_2) = \mathscr{A}(\mathbf{p}_1) + \mathscr{A}(\mathbf{p}_2)$；

2）$\mathscr{A}(a\mathbf{p}) = a\mathscr{A}(\mathbf{p})$。

首先测试上面的第一个条件。

$$\mathscr{A}(\mathbf{p}_1 + \mathbf{p}_2) = \mathbf{W}(\mathbf{p}_1 + \mathbf{p}_2) + \mathbf{b} = \mathbf{Wp}_1 + \mathbf{Wp}_2 + \mathbf{b}$$

将其和

$$\mathscr{A}(\mathbf{p}_1) + \mathscr{A}(\mathbf{p}_2) = \mathbf{Wp}_1 + \mathbf{b} + \mathbf{Wp}_2 + \mathbf{b} = \mathbf{Wp}_1 + \mathbf{Wp}_2 + 2\mathbf{b}$$

比较。显然，仅当 $\mathbf{b} = \mathbf{0}$时，上述两个表达式相等。所以，尽管该网络具有一个线性传输函数，但是它执行的是一个非线性变换。我们称这种特殊类型的非线性变换为仿射变换。 6-17

**P6.2** 在第 5 章讨论过投影，投影是一个线性变换吗?

**解**

向量 $\mathcal{X}$ 到向量 $\mathcal{V}$ 上的投影定义为

$$\mathcal{Y} = \mathcal{A}(\mathcal{X}) = \frac{(\mathcal{X},\mathcal{V})}{(\mathcal{V},\mathcal{V})}\mathcal{V}$$

其中，$(\mathcal{X},\mathcal{V})$是 $\mathcal{X}$ 和 $\mathcal{V}$ 的内积。

现在需要检查一下这个变换是否满足线性特性的两个条件。首先检查条件 1)：

$$\mathcal{A}(\mathcal{X}_1 + \mathcal{X}_2) = \frac{(\mathcal{X}_1 + \mathcal{X}_2,\mathcal{V})}{(\mathcal{V},\mathcal{V})}\mathcal{V} = \frac{(\mathcal{X}_1,\mathcal{V}) + (\mathcal{X}_2,\mathcal{V})}{(\mathcal{V},\mathcal{V})}\mathcal{V} = \frac{(\mathcal{X}_1,\mathcal{V})}{(\mathcal{V},\mathcal{V})}\mathcal{V} + \frac{(\mathcal{X}_2,\mathcal{V})}{(\mathcal{V},\mathcal{V})}\mathcal{V}$$

$$= \mathcal{A}(\mathcal{X}_1) + \mathcal{A}(\mathcal{X}_2)$$

(这里使用了内积的线性特性。)现在检查条件 2)：

$$\mathcal{Y} = \mathcal{A}(a\mathcal{X}) = \frac{(a\mathcal{X},\mathcal{V})}{(\mathcal{V},\mathcal{V})}\mathcal{V} = \frac{a(\mathcal{X},\mathcal{V})}{(\mathcal{V},\mathcal{V})}\mathcal{V} = a\mathcal{A}(\mathcal{X})$$

由此可以看出，投影是一个线性操作。

**P6.3** 考虑$\mathfrak{R}^2$ 中将向量 $\mathcal{X}$ 相对于直线 $x_1 + x_2 = 0$ 进行反射操作的变换 $\mathcal{A}$(如图 6-13 所示)。请求出和$\mathfrak{R}^2$ 中标准基集相关的该变换的矩阵。

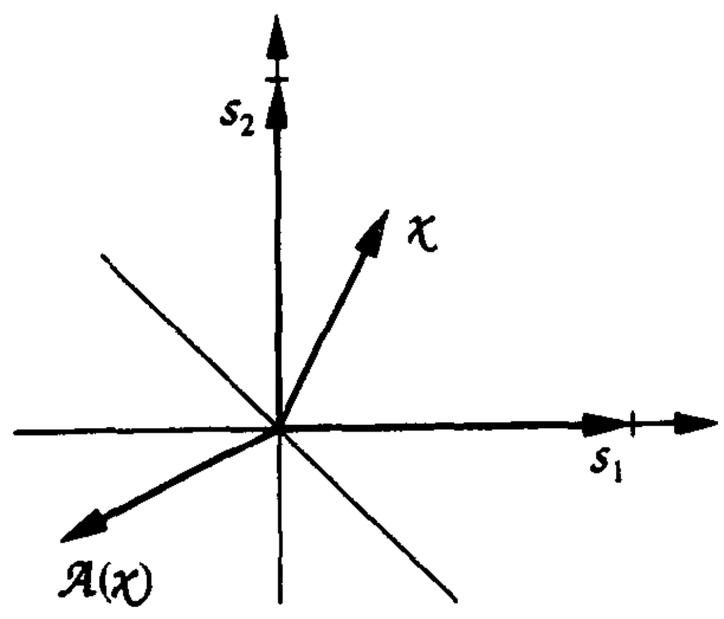



图 6-13 反射变换

**解**

6-18 求一个变换的矩阵的关键已经在式(6.6)中给出：

$$\mathcal{A}(\mathcal{V}_j) = \sum_{i=1}^{m} a_{ij}\mathcal{U}_i$$

这里需要对定义域中的每个基向量进行变换，然后按照值域的基向量形式对每个变换结果进行展开，每次展开得到矩阵表示中的一列。这里，定义域和值域的基集都是$\{s_1, s_2\}$。所以首先对 $s_1$ 进行变换。如果依据直线 $x_1 + x_2 = 0$ 反射 $s_1$(如图 6-14(a)所示)，可得

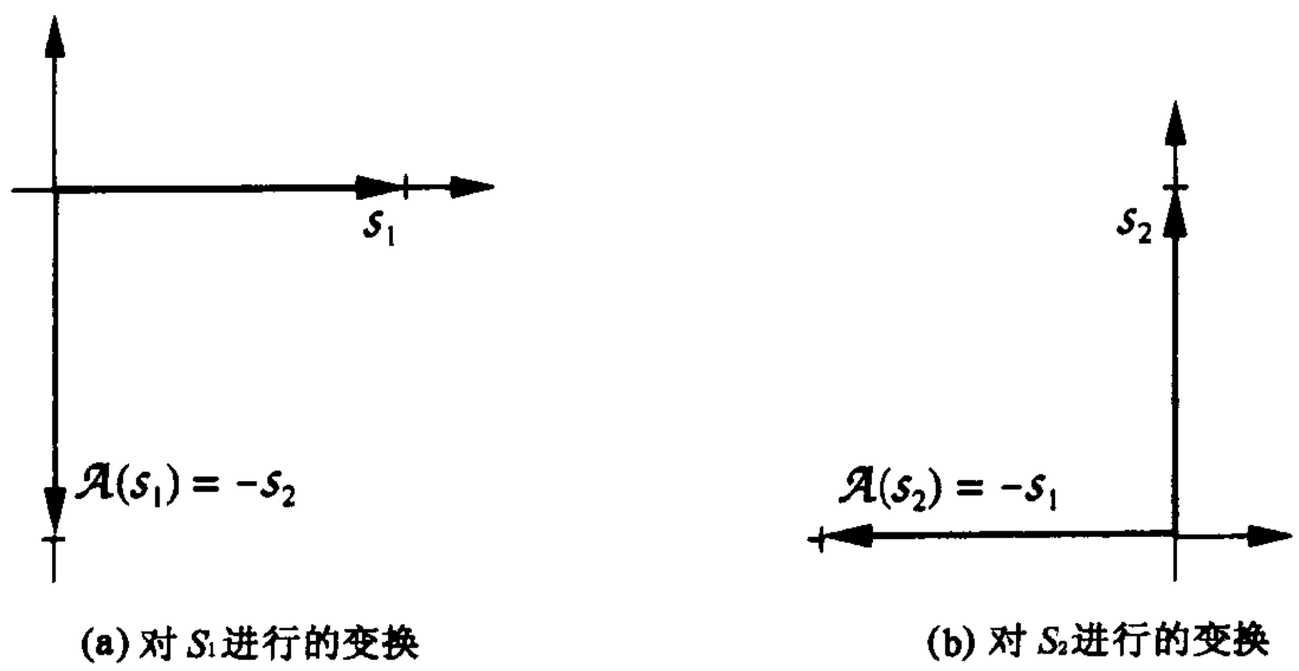
$$\mathcal{A}(s_1) = -s_2 = \sum_{i=1}^{2} a_{i1}s_i = a_{11}s_1 + a_{21}s_2 = 0s_1 + (-1)s_2$$

此式给出矩阵的第一列。下面对 $s_2$ 进行变换(如图 6-14(b)所示)，可得

$$\mathcal{A}(s_2) = -s_1 = \sum_{i=1}^{2} a_{i2}s_i = a_{12}s_1 + a_{22}s_2 = (-1)s_1 + 0s_2$$

此式给出矩阵的第二列。最后的结果是

$$\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$




(a) 对 $S_1$ 进行的变换　　(b) 对 $S_2$ 进行的变换

图 6-14　反射操作

下面通过对向量 $\mathbf{x} = [1 \quad 1]^T$ 进行变换来验证上述结果：

$$\mathbf{Ax} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

这实际上是依据直线 $x_1 + x_2 = 0$ 对向量 $\mathbf{x}$ 进行反射(如图 6-15 所示)。

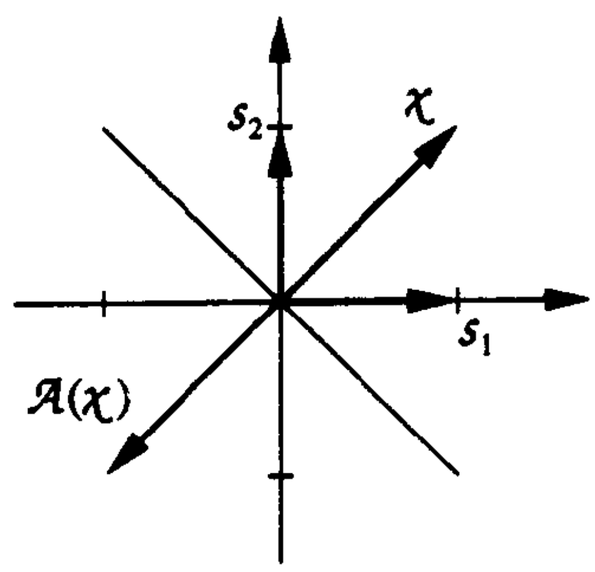



图 6-15　对反射操作的验证

6-19

(你能够猜测出该变换的特征值和特征向量吗？请使用 *Neural Network Design Demonstration* (神经网络设计演示)中的 *Linear Transfomations*(**nn6lt**)以图形的方式研究一下。请利用 MATLAB 的 **eig** 函数计算该特征值和特征向量，然后检验一下你的猜测结果是否正确。)

**P6.4** 设复数向量空间 $X$ 的基是$\{1+j,\ 1-j\}$，变换 $\mathcal{A}$：$X \to X$ 是一个共轭算子(即 $\mathcal{A}(\mathcal{X}) = \mathcal{X}^*$)。

(i) 求变换 $\mathcal{A}$ 相对于上述基集的矩阵表示；

(ii) 求该变换的特征值和特征向量；

(iii) 当将特征向量作为基向量时，求 $\mathcal{A}$ 相对于该基向量的矩阵表示。

**解**

(i) 为了求该变换的矩阵，对每个基向量进行变换，也即求每个基向量的共轭：

$$\mathcal{A}(\mathcal{V}_1) = \mathcal{A}(1+j) = 1-j = \mathcal{V}_2 = a_{11}\mathcal{V}_1 + a_{21}\mathcal{V}_2 = 0\mathcal{V}_1 + 1\mathcal{V}_2$$

$$\mathcal{A}(\mathcal{V}_2) = \mathcal{A}(1-j) = 1+j = \mathcal{V}_1 = a_{12}\mathcal{V}_1 + a_{22}\mathcal{V}_2 = 1\mathcal{V}_1 + 0\mathcal{V}_2$$

从上面两式可得变换的矩阵表示

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

（ii）为了求特征值，需要用式(6.49)：

$$|[\mathbf{A} - \lambda\mathbf{I}]| = \left|\begin{bmatrix} -\lambda & 1 \\ 1 & -\lambda \end{bmatrix}\right| = \lambda^2 - 1 = (\lambda - 1)(\lambda + 1) = 0$$

所以特征值是 $\lambda_1 = 1$，$\lambda_2 = -1$。为了求特征向量，用式(6.48)：

$$[\mathbf{A} - \lambda\mathbf{I}]\mathbf{z} = \begin{bmatrix} -\lambda & 1 \\ 1 & -\lambda \end{bmatrix}\mathbf{z} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

6-20 当 $\lambda = \lambda_1 = 1$ 时，有

$$\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}\mathbf{z}_1 = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} \mathscr{Z}_{11} \\ \mathscr{Z}_{21} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

或

$$\mathscr{Z}_{11} = \mathscr{Z}_{21}$$

所以第一个特征向量是

$$\mathbf{z}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

或是该向量的任意倍数。对第二个特征向量而言，用 $\lambda = \lambda_2 = -1$，可得：

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}\mathbf{z}_2 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} \mathscr{Z}_{12} \\ \mathscr{Z}_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

或

$$\mathscr{Z}_{12} = -\mathscr{Z}_{22}$$

所以第二个特征向量是

$$\mathbf{z}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

或是该向量的任意倍数。

注意：虽然前面这些特征向量是用一列数来表示，但实际上它们都是复数，比如：

$$\mathscr{Z}_1 = 1\mathscr{V}_1 + 1\mathscr{V}_2 = (1 + j) + (1 - j) = 2$$

$$\mathscr{Z}_2 = 1\mathscr{V}_1 + (-1)\mathscr{V}_2 = (1 + j) - (1 - j) = 2j$$

检查这两个数，它们确实是特征向量：

$$\mathscr{A}(\mathscr{Z}_1) = (2)^* = 2 = \lambda_1\mathscr{Z}_1$$

$$\mathscr{A}(\mathscr{Z}_2) = (2j)^* = -2j = \lambda_2\mathscr{Z}_2$$

（iii）为对基集进行变换，需要用式(6.33)：

6-21

$$\mathbf{A}' = [\mathbf{B}_w^{-1}\mathbf{A}\mathbf{B}_t] = [\mathbf{B}^{-1}\mathbf{A}\mathbf{B}]$$

其中

$$\mathbf{B} = [\mathbf{z}_1 \quad \mathbf{z}_2] = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

(定义域和值域用的是同一个基集。)所以有

$$\mathbf{A}' = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

正如式(6.69)所预计的，这里将变换的矩阵表示对角化了。

**P6.5** 请对角化下面的矩阵：

$$\mathbf{A} = \begin{bmatrix} 2 & -2 \\ -1 & 3 \end{bmatrix}$$

**解**

第一步是求矩阵的特征值：

$$|[\mathbf{A} - \lambda\mathbf{I}]| = \left|\begin{bmatrix} 2-\lambda & -2 \\ -1 & 3-\lambda \end{bmatrix}\right| = \lambda^2 - 5\lambda + 4 = (\lambda - 1)(\lambda - 4) = 0$$

所以，特征值是 $\lambda_1 = 1$，$\lambda_2 = 4$。再求特征向量：

$$[\mathbf{A} - \lambda\mathbf{I}]\mathbf{z} = \begin{bmatrix} 2-\lambda & -2 \\ -1 & 3-\lambda \end{bmatrix}\mathbf{z} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

当 $\lambda = \lambda_1 = 1$ 时，有

$$\begin{bmatrix} 1 & -2 \\ -1 & 2 \end{bmatrix}\mathbf{z}_1 = \begin{bmatrix} 1 & -2 \\ -1 & 2 \end{bmatrix}\begin{bmatrix} \mathscr{Z}_{11} \\ \mathscr{Z}_{21} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

或

$$\mathscr{Z}_{11} = 2\mathscr{Z}_{21}$$

所以第一个特征向量是

$$\mathbf{z}_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \qquad \boxed{6\text{-}22}$$

或是该向量的任意倍数。

当 $\lambda = \lambda_2 = 4$ 时，有

$$\begin{bmatrix} -2 & -2 \\ -1 & -1 \end{bmatrix}\mathbf{z}_1 = \begin{bmatrix} -2 & -2 \\ -1 & -1 \end{bmatrix}\begin{bmatrix} \mathscr{Z}_{12} \\ \mathscr{Z}_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

或

$$\mathscr{Z}_{12} = -\mathscr{Z}_{22}$$

所以第二个特征向量是

$$\mathbf{z}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

或该向量的任意倍数。

为了对角化该矩阵，我们用式(6.69)：

$$\mathbf{A}' = [\mathbf{B}^{-1}\mathbf{A}\mathbf{B}]$$

其中

$$\mathbf{B} = [\mathbf{z}_1 \quad \mathbf{z}_2] = \begin{bmatrix} 2 & 1 \\ 1 & -1 \end{bmatrix}$$

所以有

$$\mathbf{A}' = \begin{bmatrix} 1/3 & 1/3 \\ 1/3 & -2/3 \end{bmatrix}\begin{bmatrix} 2 & -2 \\ -1 & 3 \end{bmatrix}\begin{bmatrix} 2 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

**P6.6** 假设变换 $\mathscr{A}: \mathfrak{R}^3 \to \mathfrak{R}^2$ 相对于标准基集的矩阵表示为

$$\mathbf{A} = \begin{bmatrix} 3 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

6-23 求该变换相对于如下基集的矩阵：

$$T = \left\{\begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -2 \\ 3 \end{bmatrix}\right\}, W = \left\{\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -2 \end{bmatrix}\right\}$$

**解**

第一步是构造如下两个矩阵：

$$\mathbf{B}_t = \begin{bmatrix} 2 & 0 & 0 \\ 0 & -1 & -2 \\ 1 & 0 & 3 \end{bmatrix}, \mathbf{B}_w = \begin{bmatrix} 1 & 0 \\ 0 & -2 \end{bmatrix}$$

现在，利用式(6.33)形成新的矩阵表示：

$$\mathbf{A}' = [\mathbf{B}_w^{-1}\mathbf{A}\mathbf{B}_t]$$

$$\mathbf{A}' = \begin{bmatrix} 1 & 0 \\ 0 & -\frac{1}{2} \end{bmatrix}\begin{bmatrix} 3 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 2 & 0 & 0 \\ 0 & -1 & -2 \\ 1 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 6 & 1 & 2 \\ -\frac{1}{2} & 0 & -\frac{3}{2} \end{bmatrix}$$

所以上面矩阵就是该变换相对于基集 $T$ 和 $W$ 的矩阵表示。

**P6.7** 假设变换 $\mathscr{A}: \mathfrak{R}^2 \to \mathfrak{R}^2$。$\mathfrak{R}^2$ 的一个基是 $V = \{\mathscr{v}_1, \mathscr{v}_2\}$。

（i）当给定如下等式时，求变换 $\mathscr{A}$ 相对于基集 $V$ 的矩阵表示：

$$\mathscr{A}(\mathscr{v}_1) = \mathscr{v}_1 + 2\mathscr{v}_2$$

$$\mathscr{A}(\mathscr{v}_2) = \mathscr{v}_1 + \mathscr{v}_2$$

（ii）假设有一个新的基集 $W = \{\mathscr{w}_1, \mathscr{w}_2\}$。当给定如下等式时，求变换 $\mathscr{A}$ 相对于基集 $W$ 的矩阵表示：

$$\mathscr{w}_1 = \mathscr{v}_1 + \mathscr{v}_2$$

$$\mathscr{w}_2 = \mathscr{v}_1 - \mathscr{v}_2$$

**解**

6-24 （i）如同在式(6.6)中所定义的，两个等式分别给出了矩阵的两列。因此所求的矩阵是

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}$$

（ii）可以按照 $V$ 的基向量的形式将 $W$ 的基向量表示为

$$\mathbf{w}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{w}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

现在就可以构造出用来进行相似变换的基矩阵：

$$\mathbf{B}_w = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

然后可以从式(6.33)得到如下新的矩阵表示形式：

$$\mathbf{A}' = [\mathbf{B}_w^{-1}\mathbf{A}\mathbf{B}_w]$$

$$\mathbf{A}' = \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & -1/2 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 5/2 & 1/2 \\ -1/2 & -1/2 \end{bmatrix}$$

**P6.8** 假设所有阶数小于等于 2 的多项式的向量空间为 $P^2$，该向量空间的一个基是 $V = \{1, t, t^2\}$，$\mathcal{D}$ 是一个微分变换。

（i）求这个变换相对于基集 $V$ 的矩阵表示；

（ii）求变换的特征值和特征向量。

**解**

（i）第一步是对每个基向量进行变换：

$$\mathcal{D}(0) = 0 = (0)1 + (0)t + (0)t^2$$
$$\mathcal{D}(t) = 1 = (1)1 + (0)t + (0)t^2$$
$$\mathcal{D}(t^2) = 2t = (0)1 + (2)t + (0)t^2 \qquad \boxed{6\text{-}25}$$

变换的矩阵是

$$\mathbf{D} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{bmatrix}$$

（ii）为了求特征值，必须求解

$$|[\mathbf{D} - \lambda\mathbf{I}]| = \left|\begin{bmatrix} -\lambda & 1 & 0 \\ 0 & -\lambda & 2 \\ 0 & 0 & -\lambda \end{bmatrix}\right| = -\lambda^3 = 0$$

所以所有三个特征值都是 0。为了求特征向量，需要求解

$$[\mathbf{D} - \lambda\mathbf{I}]\mathbf{z} = \begin{bmatrix} -\lambda & 1 & 0 \\ 0 & -\lambda & 2 \\ 0 & 0 & -\lambda \end{bmatrix}\mathbf{z} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

对于 $\lambda = 0$ 有

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} \mathcal{Z}_1 \\ \mathcal{Z}_2 \\ \mathcal{Z}_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

这意味着

$$\mathcal{Z}_2 = \mathcal{Z}_3 = 0$$

所以，只能得到一个特征向量：

$$\mathbf{z} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

由此可以看出，仅当多项式的导数是其自身的缩放形式时，多项式是一个常数(0 阶多项式)。

6-26

**P6.9** 设有一个变换 $\mathcal{A}: \Re^2 \rightarrow \Re^2$。图 6-16 给出了该变换的两个实例。求变换相对于标准基集的矩阵表示。

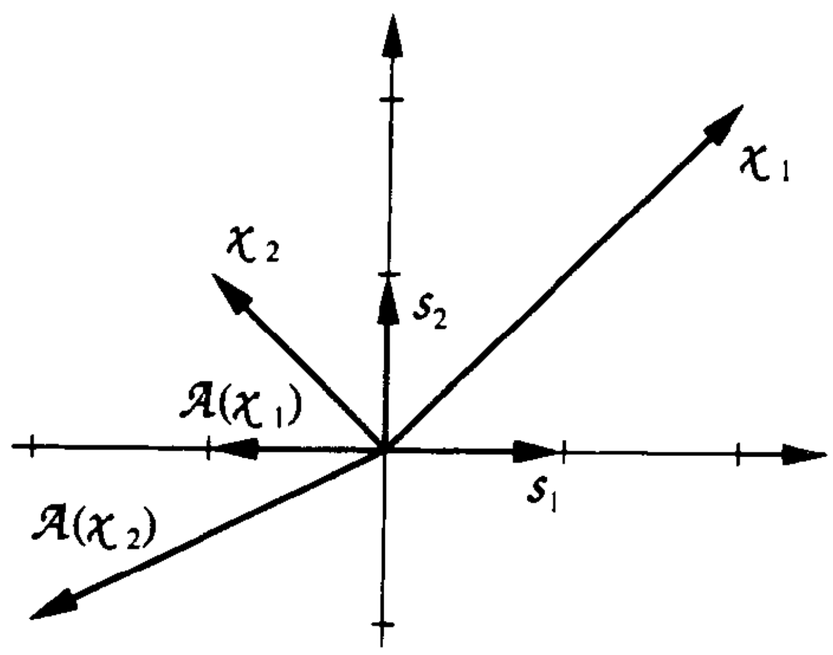



图 6-16 例题 P6.9 的变换

**解**

对于这个问题而言，因为并不知道基向量是如何被变换的，所以不能用式(6.6)求解变换的矩阵表示。但是，知道如何对图中的两个向量进行变换，也知道如何按照标准基集的形式来表示这两个向量。根据图 6-16，可以写出如下等式：

$$\mathbf{A}\begin{bmatrix}2\\2\end{bmatrix}=\begin{bmatrix}-1\\0\end{bmatrix},\quad \mathbf{A}\begin{bmatrix}-1\\1\end{bmatrix}=\begin{bmatrix}-2\\-1\end{bmatrix}$$

将上面两个等式合并在一起：

$$\mathbf{A}\begin{bmatrix}2 & -1\\2 & 1\end{bmatrix}=\begin{bmatrix}-1 & -2\\0 & -1\end{bmatrix}$$

所以

$$\mathbf{A}=\begin{bmatrix}-1 & -2\\0 & -1\end{bmatrix}\begin{bmatrix}2 & -1\\2 & 1\end{bmatrix}^{-1}=\begin{bmatrix}-1 & -2\\0 & -1\end{bmatrix}\begin{bmatrix}1/4 & 1/4\\-1/2 & 1/2\end{bmatrix}=\begin{bmatrix}3/4 & -5/4\\1/2 & -1/2\end{bmatrix}$$

这就是变换相对于标准基集的矩阵表示。

在 *Neural Network Design Demonstration* 中的 *Linear Transformations*(**nnd6lt**)用到了这个过程。

6-27

## 6.5 结束语

这一章复习了线性变换及其矩阵的一些性质，这些内容对学习神经网络至关重要。特征值、特征向量、基变换(相似变换)和对角化等概念在后面各章还会被经常用到。如果没有这些线性代数的背景知识，那么读者只能肤浅地学习神经网络。

6-28

下一章将应用这些线性代数知识来分析第一个神经网络训练算法的操作——Hebb 规则。

## 参考文献

[Brog91] W. L. Brogan, *Modern Control Theory*, 3rd Ed., Englewood Cliffs, NJ: Prentice-

Hall, 1991.

该书是一本关于线性系统学科的非常好的书。其前半部分全部是有关线性代数的内容，也有一些写得非常好的章节讨论线性微分方程求解、线性系统和非线性系统的稳定性等内容。该书有许多习题。

[Stra76] G. Strang, *Linear Algebra and Its Applications*, New York: Academic Press, 1980.

这是 Strang 撰写的一本优秀的线性代数基础教材。该书包含了许多线性代数应用实例。

6-29

## 习题

**E6.1** 矩阵的转置操作是一个线性变换吗?

**E6.2** 参考图 6-12 中所示的神经网络模型。请说明：如果基向量 **b** 等于 0，那么神经网络完成的是一个线性操作。

**E6.3** 考虑图 6-17 中的线性变换。

(i) 求这个变换相对于标准基集的矩阵表示；

(ii) 求该变换相对于基集 $\{v_1, v_2\}$ 的矩阵表示。

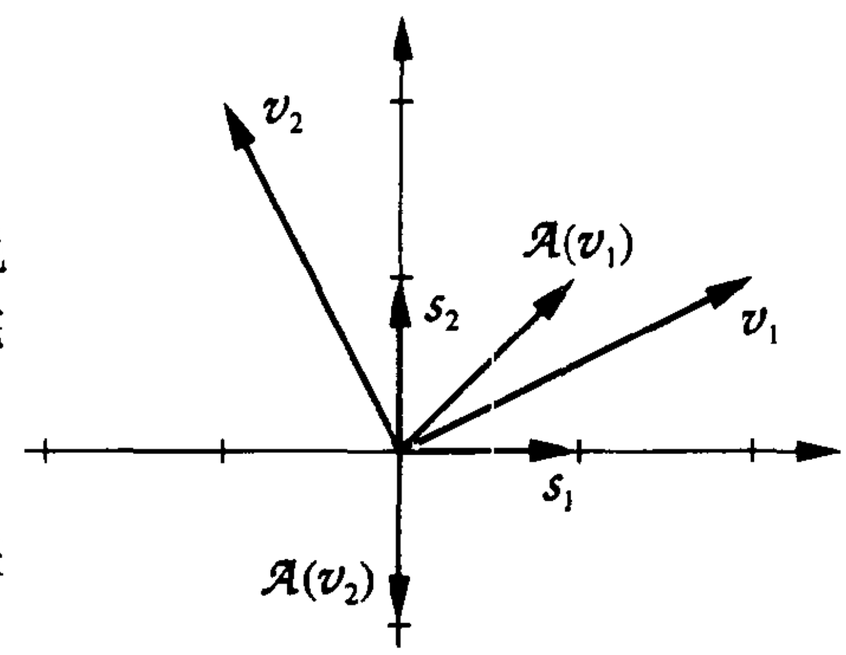



图 6-17　习题 E6.3 的变换实例

**E6.4** 设复数空间为一个向量空间 X，X 的基是 $\{1+j, 1-j\}$。$\mathcal{A}$: $X \to X$ 是一个乘 $(1+j)$ 的操作(即 $\mathcal{A}(x) = (1+j)x$)。

(i) 求变换 $\mathcal{A}$ 相对于上面所给的基集的矩阵表示；

(ii) 求变换的特征值和特征向量；

(iii) 将特征向量作为基向量，求变换 $\mathcal{A}$ 相应的矩阵表示；

(iv) 用 MATLAB 验证(ii)和(iii)。

6-30

**E6.5** 假设有一个从二次多项式空间到三次多项式空间的变换：$\mathcal{A}$: $P^2 \to P^3$。其定义如下：

$$x = a_0 + a_1 t + a_2 t^2$$

$$\mathcal{A}(x) = a_0(t+1) + a_1(t+1)^2 + a_2(t+1)^3$$

求这个变换相对于基集 $V^2 = \{1, t, t^2\}$ 和 $V^3 = \{1, t, t^2, t^3\}$ 的矩阵表示。

**E6.6** 考虑 $\alpha\sin(t+\phi)$ 形式的函数的空间。这个空间的一个基集是 $V = \{\sin t, \cos t\}$。设 $\mathcal{D}$ 是一个微分变换。

(i) 求变换 $\mathcal{D}$ 相对于基集 $V$ 的矩阵表示；

(ii) 求变换的特征值和特征向量。请按照数列的形式和 $t$ 的函数形式表示特征向量。

(iii) 将特征向量作为基向量，求变换相应的矩阵表示。

**E6.7** 设 $P^2$ 和 $P^3$ 分别是二次和三次多项式的向量空间。求积分变换 $I$: $P^2 \to P^3$ 相对于基集 $V^2 = \{1, t, t^2\}$ 和 $V^3 = \{1, t, t^2, t^3\}$ 的矩阵表示。

**E6.8** 设某个线性变换 $\mathcal{A}$: $\Re^2 \to \Re^3$ 相对于标准基集有如下矩阵表示形式：

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

求这个变换相对于如下新基集的矩阵表示。

$$V = \left\{ \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \begin{bmatrix} 2 \\ 5 \end{bmatrix} \right\}$$

6-31 **E6.9** 假设我们知道某个线性变换 $\mathscr{A}: \Re^2 \rightarrow \Re^3$ 有如下特征值和特征向量：

$$\lambda_1 = 1, \quad \mathbf{z}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}; \quad \lambda_2 = 2, \quad \mathbf{z}_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

（注意：这里是相对于标准基集来表示特征向量的。）

（i）求变换 $\mathscr{A}$ 相对于标准基集的矩阵表示；

（ii）求变换相对于如下新基集的矩阵表示：

$$V = \left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right\}$$

**E6.10** 设 $\Re^2$ 的基集为

$$V = \{\mathbf{v}_1, \mathbf{v}_2\} = \left\{ \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ -2 \end{bmatrix} \right\}$$

（注意：这里是相对于标准基集来表示基向量的。）

（i）求这个基集的互逆基向量。

（ii）设变换 $\mathscr{A}: \Re^2 \rightarrow \Re^2$ 相对于 $\Re^2$ 中的标准基集的矩阵表示为

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}$$

请按照基集 $V$ 的形式，求 $\mathbf{A}\mathbf{v}_1$ 的展式。（采用互逆基向量。）

（iii）按照基集 $V$ 的形式，求 $\mathbf{A}\mathbf{v}_2$ 的展式。

6-32 （iv）求变换 $\mathscr{A}$ 相对于基集 $V$ 的矩阵表示形式。（这一步应无需更多的计算。）

# 第7章　有监督的 Hebb 学习

## 7.1　目的

Hebb 规则是最早的神经网络学习规则之一，由 Donald Hebb 在 1949 年作为大脑的一种神经元突触调整的可能机制而提出，从那以后 Hebb 规则就一直用于人工神经网络的训练。

本章将运用前两章提出的线性代数的概念阐述 Hebb 学习的工作原理，并说明如何把 Hebb 规则用于训练人工神经网络。 7-1

## 7.2　理论和实例

上世纪初，Donald O. Hebb 出生于 Nova Scotia 的 Chester。他原想做一名小说家，并于 1925 年在 Halifax 的 Dalhousie 大学获得英语学位。考虑到作为一名一流小说家必须对人类本质有深刻的认识，毕业后的 Hebb 开始研究弗洛伊德，并对心理学产生了浓厚兴趣。后来，他到 McGill 大学攻读心理学硕士学位，并完成了关于巴甫洛夫条件反射理论的学位论文。1936 年，他获得哈佛大学的博士学位，他的学位论文研究了对老鼠视觉的早期实验的效果。后来，他加入蒙特利尔神经学院，研究脑外科手术后病人智能变化的程度。1942 年，Hebb 转到设在佛罗里达研究灵长类动物的 Yerkes 实验室，在那儿，他从事对非洲黑猩猩行为的研究。

1949 年，Hebb 在其《*The Organization of Behavior*》一书[Hebb49]中总结了他 20 年来的研究工作。该书的主导思想是：行为可以由神经元的活动来解释。而这与拥有像 B. F. Skinner 这样的支持者的行为主义心理学会的观点大相径庭，他们强调刺激与反射的关联，并反对任何生理学假说。这是一场自上而下与自下而上的哲学上的争论。Hebb 这样阐述他的方法："这种称为学习的方法因而要求个体尽量多地学习其大脑各组成部分所产生的知识(主要在生理学领域)，并尽其所能将这些知识与行为相关联(主要在心理学范畴)，在对大脑各个组成部分的活动加以综合之后，对行为进行预测，并与实际的行为相对比，从中发现差异，进而据此对整个大脑的工作原理做出深入的探索。"

**Hebb 假设**　《*The Organization of Behavior*》一书中最著名的思想就是现在称为 Hebb 学习的一个假设："*当细胞 A 的轴突到细胞 B 的距离近到足够激励它，且反复地或持续地刺激 B，那么在这两个细胞或一个细胞中将会发生某种增长过程或代谢反应，增加 A 对细胞 B 的刺激效果。*"

这个假设提出了一种细胞级学习的物质机制。尽管 Hebb 从未宣称其理论具有可靠的生理学证据，但是其后的研究表明某些细胞的确表现出 Hebb 学习的行为。Hebb 的理论对当今的神经科学研究仍具有影响。

同历史上许多思想一样，Hebb 假设也并不是全新的，Hebb 本人也强调了这一点。它吸收了许多其他科学家的思想，如弗洛伊德，以及心理学家和哲学家 William James 在 1890 年提出的相联原理："当两个大脑过程同活跃或立即相继活跃时，其中之一会重复地把活跃状态 7-2

传播给另外一个。”

### 7.2.1 线性联想器

**线性联想器** Hebb 学习规则能用于和多种神经网络结构相组合。在首次讨论 Hebb 学习时，将采用一种非常简单的结构。这样读者就能够集中研究学习规则而不专注于结构。这里将使用的网络被称为线性联想器(如图 7-1 所示，它是由 James Anderson [Ande72]和 Teuvo Kohonen [Koho72]分别独立提出的)。

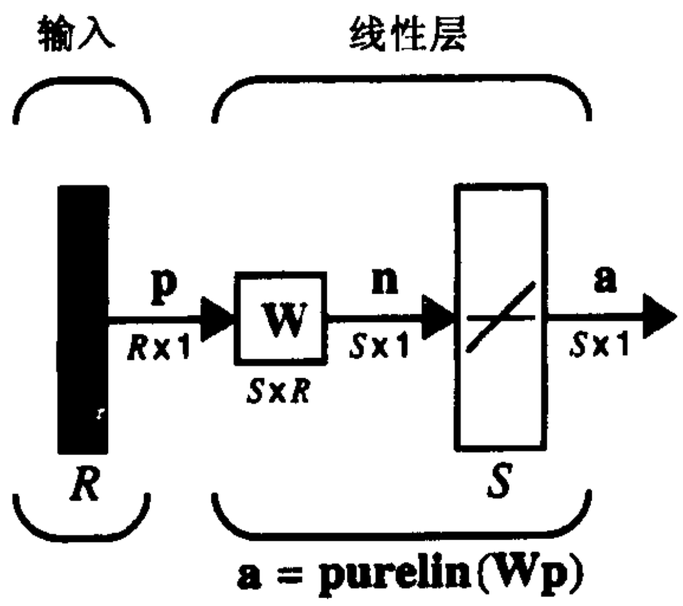



图 7-1 线性联想器

输出向量 **a** 由输入向量 **p** 根据下式决定：

$$\mathbf{a} = \mathbf{W}\mathbf{p} \tag{7.1}$$

或

$$a_i = \sum_{j=1}^{R} w_{ij} p_j \tag{7.2}$$

**联想存储器** 线性联想器是被称为联想存储器的神经网络类型中的一种神经网络，联想存储器的任务是学习 $Q$ 对标准输入/输出向量：

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \cdots, \{\mathbf{p}_Q, \mathbf{t}_Q\} \tag{7.3}$$

即是如果网络接收一个输入 $\mathbf{p} = \mathbf{p}_q$，那么它应能产生一个输出 $\mathbf{a} = \mathbf{t}_q$，这里 $q = 1, 2, \cdots, Q$。另外，如果输入发生了微小变化(即 $\mathbf{p} = \mathbf{p}_q + \delta$)，那么网络的输出只应发生轻微的改变(即 $\mathbf{a} = \mathbf{t}_q + \varepsilon$)。

7-3

### 7.2.2 Hebb 规则

为了将 Hebb 假设用于训练线性联想器的权值矩阵，那么又如何给出 Hebb 假设的数学解释呢？首先，再次重述一下该假设：若一条突触两侧的两个神经元同时被激活，那么突触的强度将会增大。

**Hebb 规则** 请注意在式(7.2)中，输入 $p_j$ 和输出 $a_i$ 之间的连接(突触)是权值 $w_{ij}$。所以，Hebb 假设意味着：如果一个正的输入 $p_j$ 产生一个正的输出 $a_i$，那么应该增加 $w_{ij}$的值。这就是该假设的一种数学解释，即

$$w_{ij}^{new} = w_{ij}^{old} + \alpha f_i(a_{iq}) g_j(p_{jq}) \tag{7.4}$$

这里 $p_{jq}$为第 $q$ 个输入向量 $\mathbf{p}_q$ 的第 $j$ 个元素，$a_{iq}$为把第 $q$ 个输入向量提交给网络时网络输出

的第 $i$ 个元素，$\alpha$ 是一个称为学习速度的正的常数。这个等式表明：权值 $w_{ij}$ 的变化与突触两边的活跃函数值的乘积成比例。本章把式(7.4)简化成如下形式。

$$w_{ij}^{new} = w_{ij}^{old} + \alpha a_{iq} p_{jq} \tag{7.5}$$

请注意：这里在严格解释的基础上扩展了 Hebb 假设，权值的变化与突触每侧活跃值的乘积成比例。因此，权值不仅在 $p_j$ 和 $a_i$ 均为正时增大，而且在 $p_j$ 和 $a_i$ 均为负时也会增大。另外，只要 $p_j$ 和 $a_i$ 的符号相反，那么 Hebb 规则的这种实现将使得权值减小。

式(7.5)定义的 Hebb 规则是一种无监督的学习规则，它不需要关于目标输出的任何相关信息。本章只关注用于有监督学习的 Hebb 规则(无监督学习的 Hebb 规则将在第 13 章讨论)，并且假定每个输入向量相应的目标输出都是已知的。对于有监督的 Hebb 规则而言，这里将用目标输出代替实际输出。由此，算法被告知的就是网络应该做什么，而不是网络当前正在做什么。得到的等式为

$$w_{ij}^{new} = w_{ij}^{old} + t_{iq} p_{jq} \tag{7.6}$$

其中 $t_{iq}$ 是第 $q$ 个目标向量 $\mathbf{t}_q$ 的第 $i$ 个元素(为了简单起见，这里设学习速度 $\alpha$ 的值为 1)。

请注意，式(7.6)也可写为如下向量形式：

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{t}_q \mathbf{p}_q^T \tag{7.7}$$

7-4

如果假定将权值矩阵初始化为 0，然后 $Q$ 个输入/输出对依次应用式(7.7)，那么有

$$\mathbf{W} = \mathbf{t}_1 \mathbf{p}_1^T + \mathbf{t}_2 \mathbf{p}_2^T + \cdots + \mathbf{t}_Q \mathbf{p}_Q^T = \sum_{q=1}^{Q} \mathbf{t}_q \mathbf{p}_q^T \tag{7.8}$$

用矩阵形式可以表示为

$$\mathbf{W} = [\mathbf{t}_1, \mathbf{t}_2 \cdots, \mathbf{t}_Q] \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \vdots \\ \mathbf{p}_Q^T \end{bmatrix} = \mathbf{T}\mathbf{P}^T \tag{7.9}$$

其中

$$\mathbf{T} = [\mathbf{t}_1 \mathbf{t}_2 \cdots \mathbf{t}_Q], \quad \mathbf{P} = [\mathbf{p}_1 \mathbf{p}_2 \cdots \mathbf{p}_Q] \tag{7.10}$$

性能分析

下面分析线性联想器的 Hebb 学习的性能。首先设输入向量 $\mathbf{p}_q$ 为标准正交向量(向量之间是正交的，每个向量的长度为单位长)。如果将 $\mathbf{p}_k$ 输入到网络，那么网络产生的输出为

$$\mathbf{a} = \mathbf{W}\mathbf{p}_k = \left[\sum_{q=1}^{Q} \mathbf{t}_q \mathbf{p}_q^T\right] \mathbf{p}_k = \sum_{q=1}^{Q} \mathbf{t}_q (\mathbf{p}_q^T \mathbf{p}_k) \tag{7.11}$$

由于 $\mathbf{p}_q$ 为标准正交向量，所以有

$$(\mathbf{p}_q^T \mathbf{p}_k) = \begin{cases} 1, & q = k \\ 0, & q \neq k \end{cases} \tag{7.12}$$

因此式(7.11)可重写为

$$\mathbf{a} = \mathbf{W}\mathbf{p}_k = \mathbf{t}_k \tag{7.13}$$

此时，网络的输出等于其相应的目标输出。这表明：如果输入原型向量是标准正交向量，Hebb 规则就能为每个输入生成正确的输出结果。

7-5

但是当输入原型向量不是正交向量时，又将如何呢？假设每个向量 $\mathbf{p}_q$ 为单位向量，但是它们之间并不正交，那么式(7.11)变为

$$\mathbf{a} = \mathbf{W}\mathbf{p}_k = \mathbf{t}_k + \underbrace{\sum_{q \neq k} \mathbf{t}_q(\mathbf{p}_q^T \mathbf{p}_k)}_{\text{误差}} \tag{7.14}$$

由于这些向量不是正交的，所以网络的输出有误差。误差的大小取决于原型输入模式之间的相关总和。

例如，假设原型输入/输出向量为

$$\left\{\mathbf{p}_1 = \begin{bmatrix} 0.5 \\ -0.5 \\ 0.5 \\ -0.5 \end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}\right\}, \quad \left\{\mathbf{p}_2 = \begin{bmatrix} 0.5 \\ 0.5 \\ -0.5 \\ -0.5 \end{bmatrix}, \mathbf{t}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right\} \tag{7.15}$$

（可以验证这两个输入向量是标准正交向量。）那么网络的权值矩阵为

$$\mathbf{W} = \mathbf{T}\mathbf{P}^T = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} 0.5 & -0.5 & 0.5 & -0.5 \\ 0.5 & 0.5 & -0.5 & -0.5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix} \tag{7.16}$$

用上述两个原型输入验证该权值矩阵，有

$$\mathbf{W}\mathbf{p}_1 = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix}\begin{bmatrix} 0.5 \\ -0.5 \\ 0.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \tag{7.17}$$

$$\mathbf{W}\mathbf{p}_2 = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix}\begin{bmatrix} 0.5 \\ 0.5 \\ -0.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{7.18}$$

7-6 获得成功！网络的输出与目标输出相等。

现在，再次考虑在第 3 章中给出的苹果和橘子的识别问题。其原型输入为

$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \text{（橘子）}, \quad \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} \text{（苹果）} \tag{7.19}$$

（注意：$\mathbf{p}_1$，$\mathbf{p}_2$ 不是正交向量）将 $\mathbf{p}_1$，$\mathbf{p}_2$ 归格化，并选取期望输出为 -1 和 1，则有

$$\left\{\mathbf{p}_1 = \begin{bmatrix} 0.5774 \\ -0.5774 \\ -0.5774 \end{bmatrix}, \mathbf{t}_1 = [-1]\right\}, \left\{\mathbf{p}_2 = \begin{bmatrix} 0.5774 \\ 0.5774 \\ -0.5774 \end{bmatrix}, \mathbf{t}_2 = [1]\right\} \tag{7.20}$$

这时的权值矩阵为

$$\mathbf{W} = \mathbf{T}\mathbf{P}^T = [-1 \quad 1]\begin{bmatrix} 0.5774 & -0.5774 & -0.5774 \\ 0.5774 & 0.5774 & -0.5774 \end{bmatrix} = [0 \quad 1.1548 \quad 0] \tag{7.21}$$

所以，如果采用上面的两个原型输入模式，分别有

$$\mathbf{W}\mathbf{p}_1 = [0 \quad 1.1548 \quad 0]\begin{bmatrix} 0.5774 \\ -0.5774 \\ -0.5774 \end{bmatrix} = [-0.6668] \tag{7.22}$$

$$\mathbf{W}\mathbf{p}_2 = [0 \quad 1.1548 \quad 0]\begin{bmatrix} 0.5774 \\ 0.5774 \\ -0.5774 \end{bmatrix} = [0.6668] \tag{7.23}$$

这里的输出接近目标输出，但与目标输出并不能十分匹配。

### 7.2.3　仿逆规则

当样本输入模式非正交时，Hebb 规则会产生误差。有多种方法可以减小这种误差。本节我们讨论其中之一，即仿逆规则。

线性联想器的任务是对于输入 $\mathbf{p}_q$ 产生输出 $\mathbf{t}_q$，即

$$\mathbf{W}\mathbf{p}_q = \mathbf{t}_q, \quad q = 1,2,\cdots,Q \tag{7.24}$$

7-7

如果无法找到使这些等式绝对成立的权值矩阵，那么也希望找到使它们近似成立的权值矩阵。一种方法是：选取一个权值矩阵，使下列性能参数最小化：

$$F(\mathbf{W}) = \sum_{q=1}^{Q} \| \mathbf{t}_q - \mathbf{W}\mathbf{p}_q \|^2 \tag{7.25}$$

如果样本输入向量 $\mathbf{p}_q$ 是标准正交的，那么用 Hebb 规则来求权值矩阵 $\mathbf{W}$，则 $F(\mathbf{W})$为零。如果输入向量不是标准正交的，那么用 Hebb 规则得到的 $F(\mathbf{W})$将不等于零，而且 $F(\mathbf{W})$是否为最小值也不十分清楚。可以证明，如果使用下面将定义的仿逆规则，则所得权值矩阵可使 $F(\mathbf{W})$最小化。

将式(7.24)写成矩阵形式：

$$\mathbf{W}\mathbf{P} = \mathbf{T}, \tag{7.26}$$

其中

$$\mathbf{T} = [\mathbf{t}_1 \quad \mathbf{t}_2 \quad \cdots \quad \mathbf{t}_Q], \mathbf{P} = [\mathbf{p}_1 \quad \mathbf{p}_2 \quad \cdots \quad \mathbf{p}_Q] \tag{7.27}$$

则式(7.25)可以写为

$$F(\mathbf{W}) = \| \mathbf{T} - \mathbf{W}\mathbf{P} \|^2 = \| \mathbf{E} \|^2 \tag{7.28}$$

这里

$$\mathbf{E} = \mathbf{T} - \mathbf{W}\mathbf{P} \tag{7.29}$$

且

$$\| \mathbf{E} \|^2 = \sum_i \sum_j e_{ij}^2 \tag{7.30}$$

请注意：如果式(7.26)有解，那么 $F(\mathbf{W})$可以为零。若存在矩阵 $\mathbf{P}$ 的逆，则解为

$$\mathbf{W} = \mathbf{T}\mathbf{P}^{-1} \tag{7.31}$$

然而，这是很少有可能的。通常矩阵 $\mathbf{P}$ 的列向量 $\mathbf{p}_q$ 是线性无关的，但 $\mathbf{p}_q$ 的维数 $R$ 比 $\mathbf{p}_q$ 的向量个数 $Q$ 要大，所以 $\mathbf{P}$ 不是一个方阵，不存在确切的逆阵。

7-8

参考文献[Albe72]表明使式(7.25)最小化的权值矩阵可由仿逆规则给出：

$$\mathbf{W} = \mathbf{T}\mathbf{P}^{+} \tag{7.32}$$

其中 $\mathbf{P}^{+}$ 为 Moore－Penrose 仿逆。实矩阵 $\mathbf{P}$ 的仿逆是满足

$$\boxed{\begin{aligned}&\mathbf{P}\mathbf{P}^{+}\mathbf{P}=\mathbf{P}\\&\mathbf{P}^{+}\mathbf{P}\mathbf{P}^{+}=\mathbf{P}^{+}\\&\mathbf{P}^{+}\mathbf{P}=(\mathbf{P}^{+}\mathbf{P})^{T}\\&\mathbf{P}\mathbf{P}^{+}=(\mathbf{P}\mathbf{P}^{+})^{T}\end{aligned}} \tag{7.33}$$

的惟一的矩阵。当矩阵 $\mathbf{P}$ 的行数 $R$ 大于其列数 $Q$，且 $\mathbf{P}$ 的列向量线性无关时，其仿逆为

$$\mathbf{P}^{+}=(\mathbf{P}^{T}\mathbf{P})^{-1}\mathbf{P}^{T} \tag{7.34}$$

为了验证仿逆规则(式 7.32)，再考虑苹果、橘子的识别问题。输入/输出原型向量为

$$\left\{\mathbf{p}_1=\begin{bmatrix}1\\-1\\-1\end{bmatrix},\mathbf{t}_1=[1]\right\},\left\{\mathbf{p}_2=\begin{bmatrix}1\\1\\-1\end{bmatrix},\mathbf{t}_2=[1]\right\} \tag{7.35}$$

(请注意：使用仿逆规则时不需要对输入向量进行规格化。)

用式(7.32)计算得到的权值矩阵为

$$\mathbf{W}=\mathbf{T}\mathbf{P}^{+}=[-1\quad 1]\left(\begin{bmatrix}1&1\\-1&1\\-1&-1\end{bmatrix}\right)^{+} \tag{7.36}$$

这里，用式(7.34)计算仿逆：

$$\mathbf{P}^{+}=(\mathbf{P}^{T}\mathbf{P})^{-1}\mathbf{P}^{T}=\begin{bmatrix}3&1\\1&3\end{bmatrix}^{-1}\begin{bmatrix}1&-1&-1\\1&1&-1\end{bmatrix}=\begin{bmatrix}0.25&-0.5&-0.25\\0.25&0.5&-0.25\end{bmatrix} \tag{7.37}$$

7-9 这就得到了如下权值矩阵：

$$\mathbf{W}=\mathbf{T}\mathbf{P}^{+}=[-1\quad 1]\begin{bmatrix}0.25&-0.5&-0.25\\0.25&0.5&-0.25\end{bmatrix}=[0\quad 1\quad 0] \tag{7.38}$$

用该权值矩阵作用于两个原型模式：

$$\mathbf{W}\mathbf{p}_1=[0\quad 1\quad 0]\begin{bmatrix}1\\-1\\-1\end{bmatrix}=[-1] \tag{7.39}$$

$$\mathbf{W}\mathbf{p}_2=[0\quad 1\quad 0]\begin{bmatrix}1\\1\\-1\end{bmatrix}=[1] \tag{7.40}$$

这里的网络输出与期望输出精确匹配。将此结果与 Hebb 规则的性能比较。正如可以从式(7.22)和式(7.23)看出的那样，Hebb 的输出是接近的，而应用仿逆规则却能产生精确的结果。

### 7.2.4 应用

**自联想存储器** 现在将 Hebb 规则应用用于一个大大简化了的实际模式识别问题。这里，将使用一种特殊类型的联想存储器——自联想存储器。在自联想存储器中，期望输出向量等于网络的输入向量(即 $\mathbf{t}_q=\mathbf{p}_q$)。这里将用自联想存储器存储一组模式，并且当其输入模式有所“破损”时，它仍然能够将其复原。

这里要存储的模式如图 7-2 所示(由于使用了自联想存储器，这些模式既是输入向量又是目标向量)。它们分别是用 6×5 栅格所显示的数字{0，1，2}。这里需要将这些数字转换成向量表示形式，分别作为网络的原型模式。如果每个白色的方格用 -1 表示，每个黑色的方格用1表示，那么一次扫描6×5栅格中的一列，就可以生成这些输入向量。例如，第一个栅格所表示的原型模式相应的输入向量为

$\mathbf{p}_1,\mathbf{t}_1$　$\mathbf{p}_2,\mathbf{t}_2$　$\mathbf{p}_3,\mathbf{t}_3$

图 7-2　自联想存储器要存储的模式

$$\mathbf{p}_1 = [-1\ 1\ 1\ 1\ 1\ -1\ 1\ -1\ -1\ -1\ -1\ 1\ 1\ -1\ \ldots\ 1\ -1]^T \tag{7.41}$$

向量 $\mathbf{p}_1$，$\mathbf{p}_2$，$\mathbf{p}_3$ 分别与数字 0，1，2 相对应，使用 Hebb 规则求权值矩阵：

$$\mathbf{W} = \mathbf{p}_1\mathbf{p}_1^T + \mathbf{p}_2\mathbf{p}_2^T + \mathbf{p}_3\mathbf{p}_2^T \tag{7.42}$$

7-10

请注意：由于这是一个自联想存储器，所以这里用 $\mathbf{p}_q$ 代替了式(7.8)中的 $\mathbf{t}_q$。

因为样本向量的元素仅限于取两个值，这里将对线性联想器进行修改，以使其输出元单元也仅取值 -1 或 1。为此，可以用一个对称的硬极限传输函数代替原来的线性传输函数。修改后的网络如图 7-3 所示。

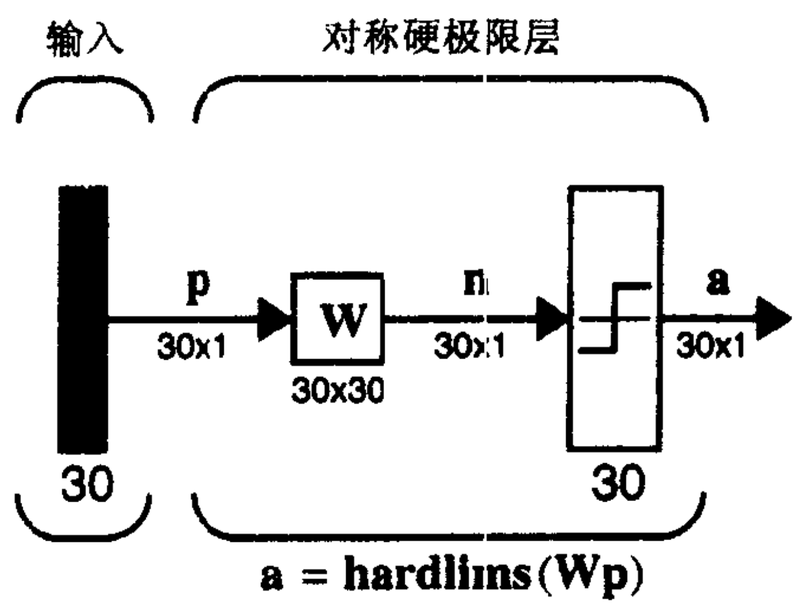



图 7-3　用于数字识别的自联想网络

现在来研究网络的运行情况。首先向网络提供破损的原型模式，然后检查网络的输出。在第一次测试中，将给网络提供的原型模式的下半部分隐去(如图 7-4 所示)，网络能够生成每个样本的正确的模式。

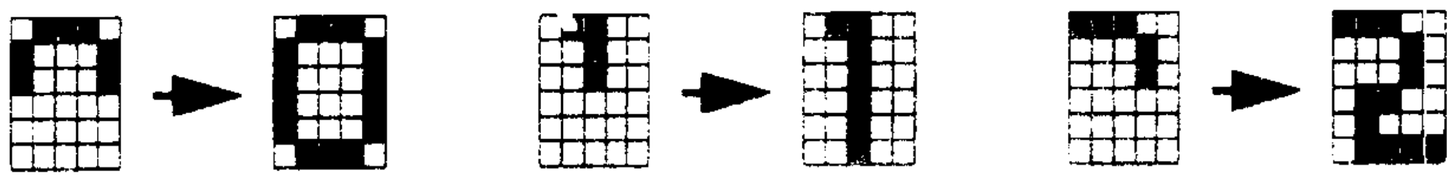


图 7-4　恢复隐去了 50% 的模式

在下一步测试中，去掉原型模式的更多的部分，图 7-5 给出去掉模式下面三分之二之后的模式。这时只有“1”被正确恢复。另外两个模式的的恢复结果与原型模式都不一样。这是联想存储器普遍存在的问题。这里希望能够设计出尽量减少产生这种错误模式数量的网络。在第 18 章讨论递归联想存储器时，还将讨论这一问题。

7-11

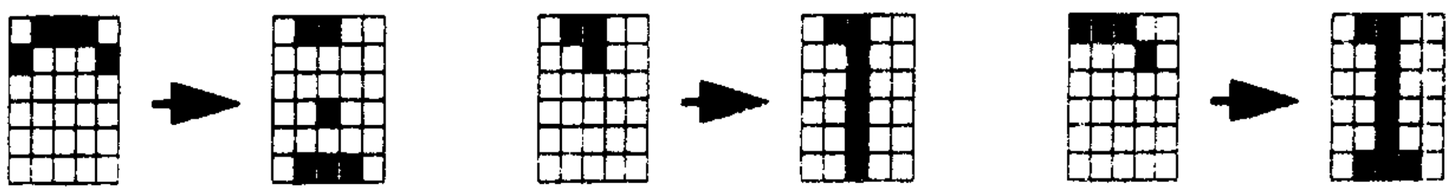


图 7-5　恢复隐去了 67% 的模式

在最后的测试中，将对加入噪声的原型模式测试自联想网络。通过随机地改变每一原型模式的 7 个元素来加入噪声。测试结果如图 7-6 所示，这里所有的模式都被正确恢复。

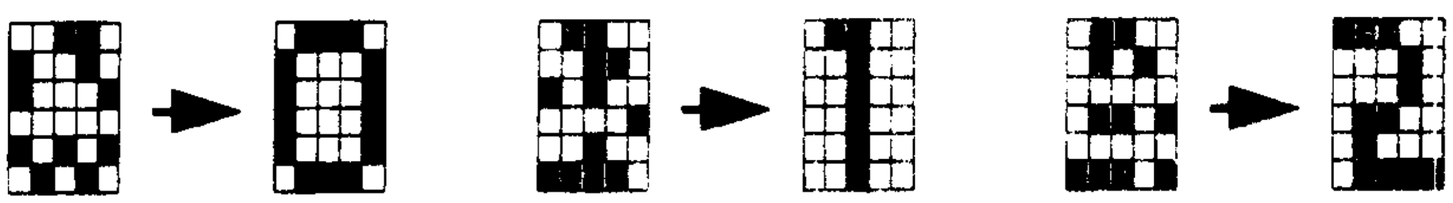


图 7-6　恢复带噪声的模式

可以使用 *Neural Network Design Demonstration* **Hebb Rule** (**nnd7hr**)对这类模式识别问题进行试验。

### 7.2.5 Hebb 学习的变形

基本的 Hebb 规则可以有许多变形。实际上，本书的后面其他章节所讨论的学习规则都与 Hebb 规则有关。

Hebb 规则的问题之一是：如果训练集中存在许多原型模式，Hebb 规则会使权值矩阵元素过多。再次考虑基本规则：

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{t}_q\mathbf{p}_q^T \tag{7.43}$$

可以使用一个称为学习速度的正参数 $\alpha$(小于 1)限制权值矩阵元素的增加量，即

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha\mathbf{t}_q\mathbf{p}_q^T \tag{7.44}$$

也可以再加上一个衰减项，使学习规则的行为像一个平滑过滤器，更加清晰地记忆最近的提供给网络的输入：

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha\mathbf{t}_q\mathbf{p}_q^T - \gamma\mathbf{W}^{old} = (1-\gamma)\mathbf{W}^{old} + \alpha\mathbf{t}_q\mathbf{p}_q^T \tag{7.45}$$

7-12 其中 $\gamma$ 为小于 1 的正的常数。如果 $\gamma$ 趋近于零，那么学习规则趋近于标准规则；如果 $\gamma$ 趋近于 1，那么学习规则将很快忘记旧的输入，而仅记忆最近的输入模式。据此可知，这些项的引入可以避免权值矩阵无限制地增大。

过滤权值变化和调整学习速度的思想非常重要，本书还将在第 10 章和第 12 到 16 章中再次对其进行讨论。

如果用期望输出与实际输出之差代替式(7.44)中的期望输出，那么可以得到另一个重要学习规则：

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha(\mathbf{t}_q - \mathbf{a}_q)\mathbf{p}_q^T \tag{7.46}$$

这就是所谓的增量规则，因为它使用了期望输出与实际输出之差。根据它的研究和提出者又被称为 Widrow-Hoff 算法。增量规则调整权值以使均方误差最小(参见第 10 章)，因而它与仿逆规则得到的结果相同，仿逆规则使误差平方和最小化(式(7.25))。增量规则的优点是每输入一个模式它就能更新一次权值，而仿逆规则要等待所有输入/输出模式已知后才能计算一次权值 。这种顺序的权值更新方法使得增量规则能适应变化的环境。第 10 章将详细讨论增量规则。

第 13 章将在不同情况下再次讨论基本的 Hebb 规则。本章仅使用了 Hebb 规则的一种有监督的学习形式。我们假定了网络的期望输出 $\mathbf{t}_q$ 为已知的，并能在学习规则中使用。第 13 章讨论的 Hebb 规则的无监督形式将使用实际的网络输出代替期望的网络输出，即如：

7-13

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha\mathbf{a}_q\mathbf{p}_q^T \tag{7.47}$$

其中 $\mathbf{a}_q$ 是给定 $\mathbf{p}_q$ 为输入时的网络输出(参见式(7.5))。Hebb 规则的这种无监督学习形式由于不需知道期望输出，实际上比有监督的 Hebb 规则更能够直接地说明 Hebb 的原理。

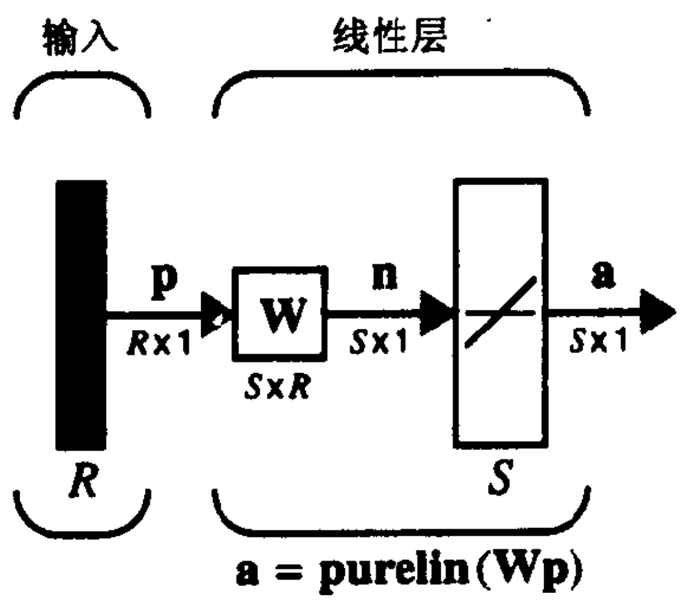
## 7.3 小结

### Hebb 假设

“当细胞 A 的轴突到细胞 B 的距离近到足够激励它，且反复地或持续地刺激 B，那么在这

两个或一个细胞中将发生某种增长过程或代谢反应，增加 A 对细胞 B 的刺激效果。”

**线性联想器**




**Hebb 规则**

$$w_{ij}^{new} = w_{ij}^{old} + t_{qi}p_{qj}$$

$$\mathbf{W} = \mathbf{t}_1\mathbf{p}_1^T + \mathbf{t}_2\mathbf{p}_2^T + \cdots + \mathbf{t}_Q\mathbf{p}_Q^T$$

$$\mathbf{W} = [\mathbf{t}_1\mathbf{t}_2\cdots\mathbf{t}_Q]\begin{bmatrix}\mathbf{p}_1^T\\ \mathbf{p}_2^T\\ \vdots\\ \mathbf{p}_Q^T\end{bmatrix} = \mathbf{T}\mathbf{P}^T$$

**仿逆规则**

$$\mathbf{W} = \mathbf{T}\mathbf{P}^+ \qquad 7\text{-}14$$

当 **P** 的行数 $R$ 大于其列数 $Q$ 且 **P** 的列向量线性无关时，仿逆可由下式求出：

$$\mathbf{P}^+ = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T$$

**Hebb 学习的变形**

**过滤学习**

$$\mathbf{W}^{new} = (1-\gamma)\mathbf{W}^{old} + \alpha\mathbf{t}_q\mathbf{P}_q^T$$

（参见第 14 章）

**增量规则**

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha(\mathbf{t}_q - \mathbf{a}_q)\mathbf{P}_q^T$$

（参见第 10 章）

**无监督的 Hebb 学习**

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha\mathbf{a}_q\mathbf{P}_q^T \qquad 7\text{-}15$$

（参见第 13 章）

## 7.4 例题

**P7.1** 考虑图 7-7 的线性联想器。设输入/输出样本向量为

$$\left\{\mathbf{p}_1=\begin{bmatrix}1\\-1\\1\\-1\end{bmatrix},\mathbf{t}_1=\begin{bmatrix}1\\-1\end{bmatrix}\right\},\left\{\mathbf{p}_2=\begin{bmatrix}1\\1\\-1\\-1\end{bmatrix},\mathbf{t}_2=\begin{bmatrix}1\\1\end{bmatrix}\right\}$$

（i）运用 Hebb 规则求该线性联想器的权值矩阵。

（ii）运用仿逆规则重复(i)题。

（iii）将输入 $\mathbf{p}_1$ 应用到由(i)所得权值矩阵处的线性联想器，然后应用到由(ii)所得权值矩阵的线性联想器。

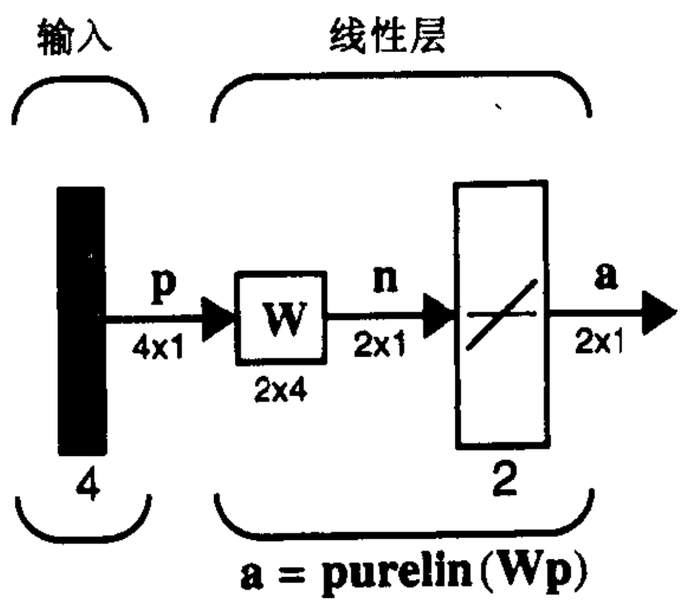



图 7-7 单神经元感知机

**解**

（i）第一步根据式(7.10)建立矩阵 **P** 和 **T**：

$$\mathbf{P}=\begin{bmatrix}1&1\\-1&1\\1&-1\\-1&-1\end{bmatrix},\quad \mathbf{T}=\begin{bmatrix}1&1\\-1&1\end{bmatrix}$$

7-16 然后使用式(7.9)求权值矩阵：

$$\mathbf{W}^h=\mathbf{T}\mathbf{P}^T=\begin{bmatrix}1&1\\-1&1\end{bmatrix}\begin{bmatrix}1&-1&1&-1\\1&1&-1&-1\end{bmatrix}=\begin{bmatrix}2&0&0&-2\\0&2&-2&0\end{bmatrix}$$

（ii）对仿逆规则使用式(7.32)：

$$\mathbf{W}=\mathbf{T}\mathbf{P}^+$$

由于 **P** 的行数为 4，大于其列数 2，且其列向量线性无关，则可用式(7.34)求仿逆：

$$\mathbf{P}^+=(\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T$$

$$\mathbf{P}^+=\left(\begin{bmatrix}1&-1&1&-1\\1&1&-1&-1\end{bmatrix}\begin{bmatrix}1&1\\-1&1\\1&-1\\-1&-1\end{bmatrix}\right)^{-1}\begin{bmatrix}1&-1&1&-1\\1&1&-1&-1\end{bmatrix}$$

$$=\begin{bmatrix}4&0\\0&4\end{bmatrix}^{-1}\begin{bmatrix}1&-1&1&-1\\1&1&-1&-1\end{bmatrix}$$

$$=\begin{bmatrix}\frac{1}{4}&0\\0&\frac{1}{4}\end{bmatrix}\begin{bmatrix}1&-1&1&-1\\1&1&-1&-1\end{bmatrix}=\begin{bmatrix}\frac{1}{4}&-\frac{1}{4}&\frac{1}{4}&-\frac{1}{4}\\\frac{1}{4}&\frac{1}{4}&-\frac{1}{4}&-\frac{1}{4}\end{bmatrix}$$

现可以计算权值矩阵：

$$\mathbf{W}^p = \mathbf{T}\mathbf{P}^+ = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & -\frac{1}{2} \\ 0 & \frac{1}{2} & -\frac{1}{2} & 0 \end{bmatrix}$$

（iii）现在测试两个权值矩阵：

$$\mathbf{W}^h\mathbf{p}_1 = \begin{bmatrix} 2 & 0 & 0 & -2 \\ 0 & 2 & -2 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 4 \\ -4 \end{bmatrix} \neq \mathbf{t}_1$$

7-17

$$\mathbf{W}^p\mathbf{p}_1 = \begin{bmatrix} \frac{1}{2} & 0 & 0 & -\frac{1}{2} \\ 0 & \frac{1}{2} & -\frac{1}{2} & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \mathbf{t}_1$$

为什么 Hebb 规则不能产生正确的结果呢？重新考虑式(7.11)。由于 $\mathbf{p}_1$ 和 $\mathbf{p}_2$ 正交(请对此进行验证)，这个等式可以写为

$$\mathbf{W}^h\mathbf{p}_1 = \mathbf{t}_1(\mathbf{p}_1^T\mathbf{p}_1)$$

但向量 $\mathbf{p}_1$ 没有规格化，故 $\mathbf{p}_1^T\mathbf{p}_1 \neq 1$，所以网络输出不会等于 $\mathbf{t}_1$。

另一方面，仿逆规则能够确保下式达到最小值：

$$\sum_{q=1}^{2} \| \mathbf{t}_p - \mathbf{W}\mathbf{p}_q \|^2$$

该式的值在本题中为零。

**P7.2** 考虑图 7-8 所示的原型模式。

（i）这些模式是否正交？

（ii）使用 Hebb 规则，为这些模式设计一个自联想存储器。

（iii）输入图 7-8 中的原型模式 $\mathbf{p}_t$，求网络响应。

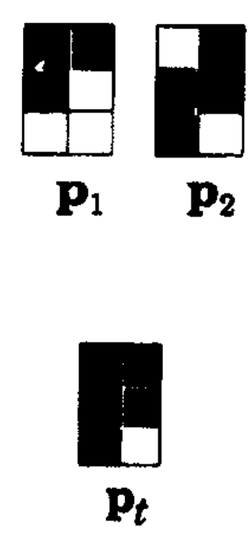


图 7-8

**解**

（i）首先将模式转换成向量。假设黑方格取值为 1，白方格取值为 -1。然后对模式进行逐列扫描，可以将这些二维模式转换成向量(也可以逐行扫描)。由此得到如下两个原型向量：

$$\mathbf{p}_1 = [1 \quad 1 \quad -1 \quad 1 \quad -1 \quad -1]^T, \quad \mathbf{p}_2 = [-1 \quad 1 \quad 1 \quad 1 \quad 1 \quad -1]^T$$

为了判断 $\mathbf{p}_1$ 和 $\mathbf{p}_2$ 是否正交，需要求它们的内积：

7-18

$$\mathbf{p}_1^T\mathbf{p}_2 = [1 \quad 1 \quad -1 \quad 1 \quad -1 \quad -1] \begin{bmatrix} -1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \end{bmatrix} = 0$$

故 $\mathbf{p}_1$ 和 $\mathbf{p}_2$ 是正交的。(由于 $\mathbf{p}_1^T\mathbf{p}_1 = \mathbf{p}_2^T\mathbf{p}_2 = 6$，所以 $\mathbf{p}_1$，$\mathbf{p}_2$ 都不是规格化向量。)

（ii）采用如图 7-3 所示的自联想存储器，这里的输入/输出的数量为 6。运用 Hebb 规则求权值矩阵：

$$\mathbf{W} = \mathbf{T}\mathbf{P}^T$$

其中

$$\mathbf{P} = \mathbf{T} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \\ -1 & 1 \\ 1 & 1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix}$$

所以权值矩阵为

$$\mathbf{W} = \mathbf{T}\mathbf{P}^T = \begin{bmatrix} 1 & -1 \\ 1 & 1 \\ -1 & 1 \\ 1 & 1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & -1 & 1 & -1 & -1 \\ -1 & 1 & 1 & 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & -2 & 0 & -2 & 0 \\ 0 & 2 & 0 & 2 & 0 & -2 \\ -2 & 0 & 2 & 0 & 2 & 0 \\ 0 & 2 & 0 & 2 & 0 & -2 \\ -2 & 0 & 2 & 0 & 2 & 0 \\ 0 & -2 & 0 & -2 & 0 & 2 \end{bmatrix}$$

（iii）为了把测试模式提交给该网络，需要将其转化为如下向量：

7-19

$$\mathbf{p}_t = [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad -1]^T$$

那么网络的响应为

$$\mathbf{a} = \mathbf{hardlims}(\mathbf{W}\mathbf{p}_t) = \mathbf{hardlims}\left(\begin{bmatrix} 2 & 0 & -2 & 0 & -2 & 0 \\ 0 & 2 & 0 & 2 & 0 & -2 \\ -2 & 0 & 2 & 0 & 2 & 0 \\ 0 & 2 & 0 & 2 & 0 & -2 \\ -2 & 0 & 2 & 0 & 2 & 0 \\ 0 & -2 & 0 & -2 & 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \end{bmatrix}\right)$$

$$\mathbf{a} = \mathbf{hardlims}\left(\begin{bmatrix} -2 \\ 6 \\ 2 \\ 6 \\ 2 \\ -6 \end{bmatrix}\right) = \begin{bmatrix} -1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \end{bmatrix} = \mathbf{p}_2$$

这是满意的网络响应吗？我们希望网络对这个输入模式如何响应？网络应该产生与输入模式最接近的原型模式。这里，测试输入模式 $\mathbf{p}_t$ 与 $\mathbf{p}_2$ 的 Hamming 距离为 1，与 $\mathbf{p}_1$ 的 Hamming 距离为 2。因此，该网络的确产生正确的响应(参见第 3 章关于 Hamming 距离的论述)。

请注意：本例中并未对原型向量进行规格化。但这并未导致发生在 P7.1 中相同的网络性能问题，原因在于 *hardlims* 的非线性特性使得网络输出只能取 1 或 -1。实际上，大多数神经网络非常有趣和有用的特性都归因于非线性特性的作用。

**P7.3** 考虑有三个原型模式(如下所示的 $\mathbf{p}_1$，$\mathbf{p}_2$，$\mathbf{p}_3$)的自联想问题。试分别运用 Hebb 规则和仿逆规则设计一个自联想网络，以识别这些模式。用下面的测试模式 $\mathbf{p}_t$ 检验网络的性能。 7-20

$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}, \mathbf{p}_3 = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \\ 1 \\ -1 \\ 1 \end{bmatrix}, \mathbf{p}_t = \begin{bmatrix} -1 \\ 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}$$

**解**

这个问题用手工求解有些枯燥，所以我们用 MATLAB 工具求解。

首先，建立原型向量：

```
p1 = [ 1  1  -1  -1  1   1  1]'
p2 = [ 1  1   1  -1  1   1  1]'
p3 = [-1  1  -1   1  1  -1  1]'
P = [p1  p2  p3]
```

现在用 Hebb 规则求权值矩阵：

```
wh = P*P'
```

为了测试网络，首先生成测试向量：

```
pt = [-1  1  -1  -1  1  -1  1]'
```

然后计算网络的响应：

```
ah = hardlims(wh*pt)
ah'
ans =
     1     1    -1    -1     1    -1     1
```

请注意：这个响应与任何原型向量都不匹配。这并不奇怪，因为原型模式并不是正交的。现在用仿逆规则再来计算。

```
pseu = inv(P'*P)*P'
wp = P*pseu
ap = hardlims(wp*pt)
ap'
ans =
    -1     1    -1     1     1    -1     1
```

7-21

请注意：该网络响应与 $\mathbf{p}_3$ 相等。这是正确的响应吗？通常希望响应为与输入模式最接近的原型模式。在本题中，$\mathbf{p}_t$ 与 $\mathbf{p}_1$，$\mathbf{p}_2$ 的 Hamming 距离均为 2，只有与 $\mathbf{p}_3$ 的 Hamming 距离为 1。因此，仿逆规则产生了正确的响应。

请用其他测试输入验证是否存在仿逆规则比 Hebb 规则产生更好结果的其他情况。

**P7.4** 考虑图 7-9 中的三个样本模式。

（i）用 Hebb 规则设计一个感知机网络识别这三个模式。

（ii）求该网络对图 7-9 中模式 $\mathbf{p}_t$ 的响应，并判断该响应是否正确。

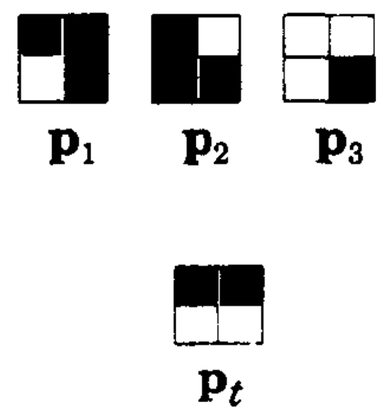



图 7-9

**解**

（i）按照前面例题的做法，将这些模式转换成如下向量：

$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix} \quad \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix} \quad \mathbf{p}_3 = \begin{bmatrix} -1 \\ -1 \\ -1 \\ 1 \end{bmatrix} \quad \mathbf{p}_t = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$$

现在需要选择每一原型输入向量的期望输出向量。由于有三个原型向量需要区分，所以输出向量需要两个元素。假设三个原型输入向量的期望输出分别为

$$\mathbf{t}_1 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \mathbf{t}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{t}_3 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

（请注意：这种选择是任意的，可以为每个向量设定 1 和 -1 的不同组合。）

7-22 所设计的感知机网络如图 7-10 所示。

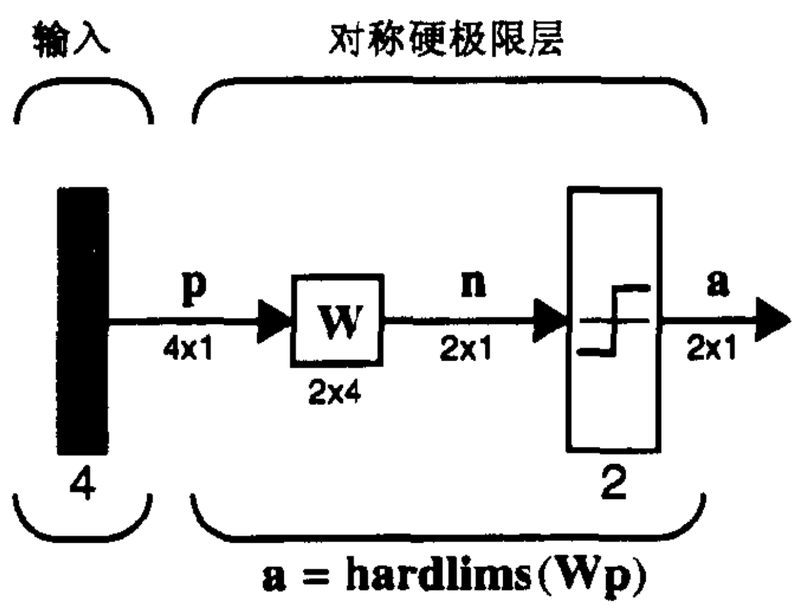



图 7-10 例题 P7.4 的感知机网络

然后用 Hebb 规则确定权值矩阵：

$$\mathbf{W} = \mathbf{TP}^T = \begin{bmatrix} -1 & -1 & 1 \\ -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ -1 & -1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} -3 & -1 & -1 & -1 \\ 1 & 3 & -1 & -1 \end{bmatrix}$$

（ii）相应于测试输入模式的网络响应为：

$$\mathbf{a} = \mathbf{hardlims}(\mathbf{WP}_t) = \mathbf{hardlims}\left( \begin{bmatrix} -3 & -1 & -1 & -1 \\ 1 & 3 & -1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} \right)$$

$$= \mathbf{hardlims}\left( \begin{bmatrix} -2 \\ -2 \end{bmatrix} \right) = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \rightarrow \mathbf{p}_1$$

网络响应表明测试输入模式与 $\mathbf{p}_1$ 最接近。这是正确的，因为到 $\mathbf{p}_1$ 的 Hamming 距离为 1，而到 $\mathbf{p}_2$ 和 $\mathbf{p}_3$ 的 Hamming 距离都是 3。

**P7.5** 假设针对 $Q$ 个长度为 $R$ 的正交样本向量，用 Hebb 规则设计了一个线性自联想存储器。向量元素为 1 或 -1。

（i）证明 $Q$ 个原型模式为权值矩阵的特征向量。 7-23

（ii）求出权值矩阵的另外($R-Q$)个特征向量。

**解**

（i）设原型向量为：

$$\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_Q$$

由于这是一个自联想存储器，这些向量是输入向量，也是期望输出向量。所以有

$$\mathbf{T} = \left[\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_Q\right] \qquad \mathbf{P} = \left[\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_Q\right]$$

如果用 Hebb 规则求权值矩阵，由式(7.8)可得

$$\mathbf{W} = \mathbf{T}\mathbf{P}^T = \sum_{q=1}^{Q} \mathbf{p}_q \mathbf{p}_q^T$$

现在，将一个原型向量作为网络输入，则有

$$\mathbf{a} = \mathbf{W}\mathbf{p}_k = \left(\sum_{q=1}^{Q} \mathbf{p}_q \mathbf{p}_q^T\right)\mathbf{p}_k = \sum_{q=1}^{Q} \mathbf{p}_q (\mathbf{p}_q^T \mathbf{p}_k)$$

因为这些模式正交，所以上式可简化为

$$\mathbf{a} = \mathbf{p}_k (\mathbf{p}_k^T \mathbf{p}_k)$$

又由于 $\mathbf{p}_k$ 的每个元素只能取 1 或 -1，求得

$$\mathbf{a} = \mathbf{p}_k R$$

综合以上结果：

$$\mathbf{W}\mathbf{p}_k = R\mathbf{p}_k$$

这表明：$\mathbf{p}_k$ 是 $\mathbf{W}$ 的特征向量，而 $R$ 则是相应的特征值。每个原型向量都是具有同一特征值的 $\mathbf{W}$ 的一个特征向量。

（ii）注意到多重特征值 $R$ 有一个与其相关的 $Q$ 维特征空间：由 $Q$ 个原型向量生成的子空间。现在考虑与特征空间正交的子空间。这个子空间内的每个向量都应与每个原型向量正交。正交子空间的维数为 $R-Q$。考虑这个正交空间的任意一个基集： 7-24

$$\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_{R-Q}$$

任取一个基向量作用于网络，可得

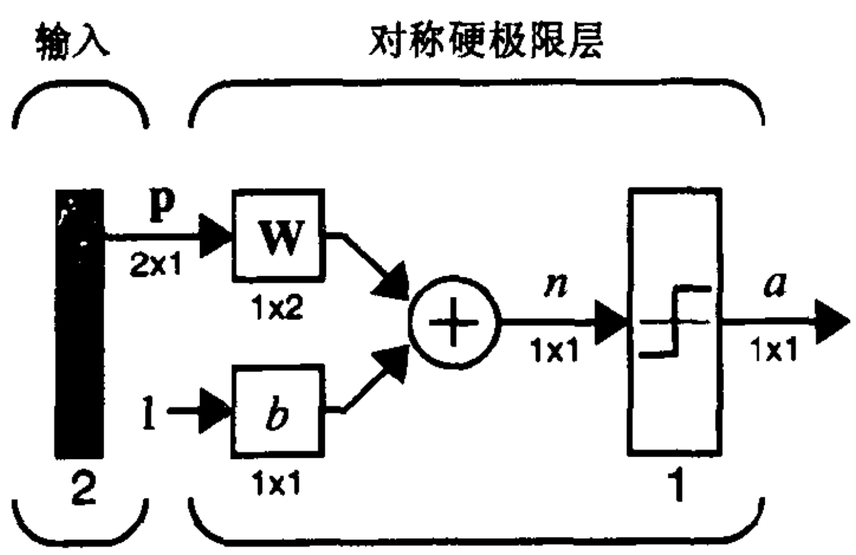
$$\mathbf{a} = \mathbf{W}\mathbf{z}_k = \left(\sum_{q=1}^{Q} \mathbf{p}_q \mathbf{p}_q^T\right)\mathbf{z}_k = \sum_{q=1}^{Q} \mathbf{p}_q (\mathbf{p}_q^T \mathbf{z}_k) = 0$$

由于每个 $\mathbf{z}_k$ 与每个 $\mathbf{p}_q$ 正交，这也说明 $\mathbf{z}_k$ 为 $\mathbf{W}$ 的以 0 为特征值的特征向量。

综上所述，权值矩阵有两个特征值 $R$ 和 0。也就是说，由原型向量生成的空间中的任意向量都将被扩大 $R$ 倍，而任何与原型向量正交的向量都将被置为零。在第 18 章讨论 Hopfield 网络的性能时，我们还会用到这个概念。

**P7.6** 本章迄今为止所使用的网络都不包含偏置向量。考虑设计能够识别下面模式的感知机网络(图 7-11)：

$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$




$$a = hardlims(\mathbf{Wp} + b)$$

图 7-11 单神经元感知机

（i）为什么求解这个问题需要偏置值？

（ii）用仿逆规则设计一个包含偏置值的网络求解此问题。

**解**

7-25 （i）在第 3 章和第 4 章中，感知机的判定边界是由下式定义的一条直线：

$$\mathbf{Wp} + b = 0$$

如果不存在偏置值，那么 $b = 0$，判定边界定义由

$$\mathbf{Wp} = 0$$

定义，必定是一条经过坐标原点的直线。现在考虑本题中给出的两个向量 $\mathbf{p}_1$ 和 $\mathbf{p}_2$。它们表示在图 7-12 中，图中还有一条经过坐标原点的任意判定边界。显然，任何穿过坐标原点的判定边界线都不可能将向量 $\mathbf{p}_1$ 和 $\mathbf{p}_2$ 分开。所以需要引入偏置值来求解本问题。

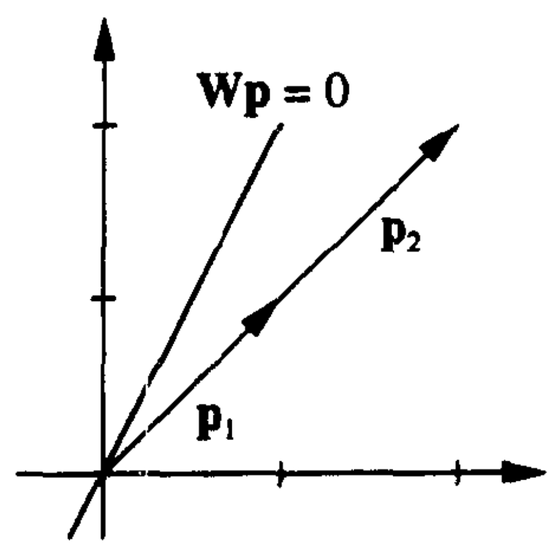



图 7-12 问题求解的图形表示

（ii）为了在存在偏置项时使用仿逆规则（或 Hebb 规则），应该将偏置值看成是输入为 1 的另外一个权值（如在所有网络图形中所示那样），然后对输入向量添加一个 1 作为最后元素：

$$\mathbf{p}'_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad \mathbf{p}'_2 = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}$$

假设这两个输入向量的期望输出分别为：

$$t_1 = 1, \quad t_2 = -1$$

所以

$$\mathbf{P} = \begin{bmatrix} 1 & 2 \\ 1 & 2 \\ 1 & 1 \end{bmatrix}, \quad \mathbf{T} = [1 \quad -1]$$

现在来构造仿逆矩阵：

$$\mathbf{P}^+ = \left( \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 1 & 2 \\ 1 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 5 \\ 5 & 9 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 1 \end{bmatrix} = \begin{bmatrix} -0.5 & -0.5 & 2 \\ 0.5 & 0.5 & -1 \end{bmatrix}$$

那么添加元素后的权值矩阵为：

$$\mathbf{W}' = \mathbf{TP}^{+} = [1 \quad -1]\begin{bmatrix} -0.5 & -0.5 & 2 \\ 0.5 & 0.5 & -1 \end{bmatrix} = [-1 \quad -1 \quad 3]$$

最后，可得标准权值矩阵和偏置值：

7-26

$$\mathbf{W} = [-1 \quad -1], \quad b = 3$$

由该权值矩阵和偏置值形成的判定边界如图 7-13 所示。这个边界将两个原型向量分离开了。

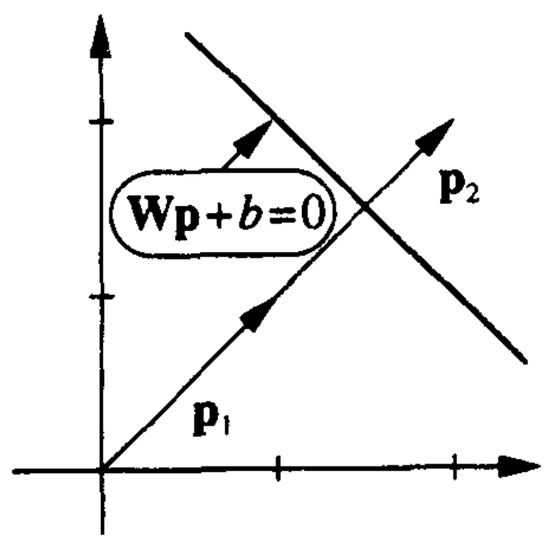



图 7-13　求解例题 P7.6 的判定边界

**P7.7** 迄今为止的所有模式识别的例子中都是用 1 和 - 1 的向量来表示模式，其中“1”代表图像元素的暗象素，“ - 1”代表亮象素。如果用“1”和“0”来表示，又将如何？Hebb 规则应作什么改变？

**解**

首先介绍一些用来区别这两种不同表示(通常指双极表示法{ - 1，1}和二进制表示法{0，1})的符号。原型输入/输出向量的双极表示法记为

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \cdots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

二进制表示法记为

$$\{\mathbf{p}'_1, \mathbf{t}'_1\}, \{\mathbf{p}'_2, \mathbf{t}'_2\}, \cdots, \{\mathbf{p}'_Q, \mathbf{t}'_Q\}$$

两种表示法的关系为：

$$\mathbf{p}'_q = \frac{1}{2}\mathbf{p}_q + \frac{1}{2}\mathbf{1}, \quad \mathbf{p}_q = 2\mathbf{p}'_q - \mathbf{1}$$

其中**1**为1的向量。

然后我们确定二进制联想网络的形式。这里使用如图 7-14 所示的网络，它与图 7-3 所示的双极联想网络的形式相比有两点不同。首先，它使用 *hardlim* 的非线性特性，而不是 *hardlims*，这样可以使其输出为 0 或 1。其次，它使用了偏置向量。需要偏置向量是因为所有的二进制向量都落在向量空间第 I 象限，故穿过坐标原点的边界线并不保证一定能够分离开所有的模式(参见例题 P7.6)。

7-27

下一步是确定该网络的权值矩阵和偏置向量。如果要使图 7-14 所示的二进制网络具有与图 7-3 所示双极网络具有同样的有效响应，必须使两个网络的净输入 **n** 相同：

$$\mathbf{W}'\mathbf{p}' + \mathbf{b} = \mathbf{W}\mathbf{p}$$

这样能够保证当双极网络产生“1”时二进制网络也产生“1”，当双极网络产生“ - 1”时，

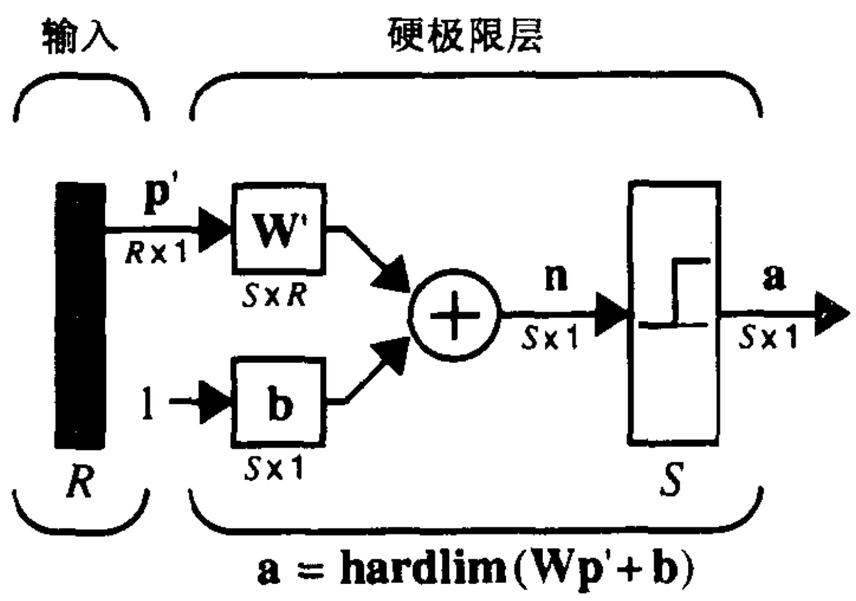



图 7-14 二进制联想网络

二进制网络产生“0”。

如果用 $\mathbf{p}$ 的一个函数代替 $\mathbf{p}'$，我们求得

$$\mathbf{W}'\left(\frac{1}{2}\mathbf{p}+\frac{1}{2}1\right)+\mathbf{b}=\frac{1}{2}\mathbf{W}'\mathbf{p}+\frac{1}{2}\mathbf{W}'1+\mathbf{b}=\mathbf{W}\mathbf{p}$$

所以，为了生成和双极网络相同的输出结果，应该选择

$$\mathbf{W}'=2\mathbf{W},\quad \mathbf{b}=-\mathbf{W}1$$

7-28 其中 $\mathbf{W}$ 为双极权值矩阵。

## 7.5 结束语

本章有两个主要目的。第一，介绍一个影响深远的神经网络学习规则：Hebb 规则。它是最早提出的神径网络学习规则之一，而且将继续影响最近所提出的一些神经网络学习理论。第二，如何用前两章所阐述的线性代数概念对该学习规则的性能进行诠释。这也是本书的主要目的之一。我们意在揭示某些重要的数学概念如何构成所有人工神经网络运行的基础。我们将继续使数学思想与神经网络应用紧密结合，进而使读者对二者的理解能够得以深化。

在第 13 章和第 18 章中还会用到 Hebb 规则。第 18 章将应用 Hebb 规则设计递归联想存储器网络——Hopfield 网络。

接下来的两章将介绍一些对理解第 10 章和第 11 章中的两个学习规则而言至关重要的数学知识。这些学习规则的属于性能学习一类，因为它们都是为了尽量使网络的性能得到优化。为了理解这些性能学习规则，需要引入一些基本的优化概念。这和学习 Hebb 规则一
7-29 样，前面的线性代数知识也将对理解这些优化问题的大有裨益。

## 参考文献

[Albe72] A. Albert, *Regression and the Moore – Penrose Pseudoinverse*, New York: Academic Press, 1972.

Albert 的著作是仿逆的基本特性及其理论的主要文献，同时还包括了所有主要的仿逆理论的证明。

[Ande72] J. Anderson, “A simple neural network generating an interactive memory,” *Mathematical Biosciences*, Vol. 14, pp. 197 – 220, 1972.

Anderson 提出了联想存储器的一种线性联想器模型。该模型使用一种推广的 Hebb 原理进行训练，学习在输入/输出向量之间建立关联关系。他主要强调了网络的生理学仿生特性。Kohonen 发表了一篇类似的论文[Koho72]，但是他们是各自独立完成了这项工作的。

[Hebb49] D. O. Hebb, The *Organization of Behavior*, New York: Wiley, 1949.

这本重要的著作主要论述了行为能由神经的活动来解释。在本书中，Hebb 提出了最早的学习规则之一，即一种在细胞级别上的学习机制。

[Koho72] T. Kohonen, "Correlation matrix memories," *IEEE Transactions on Computers*, Vol. 21, pp. 353 - 359, 1972.

Kohonen 提出了一种联想存储器的关联矩阵模型。该模型使用外积存储规则(也称为 Hebb 规则)。来学习输入/输出向量之间的关联关系，主要强调网络的数学结构。Aoderson 在同一时期发表了一篇类似的论文[Ande72]，但是他们是独立完成这项工作的。

7-30

## 习题

**E7.1** 请考察图 7-15 所示的样本模式。

(i) $\mathbf{p}_1$ 和 $\mathbf{p}_2$ 是否正交?

(ii) 请运用 Hebb 规则为这些模式设计一个自联想器网络。

(iii) 使用图 7-15 中所示的输入模式 $\mathbf{p}_t$ 来测试该网络的操作。网络能否达到预期的目标? 请给出相应的解释。

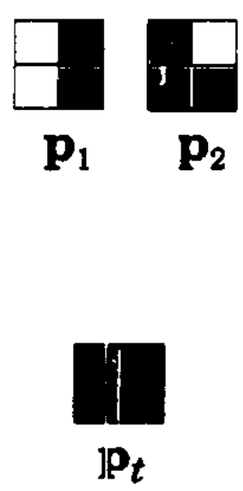



图 7-15

**E7.2** 请用仿逆规则求解 E7.1。

**E7.3** 试用 Hebb 规则确定如图 7-17 所示的感知机网络的权值矩阵，以识别图 7-16 所给出的模式。

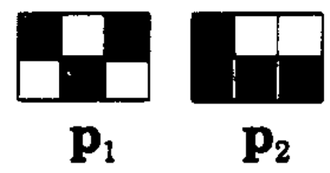



图 7-16　练习 E7.3 的感知机网络

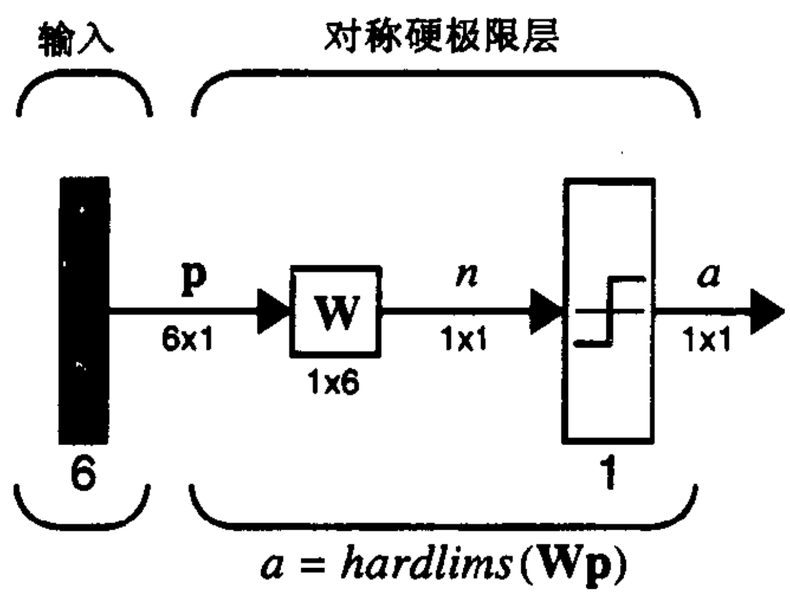



图 7-17

**E7.4** 在例题 P7.7 中阐述了当原型向量为二进制形式(同双极形式相反)时如何用 Hebb 规则训练网络。请用二进制形式表示原型向量求解 E7.1。说明这种二进制网络的响应与原来的双极网络的响应相等。

**E7.5** 试证明：如果将 Hebb 规则确定的自联想器的权值矩阵的对角线元素设置为 0，网

络仍然能够工作。也即权值矩阵由下式确定：

7-31

$$\mathbf{W} = \mathbf{P}\mathbf{P}^T - Q\mathbf{I}$$

其中 Q 为原型向量个数。(**提示**：证明原型向量仍为新的权值矩阵的特征向量。)

**E7.6** 有三个输入/输出原型向量对：

$$\left\{\mathbf{p}_1 = \begin{bmatrix}1\\0\end{bmatrix}, \mathbf{t}_1 = 1\right\}, \left\{\mathbf{p}_2 = \begin{bmatrix}1\\1\end{bmatrix}, \mathbf{t}_2 = -1\right\}, \left\{\mathbf{p}_3 = \begin{bmatrix}0\\1\end{bmatrix}, \mathbf{t}_3 = 1\right\}$$

(**i**) 请说明，除非网络使用一个偏置值，否则这个问题无法求解。

(**ii**) 请用仿逆规则设计一个处理这些原型向量的网络。证明网络可以正确地转换这些原型向量。

**E7.7** 考虑下列关于 Hebb 规则和仿逆规则的问题：一个权值矩阵能存储多少原型模式？请用 7.2.4 节讨论的数字识别问题实验来测试该问题。从数字“0”和“1”开始。一次加一个数字直到 6 为止，测试当随机改变 2，4 和 6 个象素时网络能够正确重构数字的次数。

(**i**) 首先用 Hebb 规则生成数字“0”和“1”的权值矩阵。然后随机地改变每个数字的 2 个象素，并将带噪声的数字输入到网络。重复此过程 10 次，记录网络输出端产生正确模式(无噪声数字)次数所占的百分比。改变每个数字的 4 个象素和 6 个象素，重复上述实验。然后用数字“0”、“1”和“2”，完全重复上述过程。实验一直进行下去，每次一个数字，直到用数字“0”到“6”的所有数字测试网络为止。完成全部测试后，就能画出表示重构错误与存储数字个数百分比的三条曲线，每条曲线分别对应于 2 个、4 个和 6 个象素错误。

7-32

(**ii**) 请用仿逆规则重复(i)，并比较两种规则的实验结果。

# 第8章　性能曲面和最优点

## 8.1　目的

本章介绍的是一类称为性能学习的神经网络训练的基础知识。神经网络有几种不同类型的学习规则，如联想学习(参见第7章的Hebb学习)和竞争学习(将在第14章中讨论)。性能学习是另一类重要的学习规则，其目的在于调整网络参数以优化网络性能。下面两章将介绍性能学习开发的背景知识，而性能学习的具体细节则将在第10章和第11章详细讨论。本章的主要目的是研究性能曲面，并确定性能曲面存在极大点和极小点的条件。第9章将继续讨论定位极大点和极小点的过程。 8-1

## 8.2　理论和实例

**性能学习**　有几种不同的学习规则可以归类于性能学习，本章将介绍其中两种，它们的区别在于训练网络时为优化网络性能而调整网络参数(权值和偏置值)的方法不同。

**性能指数**　这种优化过程分两个步骤进行。第一步是定义“性能”(performance)的含义。换言之，需要找到一个衡量网络性能的定量标准，即性能指数，性能指数在网络性能良好时很小，反之则很大。在本章以及第9章，我们都假设性能指数是已知的。第10章和第11章将讨论性能指数的选择方法。

优化过程的第二步是搜索减小性能指数的参数空间(调整网络权值和偏置值)。本章将研究性能曲面的特性，并建立确保极小点(即所寻求的最优点)存在的条件。所以在本章将了解性能曲面的一些概貌，第9章则将给出确定最优点的过程。

### 8.2.1　泰勒级数

**泰勒级数展开**　不妨把要最小化的性能指数用函数 $F(x)$ 表示，其中 $x$ 是要调整的参数。假定性能指数是一个解析函数，它的各级导数均存在。那么，$F(x)$ 可以表示成某些指定点 $x^*$ 上的*泰勒级数展开*：

$$
\begin{aligned}
F(x) = F(x^*) &+ \frac{\mathrm{d}}{\mathrm{d}x}F(x)\bigg|_{x=x^*}(x-x^*) \\
&+ \frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}x^2}F(x)\bigg|_{x=x^*}(x-x^*)^2 + \cdots \\
&+ \frac{1}{n!}\frac{\mathrm{d}^n}{\mathrm{d}x^n}F(x)\bigg|_{x=x^*}(x-x^*)^n + \cdots
\end{aligned}
\tag{8.1}
$$

通过限定泰勒级数展开项的数量，可以用泰勒级数近似估计性能指数。例如设 8-2

$$F(x) = \cos(x) \tag{8.2}$$

$F(x)$ 在 $x^*=0$ 点的泰勒级数展开式为

$$F(x) = \cos(x) = \cos(0) - \sin(0)(x-0) - \frac{1}{2}\cos(0)(x-0)^2$$

$$+\frac{1}{6}\sin(0)(x-0)^3+\cdots$$
$$=1-\frac{1}{2}x^2+\frac{1}{24}x^4+\cdots \tag{8.3}$$

$F(x)$的0阶近似(仅含 $x$ 的0次方项)是

$$F(x)\approx F_0(x)=1 \tag{8.4}$$

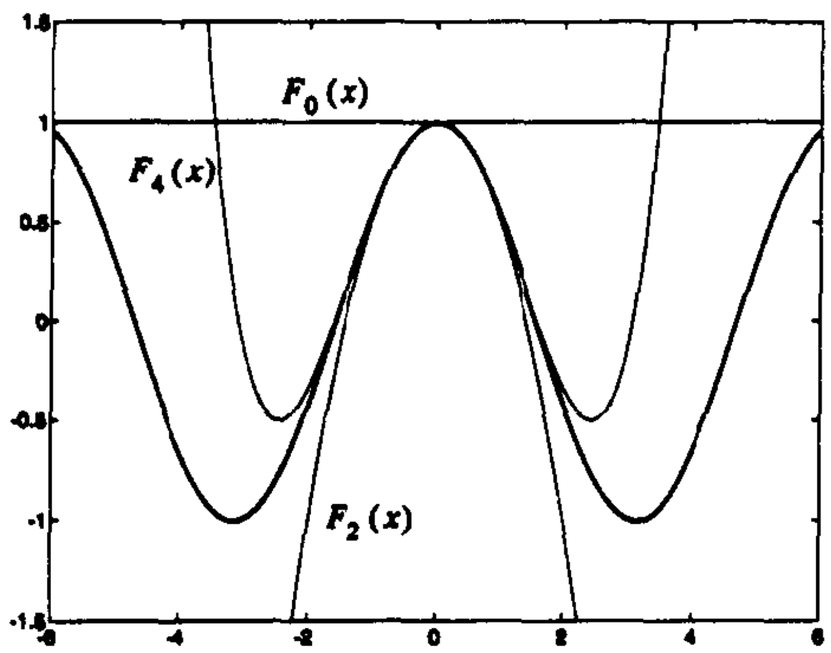
其2阶近似为

$$F(x)\approx F_2(x)=1-\frac{1}{2}x^2 \tag{8.5}$$

(注意此式的0阶近似与1阶近似相等，因为1阶导数为0。)其4阶近似为

$$F(x)\approx F_4(x)=1-\frac{1}{2}x^2+\frac{1}{24}x^4 \tag{8.6}$$

图8-1所示为 $F(x)$及其三个近似的图形。




8-3

图8-1 余弦函数及其泰勒级数近似

从图中可以看出，如果 $x$ 趋近于 $x^*=0$，所有的近似都是精确的。$x$ 离 $x^*$ 越远，则只有高阶近似是精确的。2阶近似比0阶近似的精度范围更大，4阶近似的精度范围又大于2阶近似的精度范围。式(8.1)可以说明这种现象。级数中每个相邻的后继项都包含$(x-x^*)$的高次项，$x$ 越趋近于 $x^*$，这些项将按几何级数减小。

我们将运用这种性能指数的泰勒级数近似方法，研究可能的最优点的邻域内性能指数的特性。

*试验余弦函数泰勒级数展开请用 Neural Network Design Demonstration Taylor Series* (**nnd8ts**)。

**向量的情况**

神经网络的性能指数并不仅是一个纯量 $x$ 的函数，它是所有网络参数(各个权值和偏置值)的函数，参数的数量可能是很大的。因此，需要将泰勒级数展开形式扩展为多变量形式。考虑下列 $n$ 元函数。

$$F(\mathbf{x})=F(x_1,x_2,\cdots,x_n) \tag{8.7}$$

这个函数在点 $\mathbf{x}^*$ 的泰勒级数展开为

$$F(\mathbf{x}) = F(\mathbf{x}^*) + \frac{\partial}{\partial x_1}F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*}(x_1 - x_1^*) + \frac{\partial}{\partial x_2}F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*}(x_2 - x_2^*)$$
$$+ \cdots + \frac{\partial}{\partial x_n}F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*}(x_n - x_n^*) + \frac{1}{2}\frac{\partial^2}{\partial x_1^2}F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*}(x_1 - x_1^*)^2 \quad (8.8)$$
$$+ \frac{1}{2}\frac{\partial^2}{\partial x_1 \partial x_2}F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*}(x_1 - x_1^*)(x_2 - x_2^*) + \cdots$$

**梯度　赫森矩阵**　这个表达式有些繁杂，把它写成矩阵形式会清晰些：

$$F(\mathbf{x}) = F(\mathbf{x}^*) + \nabla F(\mathbf{x})^T\Big|_{\mathbf{x}=\mathbf{x}^*}(\mathbf{x} - \mathbf{x}^*)$$
$$+ \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^T \nabla^2 F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*}(\mathbf{x} - \mathbf{x}^*) + \cdots \quad (8.9)$$

这里$\nabla F(\mathbf{x})$为梯度，其定义为　8-4

$$\nabla F(\mathbf{x}) = \left[\frac{\partial}{\partial x_1}F(\mathbf{x})\ \frac{\partial}{\partial x_2}F(\mathbf{x})\ \cdots\ \frac{\partial}{\partial x_n}F(\mathbf{x})\right]^T \quad (8.10)$$

$\nabla^2 F(\mathbf{x})$为赫森矩阵，定义为

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2}{\partial x_1^2}F(\mathbf{x}) & \frac{\partial^2}{\partial x_1 \partial x_2}F(\mathbf{x}) & \cdots & \frac{\partial^2}{\partial x_1 \partial x_n}F(\mathbf{x}) \\ \frac{\partial^2}{\partial x_2 \partial x_1}F(\mathbf{x}) & \frac{\partial^2}{\partial x_2^2}F(\mathbf{x}) & \cdots & \frac{\partial^2}{\partial x_2 \partial x_n}F(\mathbf{x}) \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2}{\partial x_n \partial x_1}F(\mathbf{x}) & \frac{\partial^2}{\partial x_n \partial x_2}F(\mathbf{x}) & \cdots & \frac{\partial^2}{\partial x_n^2}F(\mathbf{x}) \end{bmatrix} \quad (8.11)$$

梯度和赫森矩阵对于我们理解性能曲面非常重要。下节将讨论这两个概念的实际意义。

*试验二元函数的泰勒级数展开请用 Neural Network Design Demonstration* ***Vector Taylor Series*** **(nnd8ts2)**。

### 8.2.2　方向导数

梯度的第 $i$ 个元素$\partial F(\mathbf{x})/\partial x_i$，是性能指数 $F$ 在 $x_i$ 轴的一阶导数。赫森矩阵的第 $i$ 个对角元素$\partial^2 F(\mathbf{x})/\partial x_i^2$ 是性能指数 $F$ 沿 $x_i$ 轴的二阶导数。怎样求函数在任意方向上的一阶导数?

**方向导数**　设 $\mathbf{p}$ 为沿所求导数方向上的一个向量，此方向导数可由下式求出：

$$\frac{\mathbf{p}^T \nabla F(\mathbf{x})}{\|\mathbf{p}\|} \quad (8.12)$$

沿 $\mathbf{p}$ 的二阶导数也可以写成

$$\frac{\mathbf{p}^T \nabla^2 F(\mathbf{x})\mathbf{p}}{\|\mathbf{p}\|^2} \quad (8.13)$$　8-5

为了说明以上概念，考虑函数

$$F(\mathbf{x}) = x_1^2 + 2x_2^2 \quad (8.14)$$

假设求沿向量 $\mathbf{p}=[2 \quad -1]^T$ 的方向在点 $\mathbf{x}^*=[0.5 \quad 0.5]^T$ 处的导数。首先求在 $\mathbf{x}^*$ 的梯度：

$$\nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*}=\begin{bmatrix}\frac{\partial}{\partial x_1}F(\mathbf{x})\\ \frac{\partial}{\partial x_2}F(\mathbf{x})\end{bmatrix}\Bigg|_{\mathbf{x}=\mathbf{x}^*}=\begin{bmatrix}2x_2\\4x_2\end{bmatrix}\Bigg|_{\mathbf{x}=\mathbf{x}^*}=\begin{bmatrix}1\\2\end{bmatrix} \tag{8.15}$$

沿方向 $\mathbf{p}$ 的导数也可求出：

$$\frac{\mathbf{p}^T\nabla F(\mathbf{x})}{\|\mathbf{p}\|}=\frac{[2 \quad -1]\begin{bmatrix}1\\2\end{bmatrix}}{\left\|\begin{bmatrix}2\\-1\end{bmatrix}\right\|}=\frac{[0]}{\sqrt{5}}=0 \tag{8.16}$$

因此函数经过点 $\mathbf{x}^*$ 在 $\mathbf{p}$ 方向上的斜率为零。为什么会是这个结果呢？如何解释这种现象？考察式(8.12)关于方向导数的定义就能发现其分子部分是方向向量与梯度的内积。因此，任何与梯度正交的方向上的斜率都为零。

最大斜率在什么方向上？当方向向量与梯度的内积最大时斜率最大，故当方向向量与梯度同向时会出现最大斜率(注意方向向量的长度对此没有影响，因为它已被规格化。)这种情况在图 8-2 的 $F(\mathbf{x})$的平面轮廓线图和 3-D 图中表露无遗。在轮廓图中，从某个点 $\mathbf{x}^*$ 出发的 5 个向量方向各异，各个向量的一阶方向导数也已标示出来。沿梯度方向的导数最大，而与梯度正交的方向上的导数为零(与轮廓线相切)。

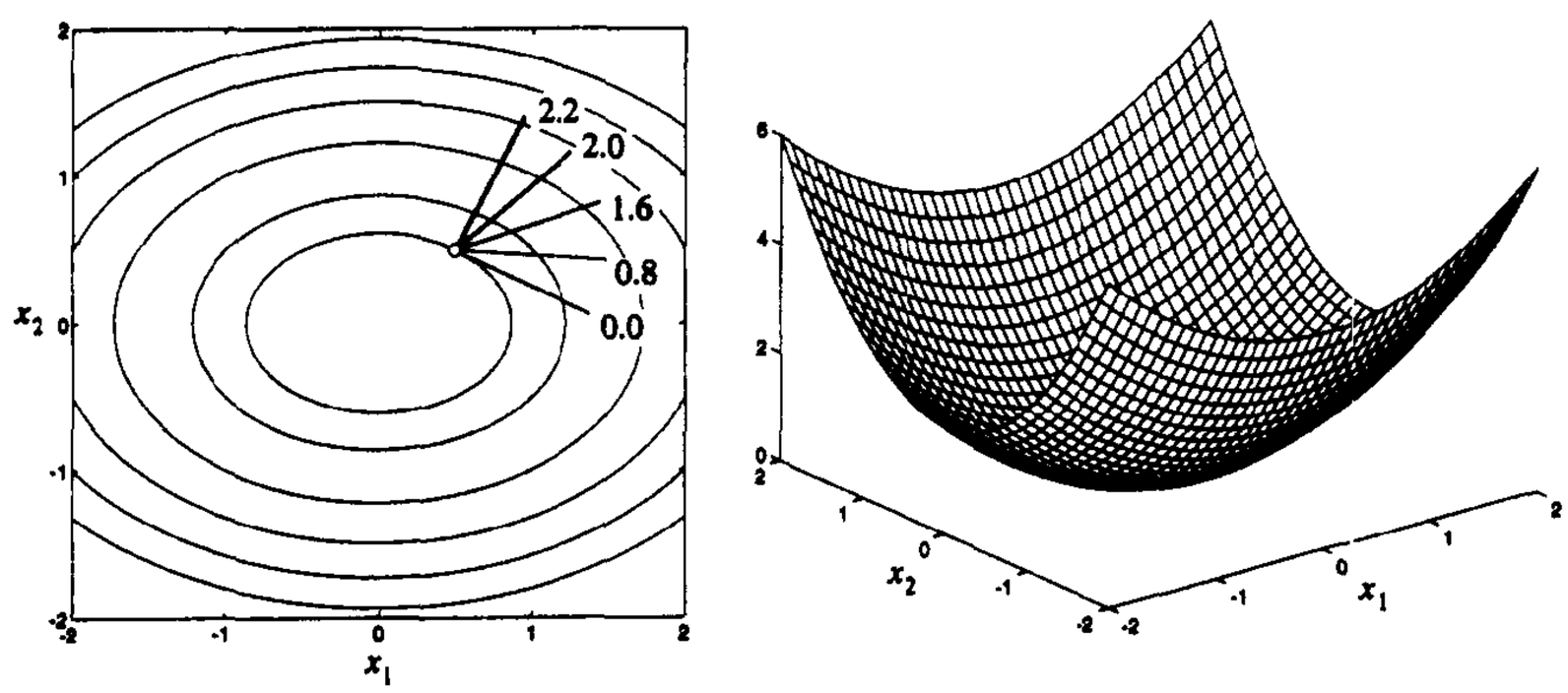



图 8-2 二次函数及其方向导数

8-6 试验方向导数请用 *Neural Network Design Demonstration Directional Derivatives* (**nnd8dd**)。

### 8.2.3 极小点

回忆一下，性能学习的目的是使性能指数得到优化。本节将定义最优点的涵义。设性能指数的极小点即最优点。对于最大化问题很容易修改此定义。

**强极小点** 称点 $\mathbf{x}^*$ 为 $F(\mathbf{x})$强极小点，如果存在某个纯量 $\delta>0$，使得当 $\delta>\|\Delta\mathbf{x}\|>0$ 时，对所有 $\Delta\mathbf{x}$ 都有 $F(\mathbf{x}^*)<F(\mathbf{x}^*+\Delta\mathbf{x})$成立。

换句话说，从一个强极小点出发沿任意方向移动任意一个小的距离都将使 $F(\mathbf{x})$增大。

**全局极小点**　称点 $\mathbf{x}^*$ 为 $F(\mathbf{x})$的惟一的全局极小点，如果 $F(\mathbf{x}^*) < F(\mathbf{x}^* + \Delta\mathbf{x})$对所有 $\Delta\mathbf{x} \neq 0$ 都成立。

对于一个强极小点 $\mathbf{x}^*$，在 $\mathbf{x}^*$ 较小的邻域之外可能会存在比 $F(\mathbf{x}^*)$更小的点，故 $\mathbf{x}^*$ 又称为局部极小点。对于一个全局极小点，$F(\mathbf{x})$在参数空间内任何其他点的值都比 $F(\mathbf{x}^*)$大。

**弱极小点**　称点 $\mathbf{x}^*$ 为 $F(\mathbf{x})$的弱极小点，如果它不是一个强极小点，且存在某个纯量 $\delta > 0$，使得对于所有 $\delta > \|\Delta\mathbf{x}\| > 0$ 的 $\Delta\mathbf{x}$ 都有 $F(\mathbf{x}^*) \leq F(\mathbf{x}^* + \Delta\mathbf{x})$成立。 8-7

从一个弱极小点无论向什么方向移动，函数值不会减少，但可能沿某些方向的值不变。

例如，考虑下列纯量函数：

$$F(x) = 3x^4 - 7x^2 - \frac{1}{2}x + 6 \tag{8.17}$$

图 8-3 所示为该函数图象。注意：大约在 $-1.1$ 和 $1.1$ 处有两个强极小点，函数在这两点的局部邻域内都增大。$1.1$ 处的点是全局最小点，因为没有其他点的函数值比该点的函数值更小。

这个函数不存在弱极小点。为了说明弱极小点我们在下面举一个二维的例子。

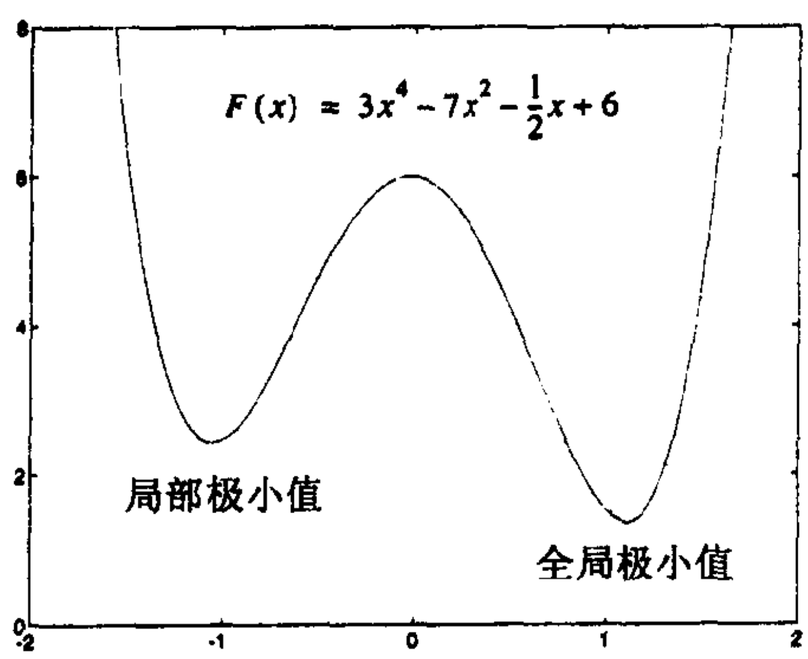



图 8-3　局部极小点和全局极小点举例

现在考虑向量情形。首先考虑下列函数：

$$F(\mathbf{x}) = (x_2 - x_1)^4 + 8x_1x_1 - x_1 + x_2 + 3 \tag{8.18}$$

**轮廓线图**　图 8-4 为该函数的轮廓线图(函数值为常数时的一系列曲线)和 3-D 曲面图(函数值小于 12)。可以看出函数有两个强局部极小点，一个在$(-0.42, 0.42)$，一个在$(0.55, -0.55)$。全局极小点为$(0.55, -0.55)$。

**鞍点**　该函数在$(-0.13, 0.13)$的点有其他有趣的特点。由于在该点邻域内曲面的形状，它称为一个鞍点(saddle point)。它的特点在于：沿线 $x_1 = -x_2$ 该鞍点为一个局部极大点，但沿一条与此线垂直的线它又是局部极小点。在例题 P8.2 和 P8.5 中我们将详细讨论这种情况。 8-8

在 *Neural Network Design Demonstration Vector taylor Series* (**nnd8ts2**)中也使用了这个函数。

作为最后一个例子，考虑下面的函数：

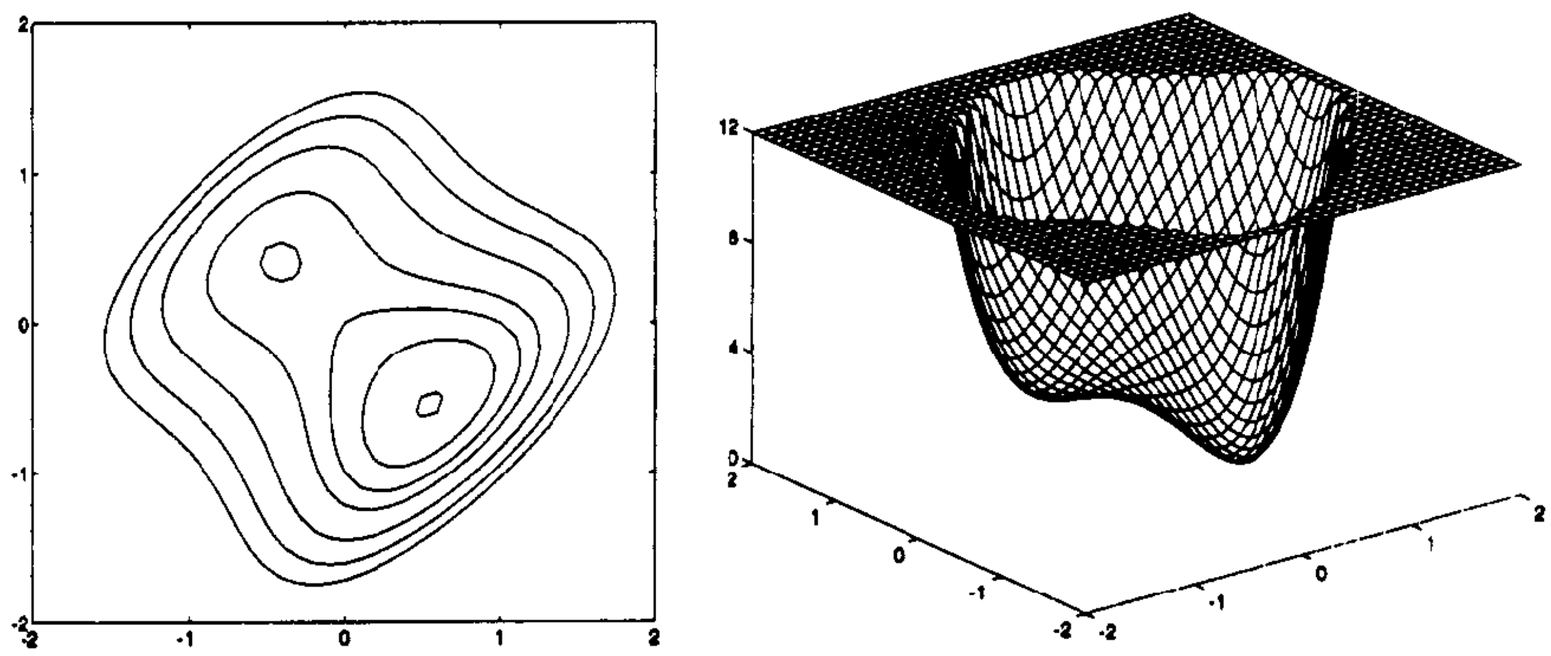


图 8-4 极小点和鞍点的向量实例

$$F(\mathbf{x}) = (x_1^2 - 1.5x_1x_2 + 2x_2^2)x_1^2 \tag{8.19}$$

图 8-5 给出了这个函数的轮廓线图和 3-D 图。可以看出沿 $x_1 = 0$ 的任意点都是弱极小点。

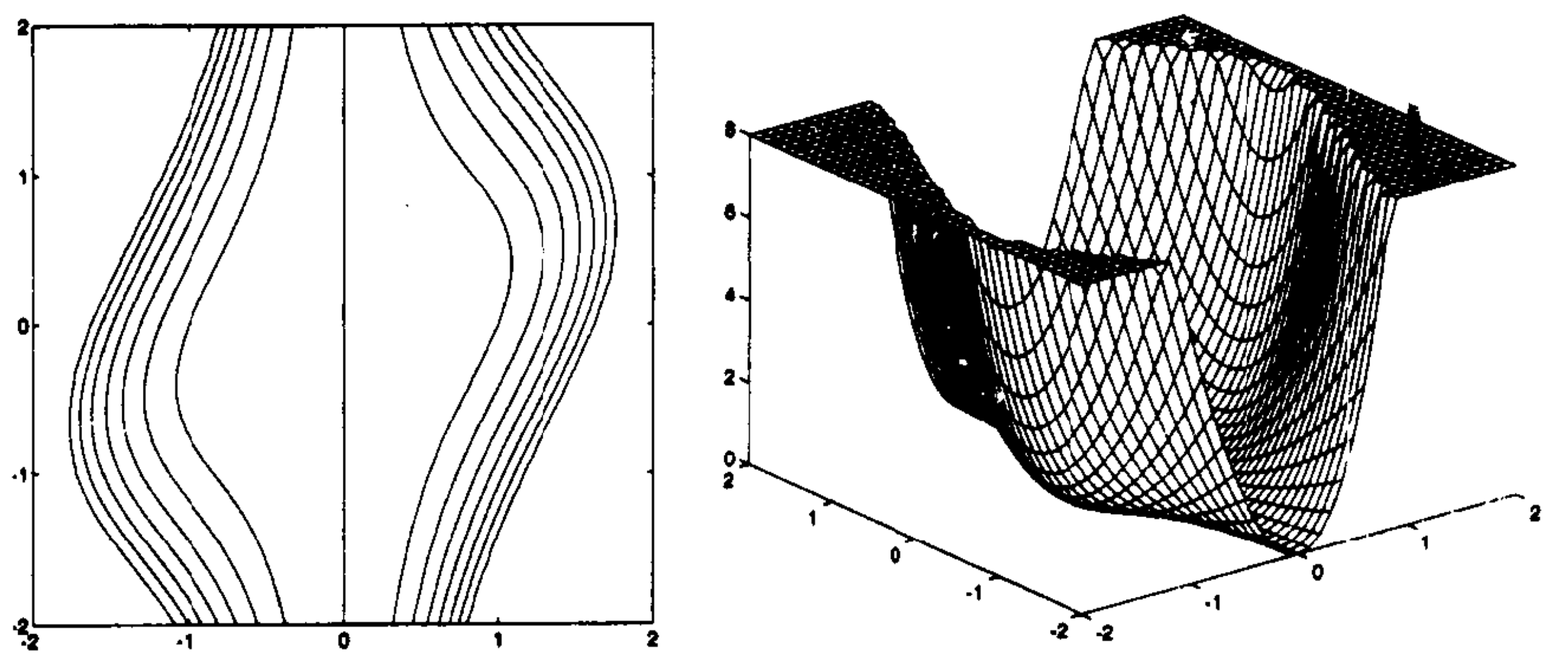


图 8-5 弱极小点实例

### 8.2.4 优化的必要条件

定义了最优点(极小点)后，必须给出这种点需要满足的条件。这里还要用到泰勒级数展开来推导这些条件：

8-9

$$\begin{aligned} F(\mathbf{x}) = F(\mathbf{x}^* + \Delta\mathbf{x}) &= F(\mathbf{x}^*) + \nabla F(\mathbf{x})^T\Big|_{\mathbf{x}=\mathbf{x}^*}\Delta\mathbf{x} \\ &+ \frac{1}{2}\Delta\mathbf{x}^T\nabla^2 F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*}\Delta\mathbf{x} + \cdots \end{aligned} \tag{8.20}$$

此处

$$\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}^* \tag{8.21}$$

1. 一阶条件

如果 $\|\Delta\mathbf{x}\|$ 很小，则式(8.20)中的高阶项可以省略，有 $F(\mathbf{x})$的近似表达式

$$F(\mathbf{x}^* + \Delta\mathbf{x}) \cong F(\mathbf{x}^*) + \nabla F(\mathbf{x})^T\Big|_{\mathbf{x}=\mathbf{x}^*}\Delta\mathbf{x} \tag{8.22}$$

要使 $\mathbf{x}^*$ 为极小点，则要使函数在 $\Delta\mathbf{x} \neq 0$ 时增大或不减小。要实现这个目标，则式

(8.22)中的第二项不能为负，即

$$\nabla F(\mathbf{x})^T\Big|_{\mathbf{x}=\mathbf{x}^*}\Delta\mathbf{x} \geqslant 0 \tag{8.23}$$

但是，如果这一项为正，即

$$\nabla F(\mathbf{x})^T\Big|_{\mathbf{x}=\mathbf{x}^*}\Delta\mathbf{x} > 0 \tag{8.24}$$

则可推导出：

$$F(\mathbf{x}^* - \Delta\mathbf{x}) \cong F(\mathbf{x}^*) - \nabla F(\mathbf{x})^T\Big|_{\mathbf{x}=\mathbf{x}^*}\Delta\mathbf{x} < F(\mathbf{x}^*) \tag{8.25}$$

这将导致自相矛盾，因为 $\mathbf{x}^*$ 为一个极小点。所以要使式(8.23)成立，式(8.24)就不能成立，惟一选择只有

$$\nabla F(\mathbf{x})^T\Big|_{\mathbf{x}=\mathbf{x}^*}\Delta\mathbf{x} = 0 \tag{8.26}$$

该式对所有的 $\Delta\mathbf{x}$ 都必须成立，即

$$\nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*} = \mathbf{0} \tag{8.27}$$

**驻点**　所以，一个极小点处的梯度一定为零。这就是局部极小点的一阶必要条件(不是充分条件)。所有满足式(8.27)的点称为驻点(stationary point)。 8-10

2. 二阶条件

设有一个驻点 $\mathbf{x}^*$。由于 $F(\mathbf{x})$在驻点的梯度为 0，则泰勒级数展式为

$$F(\mathbf{x}^* + \Delta\mathbf{x}) = F(\mathbf{x}^*) + \frac{1}{2}\Delta\mathbf{x}^T\nabla^2 F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*}\Delta\mathbf{x} + \cdots \tag{8.28}$$

同前面一样，这里只考虑那些在 $\mathbf{x}^*$ 的很小的邻域内的点，以使 $\|\Delta\mathbf{x}\|$ 很小且 $F(\mathbf{x})$能用式(8.28)的前两项近似。所以，如果

$$\Delta\mathbf{x}^T\nabla^2 F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*}\Delta\mathbf{x} > 0 \tag{8.29}$$

则在 $\mathbf{x}^*$ 将存在强极小点。

**正定矩阵　半正定矩阵**　要使此式对任意 $\Delta\mathbf{x}\neq 0$ 成立，赫森矩阵必须为正定矩阵。

(根据定义，一个正定矩阵定义为：对任意的向量 $\mathbf{z}\neq 0$ 有

$$\mathbf{z}^T\mathbf{A}\mathbf{z} > 0 \tag{8.30}$$

如果对任意向量 $\mathbf{z}$，有

$$\mathbf{z}^T\mathbf{A}\mathbf{z} \leqslant 0 \tag{8.31}$$

则称 $\mathbf{A}$ 为半正定矩阵。可以通过检验矩阵的特征值来检验这些条件。如果所有特征值为正，则矩阵为正定矩阵；如果所有特征值非负，则矩阵为半正定矩阵。)

**充分条件**　一个正定的赫森矩阵是一个强极小点存在的二阶充分条件，但不是必要条件。如果泰勒级数的二阶项为零，但三阶项为正，仍可能存在强极小点。所以强极小点存在的二阶必要条件是赫森矩阵为半正定矩阵。

为了说明这些条件，考虑下列二元函数：

$$F(\mathbf{x}) = x_1^4 + x_2^2 \tag{8.32}$$

为了求出驻点，先求梯度

8-11

$$\nabla F(\mathbf{x}) = \begin{bmatrix} 4x_1^3 \\ 2x_2 \end{bmatrix} = \mathbf{0} \tag{8.33}$$

故只有惟一驻点 $\mathbf{x}^* = 0$。现在来求二阶条件，赫森矩阵为

$$\nabla^2 F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{0}} = \begin{bmatrix} 12x_1^2 & 0 \\ 0 & 2 \end{bmatrix}_{\mathbf{x}=\mathbf{0}} = \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix} \tag{8.34}$$

这是一个半正定矩阵，即 $\mathbf{x}^* = \mathbf{0}$为强极小点的必要条件存在。这里无法从一阶和二阶条件确定该点为一个极小点，但这种可能性是存在的。事实上，尽管这里赫森矩阵是半正定的，但 $\mathbf{x}^* = \mathbf{0}$仍是一个强极小点，只是目前无法从已讨论的条件证明。

综上所述，$\mathbf{x}^*$ 为 $F(\mathbf{x})$的强极小点或弱极小点的必要条件是：

$$\nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*} = \mathbf{0} \text{ 和} \nabla^2 F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*} \text{ 为半正定}$$

$\mathbf{x}^*$ 为 $F(\mathbf{x})$的强极小点的充分条件是：

$$\nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*} = \mathbf{0} \text{ 和} \nabla^2 F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*} \text{ 为正定}$$

### 8.2.5 二次函数

本节介绍一种通用的性能指数——二次函数。这不仅因为二次函数应用广泛，而且还因为在很小的邻域内，特别是在局部极小的附近，许多函数可由二次函数来近似。所以有必要花一些时间来考察二次函数的特性。

**二次函数** 二次函数的一般形式是

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{d}^T\mathbf{x} + c \tag{8.35}$$

这里 $\mathbf{A}$ 为对称矩阵。(如果矩阵非对称，则可由产生同一 $F(\mathbf{x})$的另一对称矩阵置换。可试一下!)

求该函数的梯度，需用到下列梯度的性质：

$$\nabla(\mathbf{h}^T\mathbf{x}) = \nabla(\mathbf{x}^T\mathbf{h}) = \mathbf{h} \tag{8.36}$$

8-12 此处 $\mathbf{h}$ 为一常数向量，且

$$\nabla\mathbf{x}^T\mathbf{Q}\mathbf{x} = \mathbf{Q}\mathbf{x} + \mathbf{Q}^T\mathbf{x} = 2\mathbf{Q}\mathbf{x} \quad (\mathbf{Q}\text{ 为对称矩阵}) \tag{8.37}$$

现在可以计算 $F(\mathbf{x})$的梯度：

$$\nabla F(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{d} \tag{8.38}$$

同理可求赫森矩阵：

$$\nabla^2 F(\mathbf{x}) = \mathbf{A} \tag{8.39}$$

二次函数的所有的高阶导数为零，所以该函数的泰勒级数展开的前三项即该函数的精确表达(见式(8.20))。也可以说所有的解析函数在一个很小的邻域内(即当 $\|\Delta\mathbf{x}\|$ 很小时)都与二次函数类似。

**赫森的特征系统**

现在研究二次函数的一般形态。研究赫森矩阵的特征值和特征向量可以得到二次函数的许多性质。考虑以原点为驻点且其值为 0 的二次函数：

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} \tag{8.40}$$

如果把基进行适当的变换，会使函数的形态更加清楚(见第 6 章)。用赫森矩阵 **A** 的特征向量作为新的基向量。由于 **A** 为对称矩阵，所以其特征向量两两正交(见[Brog91])。所以可用特征向量作为列向量构成一个和式(6.68)一样的矩阵：

$$\mathbf{B} = [\mathbf{z}_1 \quad \mathbf{z}_2 \quad \cdots \quad \mathbf{z}_n] \tag{8.41}$$

该矩阵的逆等于其转置矩阵：

$$\mathbf{B}^{-1} = \mathbf{B}^T \tag{8.42}$$

(假定特征向量已被规格化。)

进行基变换，以使特征向量成为基向量(见式(6.69))，新的矩阵 **A** 为

$$\mathbf{A}' = [\mathbf{B}^T\mathbf{A}\mathbf{B}] = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} = \Lambda \tag{8.43}$$

8-13

其中 $\lambda_i$ 为 **A** 的特征值。上式也可写成

$$\mathbf{A} = \mathbf{B}\Lambda\mathbf{B}^T \tag{8.44}$$

我们将用方向导数的概念说明 **A** 的特征值和特征向量的物理意义以及如何确定二次函数的曲面特性。

由式(8.13)知 $F(\mathbf{x})$在向量 **p** 方向上的二阶导数为

$$\frac{\mathbf{p}^T\nabla^2 F(\mathbf{x})\mathbf{p}}{\|\mathbf{p}\|^2} = \frac{\mathbf{p}^T\mathbf{A}\mathbf{p}}{\|\mathbf{p}\|^2} \tag{8.45}$$

现在定义

$$\mathbf{p} = \mathbf{B}\mathbf{c} \tag{8.46}$$

这里 **c** 表示基于 **A** 的特征向量的向量 **p**(见式(6.28)及其后的讨论)。用这些概念及式(8.44)，可将式(8.45)重写成

$$\frac{\mathbf{p}^T\mathbf{A}\mathbf{p}}{\|\mathbf{p}\|^2} = \frac{\mathbf{c}^T\mathbf{B}^T(\mathbf{B}\Lambda\mathbf{B}^T)\mathbf{B}\mathbf{c}}{\mathbf{c}^T\mathbf{B}^T\mathbf{B}\mathbf{c}} = \frac{\mathbf{c}^T\Lambda\mathbf{c}}{\mathbf{c}^T\mathbf{c}} = \frac{\sum_{i=1}^{n}\lambda_i c_i^2}{\sum_{i=1}^{n}c_i^2} \tag{8.47}$$

这个结果包含若干有用的事实。首先，这个二阶导数是特征值的加权平均。所以它总不大于最大的特征值，或不小于最小特征值。换句话说，

$$\lambda_{min} \leqslant \frac{\mathbf{p}^T\mathbf{A}\mathbf{p}}{\|\mathbf{p}\|^2} \leqslant \lambda_{max} \tag{8.48}$$

二阶导数在什么条件下与最大特征值相等？如果选择

$$\mathbf{p} = \mathbf{z}_{max} \tag{8.49}$$

这里 $\mathbf{z}_{max}$是最大特征值 $\lambda_{max}$的特征向量，其结果如何？此时向量 **c** 为

$$\mathbf{c} = \mathbf{B}^T\mathbf{p} = \mathbf{B}^T\mathbf{z}_{max} = [0 \quad 0 \quad \cdots \quad 0 \quad 1 \quad 0 \quad \cdots \quad 0]^T \tag{8.50}$$

8-14

**c** 仅在与最大特征值(例如，$\mathbf{c}_{max} = 1$)相应的位置存在，因为特征向量是正交的。

用 $\mathbf{z}_{max}$ 代替式(8.47)中的 $\mathbf{p}$，则有

$$\frac{\mathbf{z}_{max}{}^T\mathbf{A}\mathbf{z}_{max}}{\|\mathbf{z}_{max}\|^2} = \frac{\sum_{i=1}^{n}\lambda_i c_i^2}{\sum_{i=1}^{n} c_i^2} = \lambda_{max} \tag{8.51}$$

所以，在最大特征值的特征向量方向上存在最大的二阶导数。事实上在每个特征向量方向的二阶导数都等于相应的特征值。在其他方向上二阶导数等于特征值的加权平均值。特征向量方向上的相应特征值即是在该方向上的二阶导数。

特征向量定义了二次交叉项为零的坐标系。特征向量被称为函数轮廓线的主轴。图 8-6 所示为这些概念在二维时的情形。该图表明第一特征值小于第二特征值，所以在第一特征向量的方向上的曲率半径(二阶导数)最小。这意味着在此方向上的轮廓线之间的距离更大。在第二特征向量方向上存在最大的曲率半径，所以在此方向上轮廓线之间距离更小。

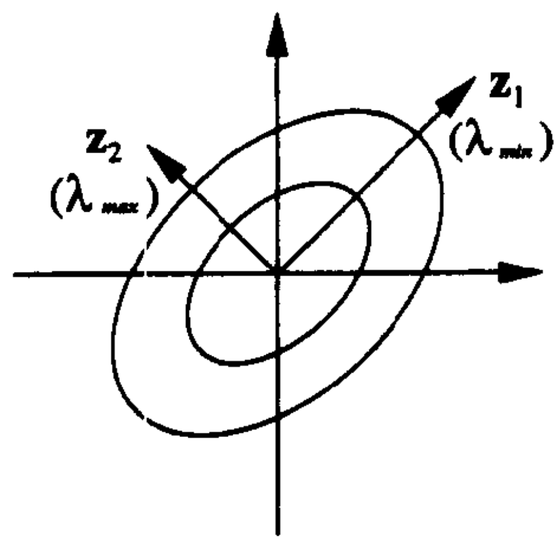



图 8-6 特征向量的二维情形

注意：在图 8-6 中仅当两个特征值同号时才有效，以确保要么存在一个强极小点，要么存在一个强极大点。本例中的轮廓线都是椭园。后面我们将讨论另外的例子，其中之一的特征值异号，另外一个特征值为零。

例一，考虑下列函数：

$$F(\mathbf{x}) = x_1^2 + x_2^2 = \frac{1}{2}\mathbf{x}^T\begin{bmatrix}2 & 0\\ 0 & 2\end{bmatrix}\mathbf{x} \tag{8.52}$$

赫森矩阵及其特征值和特征向量为

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix}2 & 0\\ 0 & 2\end{bmatrix}, \lambda_1 = 2, \mathbf{z}_1 = \begin{bmatrix}1\\ 0\end{bmatrix}, \lambda_2 = 2, \mathbf{z}_2 = \begin{bmatrix}0\\ 1\end{bmatrix} \tag{8.53}$$

(实际上任何两个相互独立的向量都可以成为本例中的特征向量。这里的特征值为多重特征值，其特征向量为一个平面。)

8-15

因为所有的特征值相等，所以在各个方向上的曲率相等，函数的轮廓线为圆。图 8-7 所示为这个函数的轮廓线图和 3-D 图(一个圆形空洞)。

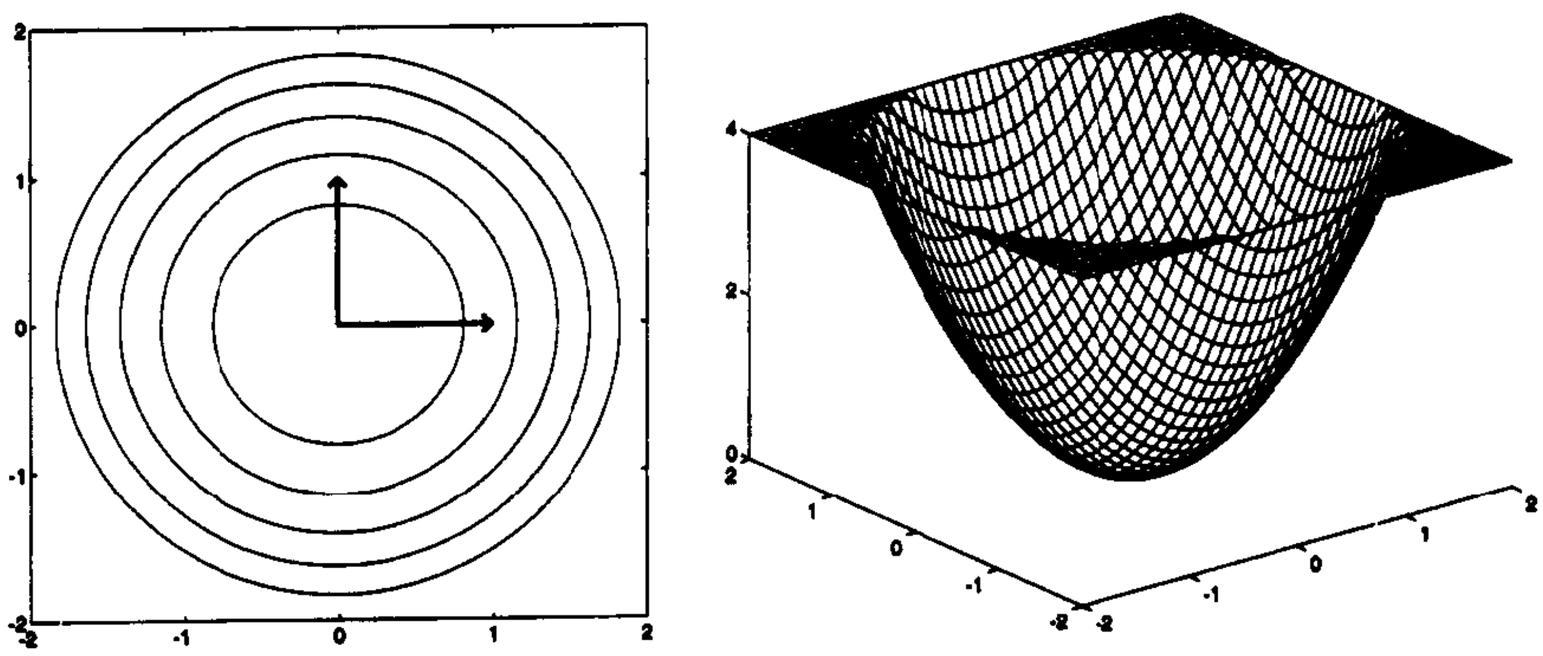


图 8-7 圆形空洞

现在考虑另一个具有相异特征值的例子，其二次函数为：

$$F(\mathbf{x}) = x_1^2 + x_1x_2 + x_2^2 = \frac{1}{2}\mathbf{x}^T\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}\mathbf{x} \tag{8.54}$$

赫森矩阵及其特征值和特征向量为

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \lambda_1 = 1, \mathbf{z}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \lambda_2 = 3, \mathbf{z}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{8.55}$$

(第 6 章讨论过特征向量不是惟一的，它们可以加上任意系数变成很多个。)这里，在 $\mathbf{z}_2$ 方向上曲率最大，所以在此方向上轮廓线密度较大。图 8-8 所示为该函数的轮廓线图及 3-D 图(一个椭圆空洞)。 8-16

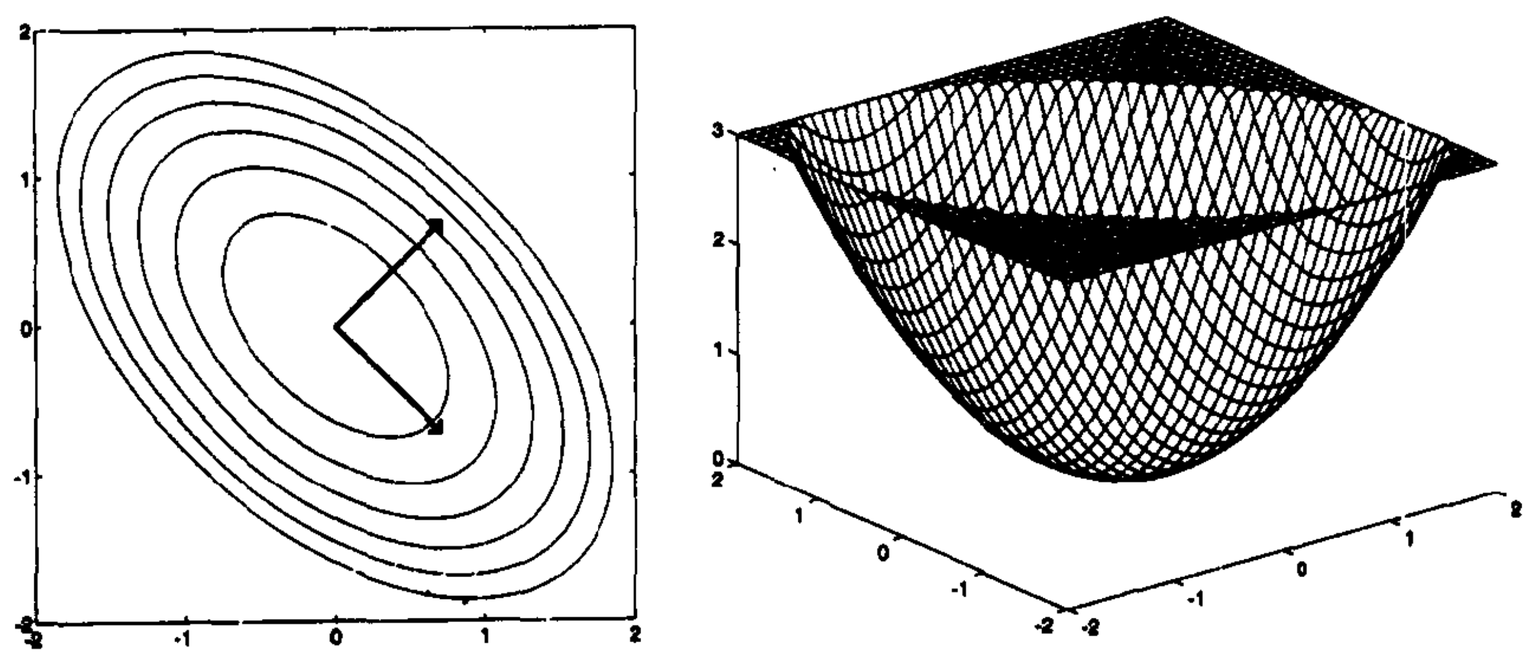


图 8-8　椭圆空洞

如果特征值异号会发生什么现象？考虑下列函数：

$$F(\mathbf{x}) = -\frac{1}{4}x_1^2 - \frac{3}{2}x_1x_2 - \frac{1}{4}x_2^2 = \frac{1}{2}\mathbf{x}^T\begin{bmatrix} -0.5 & -1.5 \\ -1.5 & -0.5 \end{bmatrix}\mathbf{x} \tag{8.56}$$

赫森矩阵及其特征值及特征向量分别是

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} -0.5 & -1.5 \\ -1.5 & -0.5 \end{bmatrix}, \lambda_1 = 1, \mathbf{z}_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \lambda_2 = -2, \mathbf{z}_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \tag{8.57}$$

第一个特征值为正，故在 $\mathbf{z}_1$ 方向上的曲率为正。第二个特征值为负，故在 $\mathbf{z}_2$ 方向上的曲率为负。由于第二个特征值的绝对值大于第一个特征值的绝对值，故在 $\mathbf{z}_2$ 方向上的轮廓线更密。

图 8-9 所示为该函数的轮廓线图和 3-D 图(为一个伸长的鞍形)。注意驻点

$$\mathbf{x}^* = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{8.58}$$

不是一个强极小点，因为赫森矩阵非正定。又由于这里的特征值反号，故赫森矩阵是不确定的(见[Brog91])，因而其驻点为鞍点。在第一个特征向量(正的特征值)上该点为函数的极小点，但是在第二个特征向量方向上(特征值为负)，该点是函数的极大点。 8-17

最后一个例子：存在一个为零的特征值。其函数为

$$F(\mathbf{x}) = \frac{1}{2}x_1^2 - x_1x_2 + \frac{1}{2}x_2^2 = \frac{1}{2}\mathbf{x}^T\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\mathbf{x} \tag{8.59}$$

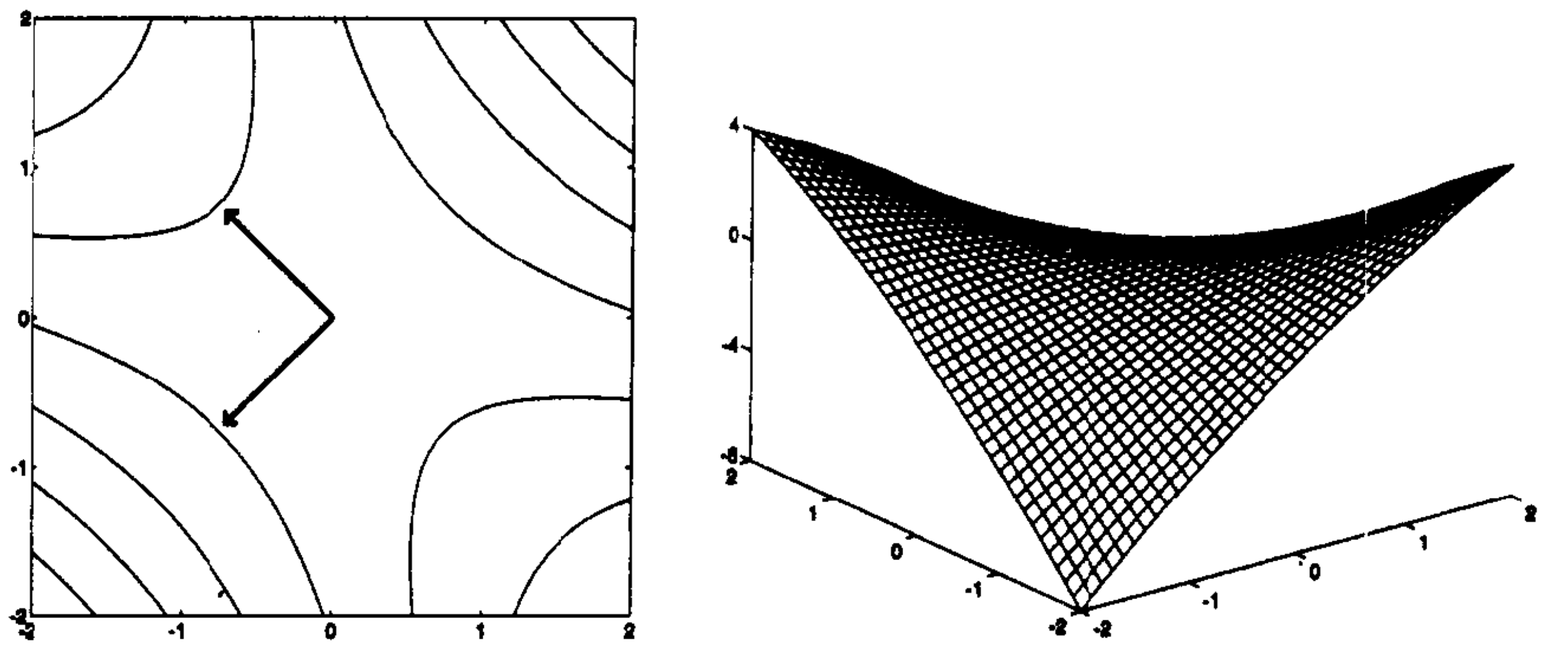


图 8-9 延伸的鞍形

赫森矩阵及其特征值和特征向量为

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \lambda_1 = 2, \mathbf{z}_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \lambda_2 = 0, \mathbf{z}_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \tag{8.60}$$

第二个特征值为零，故在 $\mathbf{z}_2$ 方向上曲率为零。图 8-10 所示为该函数的轮廓线图和 3-D 图(一个驻点凹槽)。本例中的赫森矩阵是半正定的。故在与第二个特征向量对应的直线

$$x_1 = x_2 \tag{8.61}$$

上存在一个弱极小点。

对于二次函数而言，强极小点存在的条件是赫森矩阵必须是正定的。对于高阶函数而言，
8-18 当赫森矩阵为半正定时，也可能存在一个强极小点。(见前面的“极小点”一节中的讨论。)

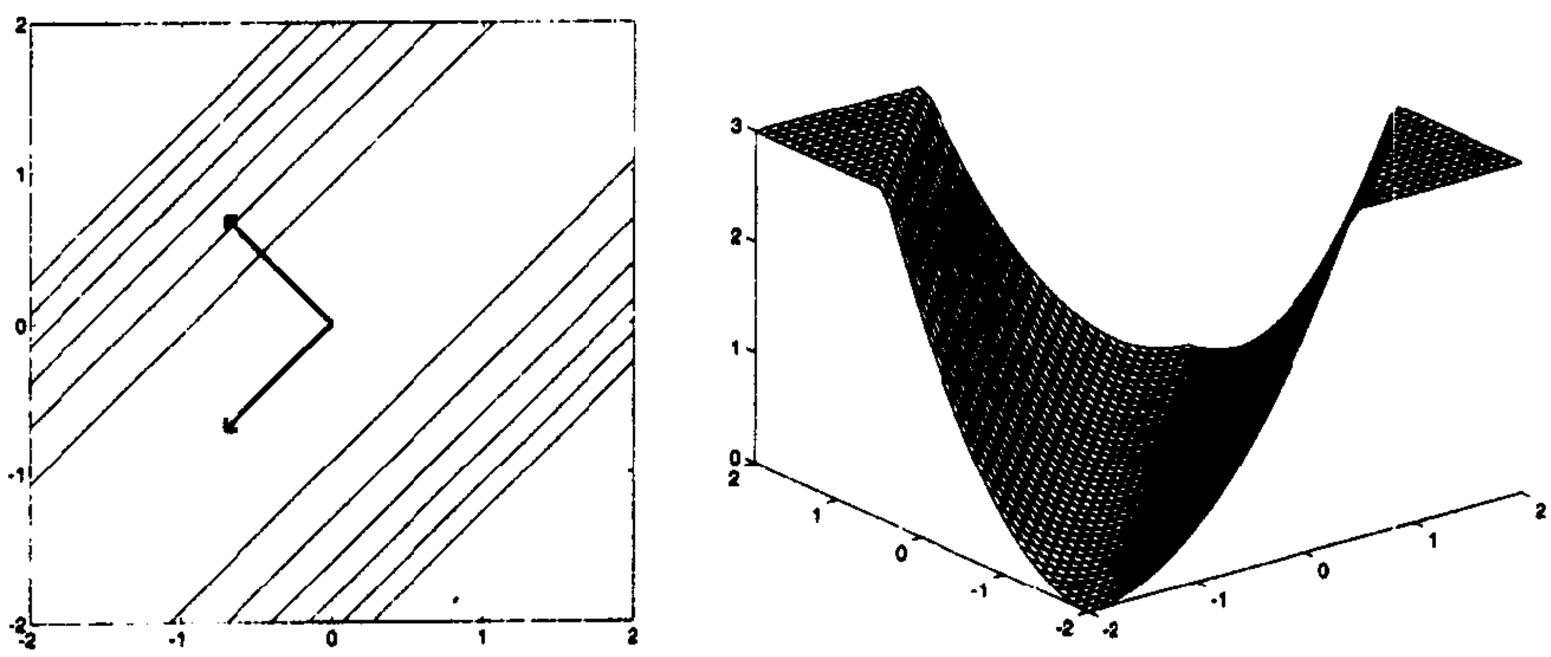


图 8-10 驻点凹槽

试验其他二次函数请用 *Neural Network Design Demonstration* **Quadratic Function** (**nnd8qf**)。

现将二次函数的一些特点小结如下：

1) 如果赫森矩阵的所有特征值为正，则函数有一个强极小点。
2) 如果赫森矩阵的所有特征值为负，则函数有一个强极大点。
3) 如果赫森矩阵的特征值有正有负，则函数有一个鞍点。

4) 如果赫森矩阵的所有特征值为非负，但某些特征值为零，则函数要么有一个弱极小点(见图 8-10)，要么没有驻点(见例题 P8.7)。

5) 如果赫森矩阵的所有特征值为非正，但某些特征值为零，则函数要么有一个弱极大点，要么没有驻点。

注意：在这些讨论中为了使问题简化，我们假设二次函数的驻点在坐标原点，而且函数在该点的函数值为零。这要求式(8.35)中的 **d** 项和 $c$ 项都为零。如果 $c$ 不等于零，则函数只在每个点上增大 $c$ 倍，轮廓线形状不会变化。当 **d** 不等于零，但 **A** 可逆时，轮廓线的形状不变，但函数的驻点移到

$$\mathbf{x}^* = -\mathbf{A}^{-1}\mathbf{d} \tag{8.62}$$

如果 **A** 不可逆(存在为零的特征值)且 **d** 不为零，则不存在驻点(见例题 P8.9)。 8-19

## 8.3 小结

**泰勒级数**

$$F(\mathbf{x}) = F(\mathbf{x}^*) + \nabla F(\mathbf{x})^T\Big|_{\mathbf{x}=\mathbf{x}^*}(\mathbf{x}-\mathbf{x}^*) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^*)^T\nabla^2 F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*}(\mathbf{x}-\mathbf{x}^*) + \cdots$$

**梯度**

$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1}F(\mathbf{x}) & \frac{\partial}{\partial x_2}F(\mathbf{x}) & \cdots & \frac{\partial}{\partial x_n}F(\mathbf{x}) \end{bmatrix}^T$$

**赫森矩阵**

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2}{\partial x_1^2}F(\mathbf{x}) & \frac{\partial^2}{\partial x_2\partial x_2}F(\mathbf{x}) & \cdots & \frac{\partial^2}{\partial x_1\partial x_n}F(\mathbf{x}) \\ \frac{\partial^2}{\partial x_2\partial x_1}F(\mathbf{x}) & \frac{\partial^2}{\partial x_2^2}F(\mathbf{x}) & \cdots & \frac{\partial^2}{\partial x_2\partial x_n}F(\mathbf{x}) \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2}{\partial x_1\partial x_1}F(\mathbf{x}) & \frac{\partial^2}{\partial x_1\partial x_2}F(\mathbf{x}) & \cdots & \frac{\partial^2}{\partial x_n^2}F(\mathbf{x}) \end{bmatrix}$$

**方向导数**

**一阶方向导数**

$$\frac{\mathbf{p}^T\nabla F(\mathbf{x})}{\|\mathbf{p}\|}$$

**二阶方向导数**

$$\frac{\mathbf{p}^T\nabla^2 F(\mathbf{x})\mathbf{p}}{\|\mathbf{p}\|^2}$$ 8-20

**极小点**

**强极小点**

称 $\mathbf{x}^*$ 为 $F(\mathbf{x})$的一个强极小点，如果存在纯量 $\delta > 0$，使得 $F(\mathbf{x}) < F(\mathbf{x}+\Delta\mathbf{x})$对所有的

$\Delta\mathbf{x}(\delta > \|\mathbf{x}\| > 0)$都成立。

**全局极小点**

称 $\mathbf{x}^*$ 为 $F(\mathbf{x})$的惟一全局极小点，如果 $F(\mathbf{x}) < F(\mathbf{x}+\Delta\mathbf{x})$对所有的 $\Delta\mathbf{x}\neq 0$ 都成立。

**弱极小点**

称 $\mathbf{x}^*$ 为 $F(\mathbf{x})$的一个弱极小点，如果它不是一个强极小点，且存在纯量 $\delta>0$，使对于任意的满足 $\delta > \|\Delta\mathbf{x}\| > 0$ 的 $\Delta\mathbf{x}$，都有 $F(\mathbf{x}) \leqslant F(\mathbf{x}+\Delta\mathbf{x})$。

**最优化的必要条件**

**一阶条件**

$$\nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*} = \mathbf{0} \quad \text{（驻点）}$$

**二阶条件**

$$\nabla^2 F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*} \geqslant 0 \quad \text{（赫森矩阵为半正定）}$$

**二次函数**

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{d}^T\mathbf{x} + c$$

**梯度**

$$\nabla F(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{d}$$

**赫森矩阵**

$$\nabla^2 F(\mathbf{x}) = \mathbf{A}$$

**方向导数**

8-21

$$\lambda_{min} \leqslant \frac{\mathbf{p}^T\mathbf{A}\mathbf{p}}{\|\mathbf{p}\|^2} \leqslant \lambda_{max}$$

## 8.4 例题

**P8.1** 图 8-1 所示为余弦函数在 $x^* = 0$ 的三个近似。试在点 $x^* = \pi/2$ 重复该过程。

**解**

将要作近似的函数为

$$F(x) = \cos(x)$$

$F(x)$在点 $x^* = \pi/2$ 的泰勒级数展开为

$$F(x) = \cos(x) = \cos\left(\frac{\pi}{2}\right) - \sin\left(\frac{\pi}{2}\right)\left(x - \frac{\pi}{2}\right) - \frac{1}{2}\cos\left(\frac{\pi}{2}\right)\left(x - \frac{\pi}{2}\right)^2 + \frac{1}{6}\sin\left(\frac{\pi}{2}\right)\left(x - \frac{\pi}{2}\right)^3 + \cdots$$

$$= -\left(x - \frac{\pi}{2}\right) + \frac{1}{6}\left(x - \frac{\pi}{2}\right)^3 - \frac{1}{120}\left(x - \frac{\pi}{2}\right)^5 + \cdots$$

$F(x)$的零阶近似是

$$F(x) \approx F_0(x) = 0$$

$F(x)$的一阶近似是

$$F(x) \approx F_1(x) = -\left(x - \frac{\pi}{2}\right) = \frac{\pi}{2} - x$$

(注意：由于二阶导数为零，故二阶近似等于一阶近似。)

$F(x)$三阶近似是

$$F(x) \approx F_3(x) = -\left(x - \frac{\pi}{2}\right) + \frac{1}{6}\left(x - \frac{\pi}{2}\right)^3$$

图 8-11 所示为这三个近似的图象。这里的零阶近似非常差，而一阶近似在一个适当的范围内是精确的。将这一结果与图 8-1 对比发现，在那种情况下，我们在一个局部极大点 $x^* = 0$ 展开，所以一阶导数为零。

*检查泰勒级数在其他点的展开请用 Neural Network Design Demonstration Taylor Series* (**nnd8ts**)。

8-22

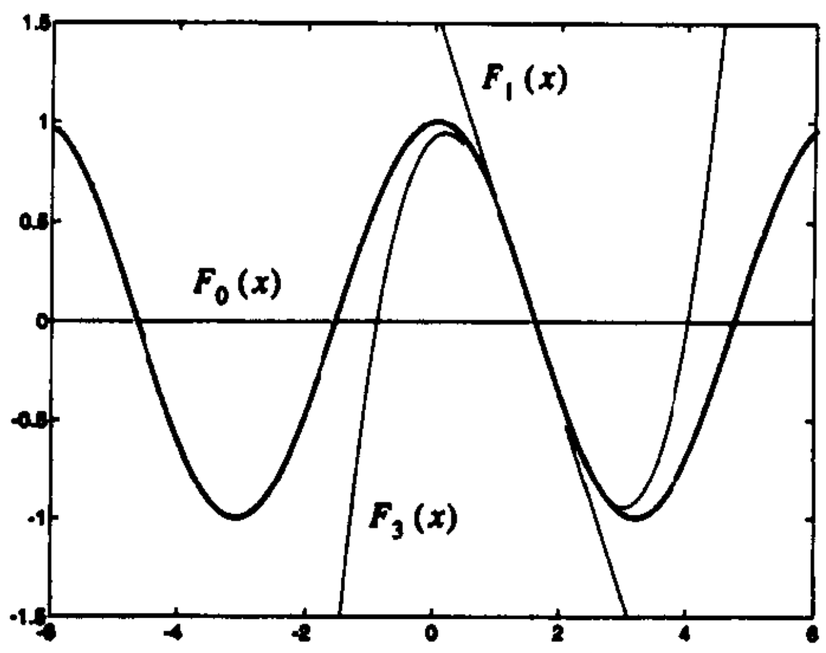



图 8-11　在 $x = \pi/2$ 的余弦函数近似

**P8.2** 回到图 8-4 所示的函数。已知该函数有两个强极小点。求该函数在两个极小点的泰勒级数展开。

**解**

函数的表达式为

$$F(\mathbf{x}) = (x_2 - x_1)^4 + 8x_1x_2 - x_1 + x_2 + 3$$

欲求其二阶泰勒级数展开，必须先求出 $F(\mathbf{x})$的梯度及赫森矩阵。梯度为

$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} -4(x_2 - x_1)^3 + 8x_2 - 1 \\ 4(x_2 - x_1)^3 + 8x_1 + 1 \end{bmatrix}$$

赫森矩阵为

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2}{\partial x_1^2} F(\mathbf{x}) & \frac{\partial^2}{\partial x_1 \partial x_2} F(\mathbf{x}) \\ \frac{\partial^2}{\partial x_2 \partial x_1} F(\mathbf{x}) & \frac{\partial^2}{\partial x_2^2} F(\mathbf{x}) \end{bmatrix}$$

8-23

$$= \begin{bmatrix} 12(x_2 - x_1)^2 & -12(x_2 - x_1)^2 + 8 \\ -12(x_2 - x_1)^2 + 8 & 12(x_2 - x_1)^2 \end{bmatrix}$$

在点 $\mathbf{x}^1 = [-0.42 \quad 0.42]^T$ 有一个强极小点，在 $\mathbf{x}^2 = [0.55 \quad -0.55]^T$ 有另一强极小点。在这两点对 $F(\mathbf{x})$进行二阶泰勒级数展开：

$$F^1(\mathbf{x}) = F(\mathbf{x}^1) + \nabla F(\mathbf{x})^T\Big|_{\mathbf{x}=\mathbf{x}^1}(\mathbf{x} - \mathbf{x}^1) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^1)^T\nabla^2 F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^1}(\mathbf{x} - \mathbf{x}^1)$$

$$= 2.93 + \frac{1}{2}\left(\mathbf{x} - \begin{bmatrix} -0.42 \\ 0.42 \end{bmatrix}\right)^T \begin{bmatrix} 8.42 & -0.42 \\ -0.42 & 8.42 \end{bmatrix}\left(\mathbf{x} - \begin{bmatrix} -0.42 \\ 0.42 \end{bmatrix}\right)$$

可将它简化为

$$F^1(\mathbf{x}) = 4.49 - [-3.7128 \quad 3.7128]\mathbf{x} + \frac{1}{2}\mathbf{x}^T\begin{bmatrix} 8.42 & -0.42 \\ -0.42 & 8.42 \end{bmatrix}\mathbf{x}$$

同理可得 $\mathbf{x}^2$ 点的展开为

$$F^2(\mathbf{x}) = 7.41 - [11.781 \quad -11.781]\mathbf{x} + \frac{1}{2}\mathbf{x}^T\begin{bmatrix} 14.71 & -6.71 \\ -6.71 & 14.71 \end{bmatrix}\mathbf{x}$$

图 8-12 所示为原函数及其两个近似的图象。

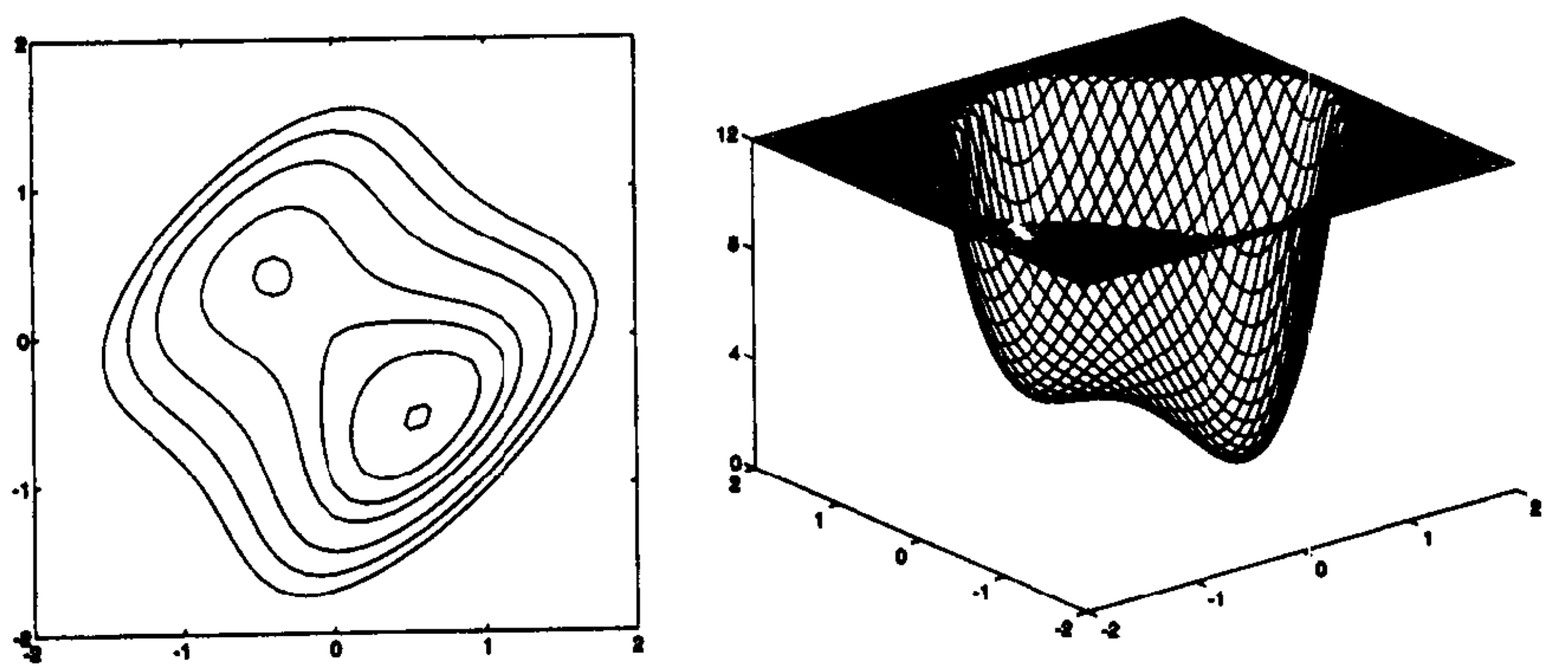
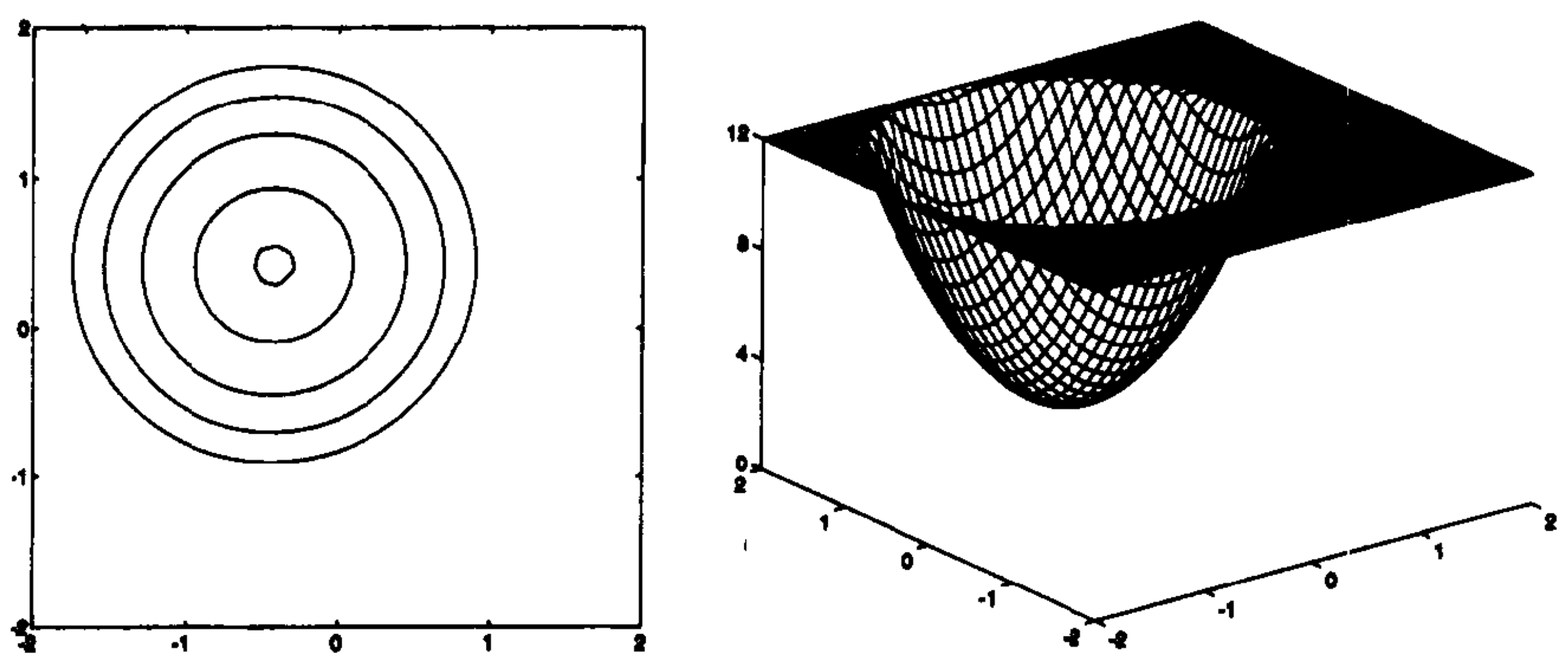


图 8-12 例题 P8.2 的函数 $F(\mathbf{x})$



8-24

图 8-13 例题 P8.2 的函数 $F^1(\mathbf{x})$

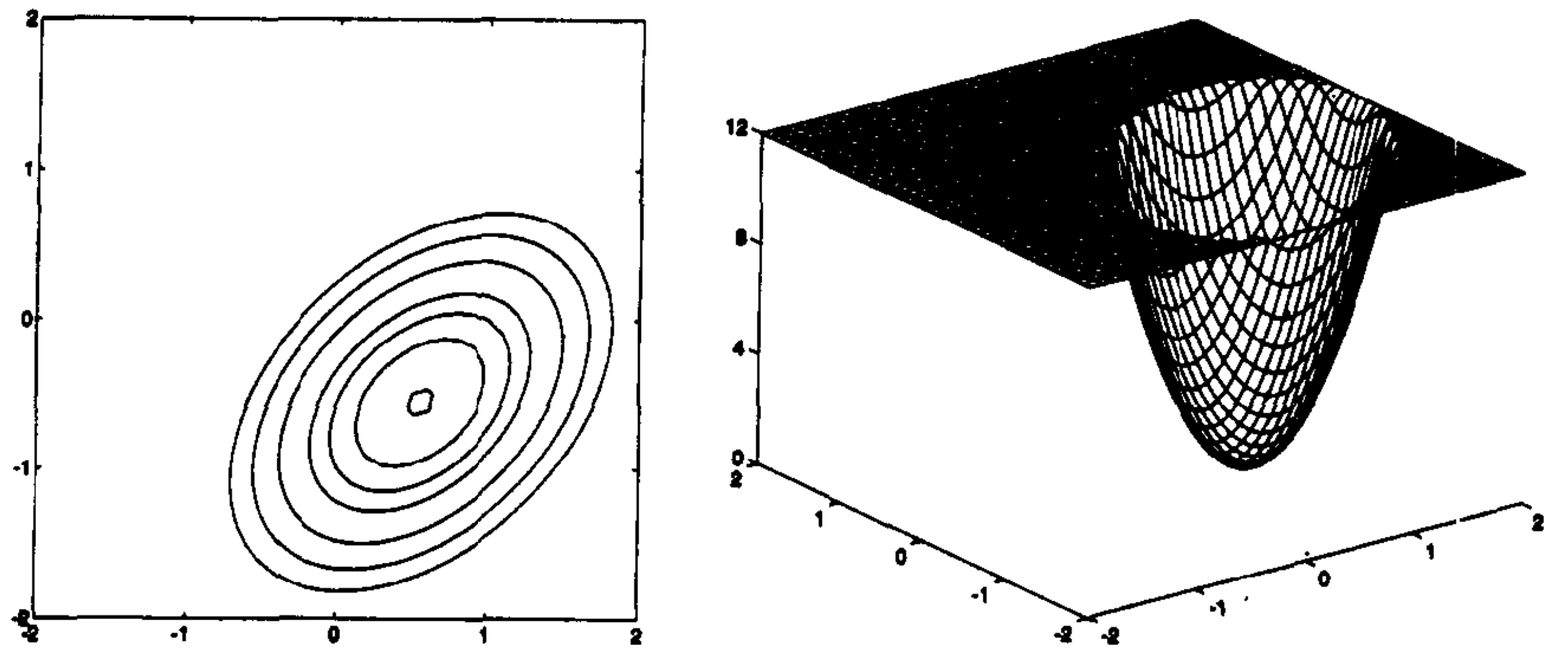


图 8-14　例题 P8.2 的函数 $F^2(\mathbf{x})$

检查函数在其他点的泰勒级数展开请用 *Neural Network Design Demonstration Vector Taylor Series* (**nnd8ts2**)。

**P8.3** 给定下列函数，求在 $\mathbf{x}=[0\quad 0]^T$ 处与轮廓线相切的切线方程。

$$F(\mathbf{x}) = (2 + x_1)^2 + 5(1 - x_1 - x_2^2)^2$$

**解**

解决这个问题要用到方向导数。$F(\mathbf{x})$沿一条轮廓线的切线方向的导数是什么？由于轮廓线是函数值不变的线，则沿轮廓线的 $F(\mathbf{x})$的导数为零。所以设方向导数为零，可求轮廓线切线方程。

8-25

首先求梯度：

$$\nabla F(\mathbf{x}) = \begin{bmatrix} 2(2 + x_1) + 10(1 - x_1 + x_2^2)(-1) \\ 10(1 - x_1 - x_2^2)(-2x_2) \end{bmatrix}$$

$$= \begin{bmatrix} -6 + 12x_1 + 10x_2^2 \\ -20x_2 + 20x_1x_2 + 20x_2^3 \end{bmatrix}$$

在 $\mathbf{x}^* = [0\quad 0]^T$，有

$$\nabla F(\mathbf{x}^*) = \begin{bmatrix} -6 \\ 0 \end{bmatrix}$$

由于 $F(\mathbf{x})$在向量 $\mathbf{p}$ 方向上的导数是

$$\frac{\mathbf{p}^T \nabla F(\mathbf{x})}{\|\mathbf{p}\|}$$

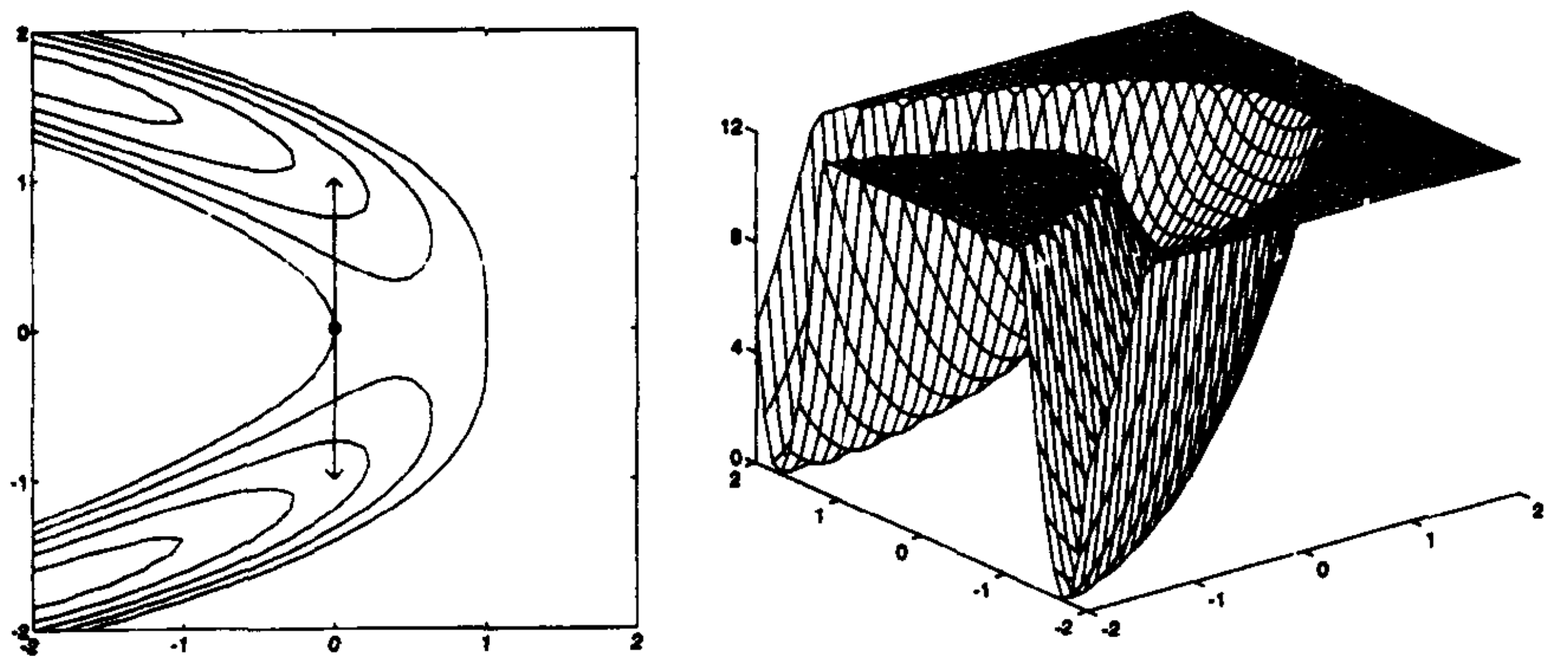
所以欲求穿过点 $\mathbf{x}^* = [0\quad 0]^T$ 且导数为零的切线方程，可以设在 $\Delta\mathbf{x}$ 方向的方向导数的分子为零：

$$\Delta\mathbf{x}^T \nabla F(\mathbf{x}^*) = 0$$

这里 $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}^*$。在这种情形下有

$$\mathbf{x}^T \begin{bmatrix} -6 \\ 0 \end{bmatrix} = 0 \quad 或\ x_1 = 0$$

这个结果见图 8-15。



8-26

图 8-15 例题 P8.3 中 $F(\mathbf{x})$的图

**P8.4** 求下列四阶多项式的所有驻点并检验它们是否为极小点。

$$F(x) = x^4 - \frac{2}{3}x^3 - 2x^2 + 2x + 4$$

**解**

欲求驻点，先令 $F(\mathbf{x})$导数为零：

$$\frac{\mathrm{d}}{\mathrm{d}x}F(x) = 4x^3 - 2x^2 - 4x + 2 = 0$$

使用 MATLAB 求这个多项式方程的根：

```
coef=[4  -2  -4  2];
stapoints=roots(coef);
stapoints'
ans =
    1.0000      -1.0000      0.5000
```

现在求以上各点处的二阶导数值。$F(x)$的二阶导数为

$$\frac{\mathrm{d}^2}{\mathrm{d}x^2}F(x) = 12x^2 - 4x - 4$$

各驻点的二阶导数为

$$\left(\frac{\mathrm{d}^2}{\mathrm{d}x^2}F(1) = 4\right),\left(\frac{\mathrm{d}^2}{\mathrm{d}x^2}F(-1) = 12\right),\left(\frac{\mathrm{d}^2}{\mathrm{d}x^2}F(0.5) \doteq -3\right)$$

所以在 1 和 -1 处存在强局部极小点(因为二阶导数为正)，在 0.5 处存在强局部极大点(因为二阶导数为负)。为了找出全局极小点，分别求出在两个局部极小点的函数值：

$$(F(1) = 4.333),\ (F(-1) = 1.667)$$

故全局极小点在 $x = -1$。但能否肯定这一点是全局极小点？当 $x \to \infty$ 或 $x \to -\infty$ 时会发生什么现象？本例中，由于 $x$ 最高次项的系数大于零且该项为偶次项($x^4$)，当 $x \to \pm\infty$ 时 $F(x)$都趋向 $\infty$，故可以肯定在 $x = -1$ 处存在全局最小。函数图见图 8-16。

8-27

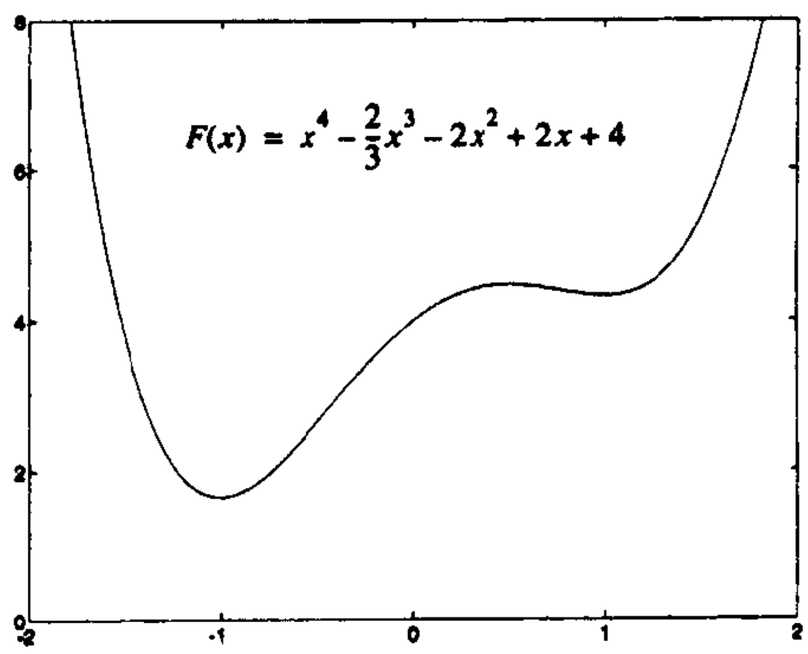



图 8-16　例题 P8.4 的 $F(x)$图

**P8.5** 例题 P8.2 中函数有三个驻点：

$$\mathbf{x}^1 = \begin{bmatrix} -0.42 \\ 0.42 \end{bmatrix}, \mathbf{x}^2 = \begin{bmatrix} -0.13 \\ 0.13 \end{bmatrix}, \mathbf{x}^3 = \begin{bmatrix} 0.55 \\ -0.55 \end{bmatrix}$$

试检验这些驻点是否为局部极小点。

**解**

从例题 P8.2 可知函数 $F(\mathbf{x})$的赫森矩阵为

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} 12(x_2 - x_1)^2 & -12(x_2 - x_1)^2 + 8 \\ -12(x_2 - x_1)^2 + 8 & 12(x_2 - x_1)^2 \end{bmatrix}$$

矩阵的正定性可由特征值确定。如果所有特征值为正的，则矩阵为正定矩阵，即存在一个强极小点。如果特征值非负，则矩阵为半正定阵，即要么存在强极小点，要么存在一个弱极小点。如果特征值一正一负，则矩阵为不定型，存在一个鞍点。

在 $\mathbf{x}^1$ 的赫森矩阵为

$$\nabla^2 F(\mathbf{x}^1) = \begin{bmatrix} 8.42 & -0.42 \\ -0.42 & 8.42 \end{bmatrix}$$

该矩阵的特征值是

$$\lambda_1 = 8.84, \quad \lambda_2 = 8.0$$

8-28

故 $\mathbf{x}^1$ 一定是一个强极小点。

在 $\mathbf{x}^2$ 的赫森矩阵为

$$\nabla^2 F(\mathbf{x}^2) = \begin{bmatrix} 0.87 & 7.13 \\ 7.13 & 0.87 \end{bmatrix}$$

该矩阵的特征值为

$$\lambda_1 = -6.26, \quad \lambda_2 = 8.0$$

所以 $\mathbf{x}^2$ 一定是一个鞍点。在一个方向上的曲率为负，在另一个方向上的曲率为正。负的曲率在第一特征向量方向上，正的曲率在第二个特征向量的方向上。特征向量为

$$\mathbf{z}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \mathbf{z}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

（注意，这与我们在 8.2.3 节的讨论一致。）

在 $\mathbf{x}^3$ 的赫森矩阵为

$$\nabla^2 F(\mathbf{x}^3) = \begin{bmatrix} 14.7 & -6.71 \\ -6.71 & 14.7 \end{bmatrix}$$

该矩阵特征值为

$$\lambda_1 = 21.42, \quad \lambda_2 = 8.0$$

所以 $\mathbf{x}^3$ 是一个强极小点。

可用 *Neural Network Design Demonstration* *Vector Taylor Series* (**nnd8ts2**)检验这些结果。

**P8.6** 现在将本章的概念用于一个神经网络问题。见图 8-17 所示的线性网络，设该网络的期望输入/输出为：

$$\{(p_1 = 2),(t = 0.5)\},\{(p_2 = -1),(t_2 = 0)\}$$

试确定网络的下列性能指数函数：

8-29

$$F(\mathbf{x}) = (t_1 - a_1(\mathbf{x}))^2 + (t_2 - a_2(\mathbf{x}))^2$$

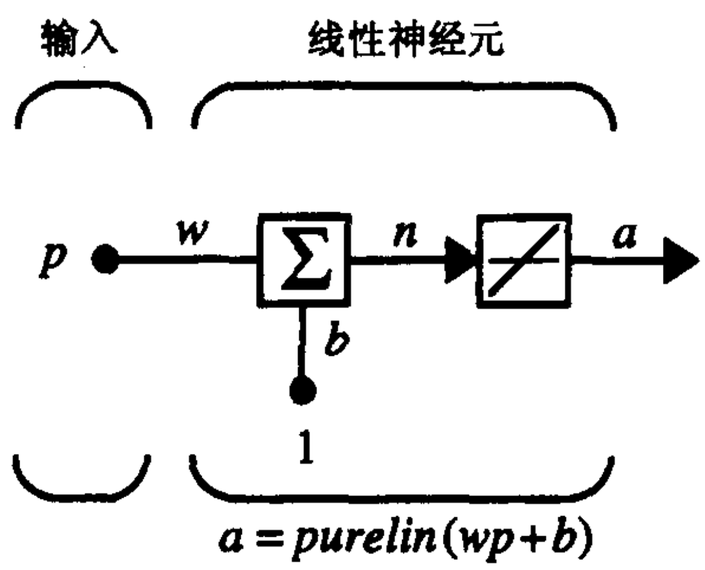



图 8-17 例题 P8.6 的线性网络

**解**

该网络的参数为 $w$ 和 $b$，构成参数向量

$$\mathbf{x} = \begin{bmatrix} w \\ b \end{bmatrix}$$

确定 $F(\mathbf{x})$的步骤如下：首先确定性能指数 $F(\mathbf{x})$为一个二次函数，然后求出其赫森矩阵的特征值和特征向量，并用它们描绘函数的轮廓线图。

先把 $F(\mathbf{x})$写成参数向量 $\mathbf{x}$ 的显式形式：

$$F(\mathbf{x}) = e_1^2 + e_2^2$$

其中

$$(e_1 = t_1 - (wp_1 + b)),(e_2 = t_2 - (wp_2 + b))$$

这也可写成矩阵形式：

$$F(\mathbf{x}) = \mathbf{e}^T\mathbf{e}$$

其中

$$\mathbf{e} = \mathbf{t} - \begin{bmatrix} p_1 & 1 \\ p_2 & 1 \end{bmatrix}\mathbf{x} = t - \mathbf{G}\mathbf{x}$$

现在性能指数函数可写成如下形式：

$$F(\mathbf{x}) = [\mathbf{t} - \mathbf{G}\mathbf{x}]^T[\mathbf{t} - \mathbf{G}\mathbf{x}] = \mathbf{t}^T\mathbf{t} - 2\mathbf{t}^T\mathbf{G}\mathbf{x} + \mathbf{x}^T\mathbf{G}^T\mathbf{G}\mathbf{x}$$ 8-30

与式(8.35)

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{d}^T\mathbf{x} + c$$

比较，可知这个线性网络的性能指数函数是二次函数，且

$$c = \mathbf{t}^T\mathbf{t}, \quad \mathbf{d} = -2\mathbf{G}^T\mathbf{t}, \quad \mathbf{A} = 2\mathbf{G}^T\mathbf{G}$$

该二次函数的梯度由式(8.38)得给出：

$$\nabla F(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{d} = 2\mathbf{G}^T\mathbf{G}\mathbf{x} - 2\mathbf{G}^T\mathbf{t}$$

使梯度为零的点即函数的驻点(也是函数轮廓线的中心点)：

$$\mathbf{x}^* = -\mathbf{A}^{-1}\mathbf{d} = [\mathbf{G}^T\mathbf{G}]^{-1}\mathbf{G}^T\mathbf{t}$$

由于

$$\mathbf{G} = \begin{bmatrix} p_1 & 1 \\ p_2 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ -1 & 1 \end{bmatrix}, \mathbf{t} = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}$$

故有

$$\mathbf{x}^* = [\mathbf{G}^T\mathbf{G}]^{-1}\mathbf{G}^T\mathbf{t} = \begin{bmatrix} 5 & 1 \\ 1 & 2 \end{bmatrix}^{-1}\begin{bmatrix} 1 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.167 \\ 0.167 \end{bmatrix}$$

(所以网络最优参数是 $w = 0.167$，$b = 0.167$。)

由式(8.39)得二次函数的赫森矩阵为

$$\nabla^2 F(\mathbf{x}) = \mathbf{A} = 2\mathbf{G}^T\mathbf{G} = \begin{bmatrix} 10 & 2 \\ 2 & 4 \end{bmatrix}$$

为了描出轮廓线，须先求出赫森矩阵的特征值和特征向量。在这种情形下有：

$$\left\{(\lambda_1 = 10.6), \left(\mathbf{z}_1 = \begin{bmatrix} 1 \\ 0.3 \end{bmatrix}\right)\right\}, \left\{(\lambda_2 = 3.4), \left(\mathbf{z}_2 = \begin{bmatrix} 0.3 \\ -1 \end{bmatrix}\right)\right\}$$

所以 $\mathbf{x}^*$ 是一个强极小点。同时，由于这里的第一个特征值比第二个特征值大，所以轮廓线是椭圆，其长轴在第二个特征向量方向上。轮廓线的中心点在 $\mathbf{x}^*$。如图 8-18 所示。 8-31

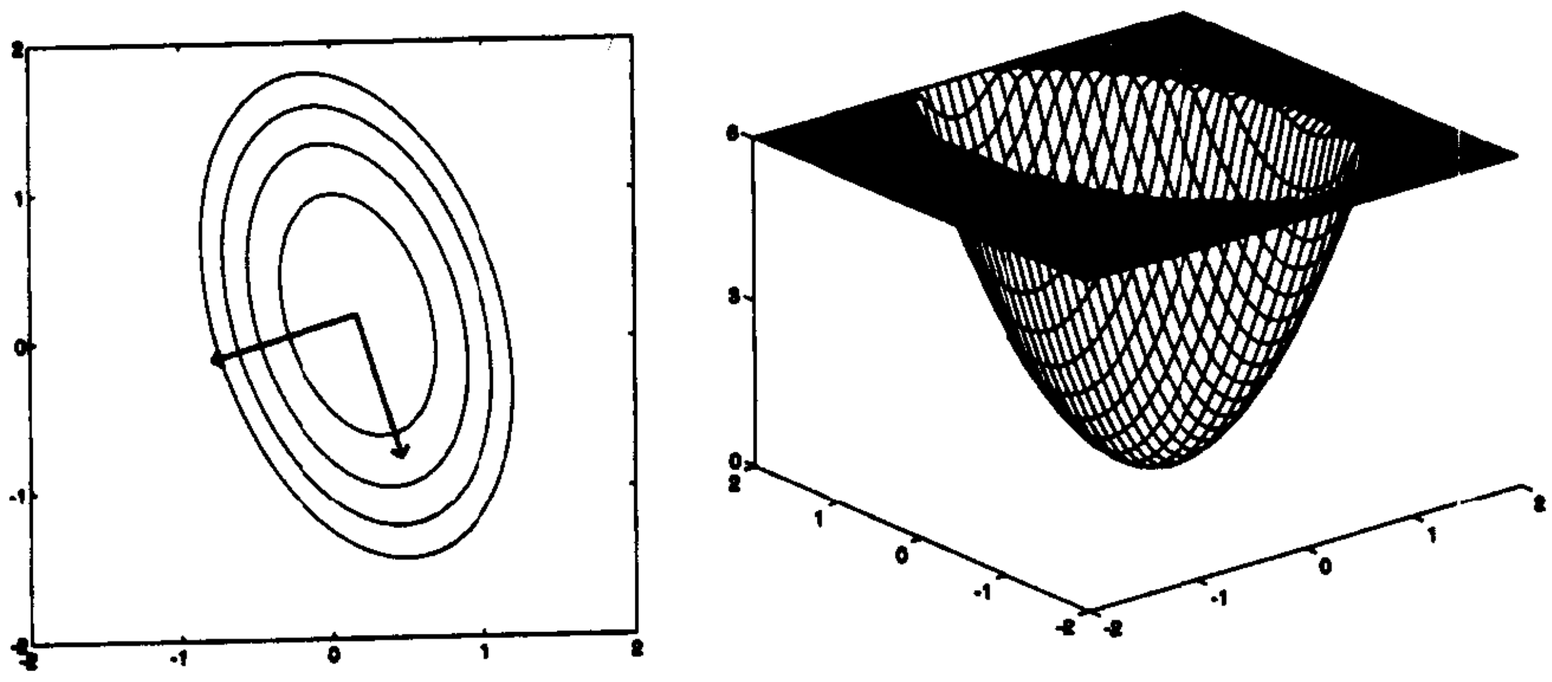


图 8-18　例题 P8.6 的函数图象

**P8.7** 本例讨论一个设有驻点的二次函数，该函数为

$$F(\mathbf{x}) = [1 \ -1]\mathbf{x} + \frac{1}{2}\mathbf{x}^T\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}\mathbf{x}$$

试描出该函数的轮廓线图。

**解**

同例题 P8.6 一样，我们需要先找出赫森矩阵的特征值和特征向量。由函数表达式知其赫森矩阵为

$$\nabla^2 F(\mathbf{x}) = \mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \tag{8.63}$$

其特征值和特征向量为

$$\left\{(\lambda_1 = 0), \left(\mathbf{z}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}\right)\right\}, \left\{(\lambda_2 = 2), \left(\mathbf{z}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right)\right\}$$

第一个特征值为零，因此在第一个特征向量方向上曲率为零。第二个特征值大于零，因此在第二个特征向量方向上的曲率为正。如果 $F(\mathbf{x})$没有线性项，则 $F(\mathbf{x})$的图象为如图 8-10 所示的一个驻点凹槽。本例中我们必须确定线性项是否产生沿凹槽方向(第一个特征向量的方向)的斜坡。

8-32

线性项为

$$F_{lin}(\mathbf{x}) = [1 \quad -1]\mathbf{x}$$

由式(8.36)知该项的梯度为

$$\nabla F_{lin}(\mathbf{x}) = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

这表明线性项在这个梯度方向上增加最快。由于本例中二次项曲率为零，则整个函数在这个方向上是一个线性的斜坡。

所以 $F(\mathbf{x})$在第二个特征向量方向上的曲率大于零，而在第一个特征向量方向上是线性斜坡。图 8-19 为该函数的轮廓线图和 3-D 图。

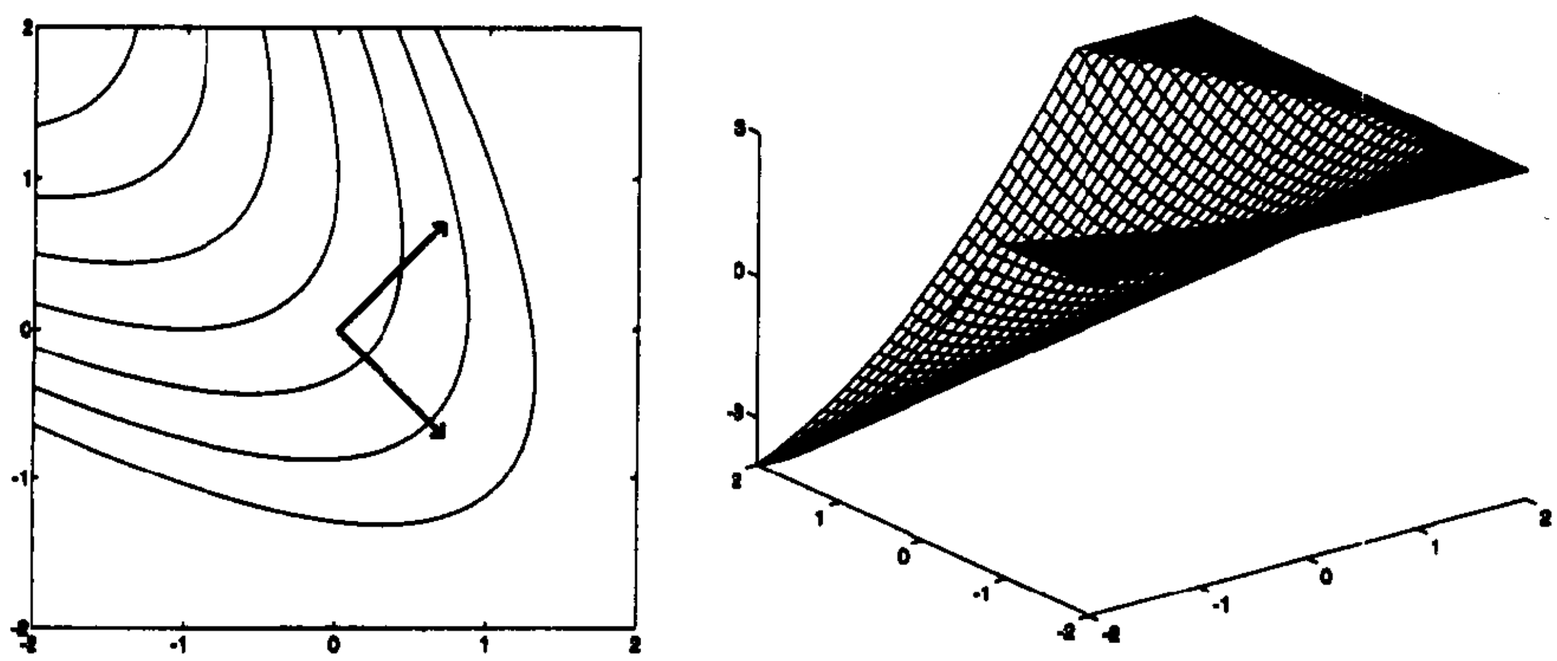


图 8-19 例题 P8.7 的下降凹槽的函数

对于任意特征值为零的赫森矩阵，不可能由下式求出二次函数的驻点：

$$\mathbf{x}^* = -\mathbf{A}^{-1}\mathbf{d}$$

因为在这里赫森矩阵不可逆。这可能意味着像图 8-10 所示的那样存在一个弱极小点，或者如本例中的情况，没有驻点。 8-33

## 8.5 结束语

性能学习是最重要的神经网络学习规则之一。通过性能学习，网络参数能得到调节从而优化网络性能。本章介绍了一些研究性能学习规则必备的工具。学习本章要求达到：

（i）掌握泰勒级数展开及函数的近似表示方法；

（ii）求方向导数；

（iii）掌握求驻点的方法和极小点检验方法；

（iv）画二次函数的轮廓线图形。

在后面各章会广泛地运用这些概念，包括性能学习(第 9 ~ 12 章)和递归网络(第 17 ~ 18 章)。下一章，我们将以本章的概念为基础设计优化性能函数的算法。然后在后面的各章中将这些算法用于神经网络的训练。 8-34

## 参考文献

[Brog91] W. L. Brogan, *Modern Control Theory*, 3rd Ed., Englewood Cliffs, NJ: Prentice - Hall, 1991.

本书是一本关于线性系统的好书。前半部分论述线性代数。该书对线性微分方程求解和线性与非线性系统的稳定性作了很好的阐述。书中包含了许多实际问题。

[Gill81] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, New York: Academic Press, 1981.

本书阐述了优化算法的实际实现。它提出了一种优化方法的激励机制，并讨论了影响算法性能的实现细节问题。

[Himm72] D. M. Himmelblau, *Applied Nonlinear Programming*, New York: McGraw - Hill, 1972.

本书是综合性的非线性优化的课本。它讨论了约束和非约束的优化问题。该书内容详实，包含许多例题详解。

[Sca185] L. E. Scales, *Introduction to Non - Linear Optimization*, New York: Springer-Verlag, 1985.

这是一本可读性很强的描述主要的优化算法的书，书的重点是论述优化的方法而不是存在定理和收敛性的证明。它通过图表和例子直观地解释算法。多数算法都给出了伪码。 8-35

## 习题

**E8.1** 考虑下列纯量函数：

$$F(x) = \frac{1}{x^3 - \frac{3}{4}x - \frac{1}{2}}$$

（i）求 $F(x)$在点 $x = -0.5$ 的二阶泰勒级数近似。

(ii) 求 $F(x)$在点 $x=1.1$ 的二阶泰勒级数近似。

(iii) 画出 $F(x)$和两个近似并讨论它们的精确度。

**E8.2** 考虑下列二元函数:

$$F(\mathbf{x}) = e^{(2x_1^2+2x_2^2+x_1-5x_2+10)}$$

(i) 求 $F(\mathbf{x})$在点 $\mathbf{x}=[0\ 0]^T$ 的二阶泰勒级数近似。

(ii) 求该近似的驻点。

(iii) 求 $F(\mathbf{x})$的驻点(注意 $F(\mathbf{x})$的指数由二次函数组成)。

(iv) 说明两个驻点的区别。(用 MATLAB 画出两个函数图形。)

**E8.3** 在点 $\mathbf{x}=[1\quad 1]^T$ 处求下列函数在方向 $\mathbf{p}=[-1\quad 1]^T$ 上的一阶和二阶方向导数:

(i) $F(\mathbf{x})=\frac{7}{2}x_1^2-6x_1x_2-x_2^2$

(ii) $F(\mathbf{x})=5x_1^2-6x_1x_2+5x_2^2+4x_1+4x_2$

(iii) $F(\mathbf{x})=\frac{9}{2}x_1^2-2x_1x_2+3x_2^2+2x_1-x_2$

8-36 (iv) $F(\mathbf{x})=-\frac{1}{2}(7x_1^2+12x_1x_2-2x_2^2)$

**E8.4** 对函数

$$F(x) = x^4-\frac{1}{2}x^2+1$$

(i) 求驻点;

(ii) 检验驻点是否是极小点和极大点;

(iii) 用 MATLAB 画出函数图象,验证你的答案。

**E8.5** 已知下列二元函数:

$$F(\mathbf{x}) = (x_1+x_2)^4-12x_1x_2+x_1+x_2+1$$

(i) 验证该函数有三个驻点

$$\mathbf{x}^1=\begin{bmatrix}-0.6504\\-0.6504\end{bmatrix},\mathbf{x}^2=\begin{bmatrix}0.085\\0.085\end{bmatrix},\mathbf{x}^3=\begin{bmatrix}0.5655\\0.5655\end{bmatrix}$$

(ii) 检验以上驻点,找出所有极小点、极大点和鞍点;

(iii) 求该函数在每个驻点上的二阶泰勒级数近似;

(iv) 用 MATAB 画出函数及其近似的图象。

**E8.6** 对于习题 E8.3 的函数:

(i) 求出驻点;

(ii) 检验驻点,找出极小点、极大点和鞍点;

(iii) 利用赫森矩阵的特征值和特征向量粗略画出轮廓线图;

(iv) 用 MATLAB 画出函数图以验证你的答案。

**E8.7** 例题 P8.7 中的函数没有驻点。试仅改变向量 **d** 以产生一个驻点。找出一个新的

8-37 非零向量 **d**,以产生一个弱极小点。

# 第9章 性能优化

## 9.1 目的

从第 8 章起本书讨论了性能优化问题，介绍了分析性能曲面的一个工具——泰勒级数展开，并运用这个工具确定最优点必须满足的条件。本章将继续应用泰勒级数展开寻求定位最优点的算法。我们将讨论三类优化算法：最速下降法(steepest descent)，牛顿法以及共轭梯度法(conjugate gradient)。在第 10 ~ 12 章这些算法将用于神经网络的训练。 9-1

## 9.2 理论和实例

前面一章我们开始了性能曲面的研究。现在我们来寻求搜索参数空间和确定性能曲面最优点的算法(求给定神经网络的最优权值和偏置值。)

有意思的是本章的多数算法已经形成和发展了几百年。优化的基本原理早在 17 世纪就由开普勒、费马、牛顿和莱布尼茨这些科学家和数学家提出了。自 1950 年以来，这些原理又被用于高速数字计算机。这方面的成功激起了人们对新的算法进行卓有成效的研究，使得优化理论领域成为数学的一个主要的分支。现在，神经网络的研究者已进入这一巨大的优化理论宝库，并试图将它用于神经网络的训练。这方面的应用刚刚开始，前景光明。

本章的目标是构造优化性能指数 $F(\mathbf{x})$的算法。优化的目的是求出使 $F(\mathbf{x})$最小化的 $\mathbf{x}$ 的值。在这里，所有将要讨论的算法都是迭代的。首先，给定一个初始猜测值 $\mathbf{x}_0$，然后按照等式

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \tag{9.1}$$

或

$$\Delta\mathbf{x}_k = (\mathbf{x}_{k+1} - \mathbf{x}_k) = \alpha_k \mathbf{p}_k \tag{9.2}$$

逐步修改我们的猜测。这里向量 $\mathbf{p}_k$代表一个搜索方向，一个大于零的纯量 $\alpha_k$ 为学习速度，它确定了学习步长。

本章的算法根据搜索方向 $\mathbf{p}_k$ 的不同而不同。我们将讨论三种不同的可能性。另外还有许多种确定学习速度的方法。

### 9.2.1 最速下降法

当用式(9.1)进行最优点迭代时，函数应该在每次迭代时都减小，即

$$F(\mathbf{x}_{k+1}) < F(\mathbf{x}_k) \tag{9.3}$$ 9-2

如何选择向量 $\mathbf{p}_k$使对于充分小的学习速度 $\alpha_k$ 这个迭代都能快速收剑？考虑式(8.9)的 $F(\mathbf{x})$在 $\mathbf{x}_k$的一阶泰勒级数展开：

$$F(\mathbf{x}_{k+1}) = F(\mathbf{x}_k + \Delta\mathbf{x}_k) \approx F(\mathbf{x}_k) + \mathbf{g}_k^T \Delta\mathbf{x}_k \tag{9.4}$$

这里 $\mathbf{g}_k$为在旧猜测值 $\mathbf{x}_k$的梯度：

$$\mathbf{g}_k \equiv \nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_k} \tag{9.5}$$

要使 $F(\mathbf{x}_{k+1}) < F(\mathbf{x}_k)$，式(9.4)右边的第二项必须为负，即

$$\mathbf{g}_k^T \Delta\mathbf{x}_k = \alpha_k \mathbf{g}_k^T \mathbf{p}_k < 0 \tag{9.6}$$

我们将选择较小的正数 $\alpha_k$。这就隐含

$$\mathbf{g}_k^T \mathbf{p}_k < 0 \tag{9.7}$$

**下降方向** 满足上式的任意向量称为一个下降方向(descent direction)。如果沿此方向取足够小的步长，函数一定递减。这带来了另一个问题：最速下降的方向在哪里？(即在什么方向上函数递减速度最快?) 这种情况发生于下式为最大的负数时：

$$\mathbf{g}_k^T \mathbf{p}_k \tag{9.8}$$

(设 $\mathbf{p}_k$ 长度不变，只改变方向。)这是梯度和方向向量之间的内积。当方向向量与梯度反向时该内积为负，而绝对值最大。(见 8.2.2 节关于方向导数的讨论。)所以最速下降方向的向量为

$$\mathbf{p}_k = -\mathbf{g}_k \tag{9.9}$$

**最速下降法** 在式(9.1)的迭代中使用此式得最速下降的方法：

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k \tag{9.10}$$

**学习速度** 对最速下降法，有两个用来确定学习速度 $\alpha_k$ 的常见方法。第一个方法是使基于 $\alpha_k$ 的性能指数 $F(\mathbf{x})$每次迭代最小化，即沿下列方向实现最小化：

9-3

$$\mathbf{x}_k - \alpha_k \mathbf{g}_k \tag{9.11}$$

另一个方法是选择固定的 $\alpha_k$ 值(例如取 $\alpha_k = 0.02$)，或使用预先确定的变量值(例如 $\alpha_k = 1/k$)。在下面例子中我们将详细讨论 $\alpha_k$ 的取值问题。

试给出下列函数的最速下降算法：

$$F(\mathbf{x}) = x_1^2 + 25x_2^2 \tag{9.12}$$

给定迭代初值为

$$\mathbf{x}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \tag{9.13}$$

第一步先求梯度：

$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 2x_1 \\ 50x_2 \end{bmatrix} \tag{9.14}$$

求迭代初值处的梯度

$$\mathbf{g}_0 = \nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_0} = \begin{bmatrix} 1 \\ 25 \end{bmatrix} \tag{9.15}$$

假定采用固定的学习速度 $\alpha = 0.01$。最速下降算法的第一次迭代为

$$\mathbf{x}_1 = \mathbf{x}_0 - \alpha \mathbf{g}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - 0.01\begin{bmatrix} 1 \\ 25 \end{bmatrix} = \begin{bmatrix} 0.49 \\ 0.25 \end{bmatrix} \tag{9.16}$$

第二次迭代为

$$\mathbf{x}_2 = \mathbf{x}_1 = -\alpha\mathbf{g}_1 = \begin{bmatrix} 0.49 \\ 0.25 \end{bmatrix} - 0.01\begin{bmatrix} 0.98 \\ 12.5 \end{bmatrix} = \begin{bmatrix} 0.4802 \\ 0.125 \end{bmatrix} \tag{9.17}$$

9-4

继续迭代下去可得图 9-1 所示的迭代轨迹。

注意到对于较小的学习速度最速下降轨迹的路径总是与轮廓线正交，这是因为梯度与轮廓线总是正交的。(见前面 8.2.2 节的讨论。)

如果改变学习速度，该算法的性能会如何变化？如果学习速度增加到 $\alpha = 0.035$，可得图 9-2 所示的轨迹。注意这时的轨迹是一条振荡线。可见如果学习速度太大，算法会变得不稳定，振荡不会衰减，反而会增大。

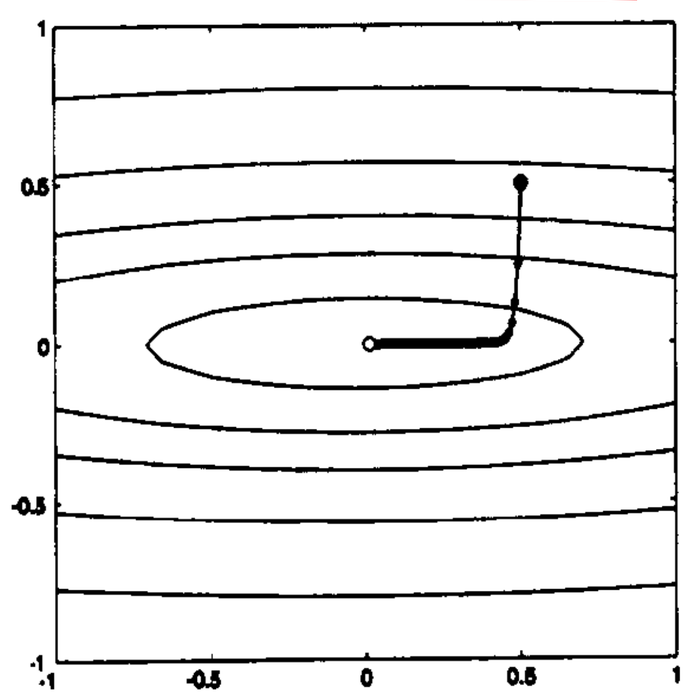



图 9-1 $\alpha = 0.01$ 时的最速下降轨迹

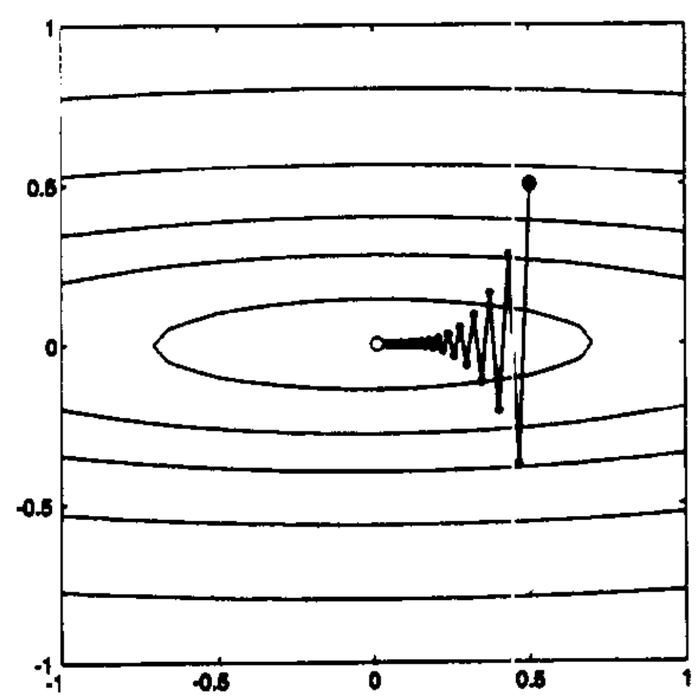



图 9-2 $\alpha = 0.035$ 时的最速下降轨迹

我们总是希望学习速度更快，所以增大步长以期快速收敛。但是，从本例中可以看出，如果学习速度太快，算法将变得不稳定。如何确定最大可行的学习速度？对于任意函数，这是不可能的，但对于二次函数，我们可以确定一个上界。

9-5

1. 稳定的学习速度

假定性能指数是一个二次函数：

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{d}^T\mathbf{x} + c \tag{9.18}$$

由式(8.38)知二次函数的梯度为

$$\nabla F(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{d} \tag{9.19}$$

将这个表达式代入最速下降算法的表达式(假定学习速度为常数)，得

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha\mathbf{g}_k = \mathbf{x}_k - \alpha(\mathbf{A}\mathbf{x}_k + \mathbf{d}) \tag{9.20}$$

或

$$\mathbf{x}_{k+1} = [\mathbf{I} - \alpha\mathbf{A}]\mathbf{x}_k - \alpha\mathbf{d} \tag{9.21}$$

这是一个线性动态系统，如果矩阵$[\mathbf{I} - \alpha\mathbf{A}]$的特征值小于 1，该系统就是稳定的(见[Brog91])。可用赫森矩阵 $\mathbf{A}$ 的特征值来表示该矩阵的特征值。设赫森矩阵的特征值和特征向量分别为$\{\lambda_1, \lambda_2, \cdots, \lambda_n\}$和$\{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n\}$。那么

$$[\mathbf{I} - \alpha\mathbf{A}]\mathbf{z}_i = \mathbf{z}_i - \alpha\mathbf{A}\mathbf{z}_i = \mathbf{z}_i - \alpha\lambda_i\mathbf{z}_i = (1 - \alpha\lambda_i)\mathbf{z}_i \tag{9.22}$$

所以$[\mathbf{I} - \alpha\mathbf{A}]$的特征向量与 $\mathbf{A}$ 的特征向量相同，特征值为$(1 - \alpha\lambda_i)$。于是最速下降算法

的稳定条件为

$$|(1-\alpha\lambda_i)| < 1 \tag{9.23}$$

如果二次函数有一个强极小点，则其特征值为正数，式(9.23)可化为

$$\alpha < \frac{2}{\lambda_i} \tag{9.24}$$

由于该式对赫森矩阵的所有的特征值都成立，所以有

$$\alpha < \frac{2}{\lambda_{max}} \tag{9.25}$$

9-6 最大的稳定学习速度与二次函数的最大的曲率成反比。曲率说明梯度变化的快慢。如果梯度变化太快，可能会导致跳过极小点，进而使新的迭代点的梯度的值大于原迭代点的梯度的值(但方向相反)。这会导致每次迭代的步长增大。

现在用这个结论来分析前面的例子。那个二次函数的赫森矩阵为

$$\mathbf{A} = \begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix} \tag{9.26}$$

**A** 的特征值和特征向量为

$$\left\{(\lambda_1 = 2), \left(\mathbf{z}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right)\right\}, \left\{(\lambda_2 = 50), \left(\mathbf{z}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right)\right\} \tag{9.27}$$

所以允许最大的学习速度为

$$\alpha < \frac{2}{\lambda_{max}} = \frac{2}{50} = 0.04 \tag{9.28}$$

图 9-3 所示为这个结果的实验，它表示学习速度略小于 0.04($\alpha = 0.039$)和略大于 0.04($\alpha = 0.041$)的最速下降轨迹。

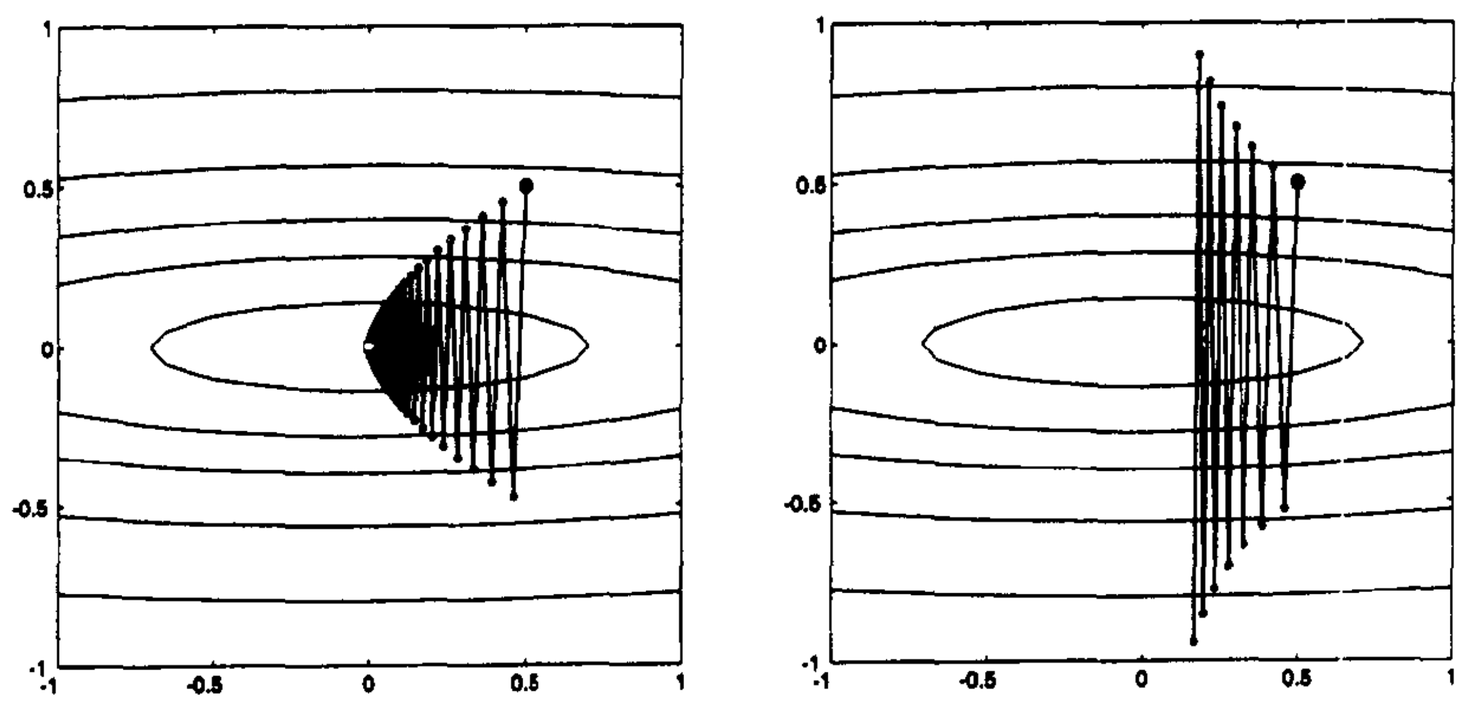



图 9-3 $\alpha = 0.039$(左)和 $\alpha = 0.041$(右)的最速下降轨迹

这个例子说明许多问题。学习速度受限于赫森矩阵的最大特征值。在最大特征值的特征向量方向上算法收敛最快，且这个方向上不能越过极小点太远。(本例中的初始迭代方向几乎与 $x_2$ 轴即 $\mathbf{z}_2$ 平行。)然而，在最小特征值的特征向量(本例中的 $\mathbf{z}_1$)方向上算法将收敛最慢。最后，最小特征值与学习速度共同决定算法收敛的快慢。特征值的大小相差越大，最速

9-7

下降算法收敛越慢。

试验二次函数的最速下降法请用 *Neural Network Design Demonstration* **Steepest Descent for a Quadratic**(**nnd9sdq**)。

2. 沿直线最小化

选择学习速度的另一种方法是用 $\alpha_k$ 使每次迭代的性能指数最小化。即选择 $\alpha_k$ 使下式最小化：

$$F(\mathbf{x}_k + \alpha_k \mathbf{p}_k) \tag{9.29}$$

对任意函数的这种最小化需要线性搜索(将在第 12 章讨论)。对二次函数解析线性最小化是可能的。式(9.29)对 $\alpha_k$ 的导数($F(\mathbf{x})$为二次函数)为

$$\frac{d}{d\alpha_k}F(\mathbf{x}_k + \alpha_k \mathbf{p}_k) = \nabla F(\mathbf{x})^T\Big|_{\mathbf{x}=\mathbf{x}_k}\mathbf{p}_k + \alpha_k \mathbf{p}_k^T \nabla^2 F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_k}\mathbf{p}_k \tag{9.30}$$

设该导数为零并求出 $\alpha_k$ 为

$$\alpha_k = -\frac{\nabla F(\mathbf{x})^T\Big|_{\mathbf{x}=\mathbf{x}_k}\mathbf{p}_k}{\mathbf{p}_k^T \nabla^2 F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_k}\mathbf{p}_k} = -\frac{\mathbf{g}_k^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A}_k \mathbf{p}_k} \tag{9.31}$$

这里 $\mathbf{A}_k$ 为在 $\mathbf{x}_k$ 的赫森矩阵：

$$\mathbf{A}_k = \nabla^2 F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_k} \tag{9.32}$$

(二次函数的赫森矩阵不是 $k$ 的函数。)

现在用沿直线最小化来实现下列二次函数的最速下降：

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}\mathbf{x} \tag{9.33}$$

迭代初值点为 9-8

$$\mathbf{x}_0 = \begin{bmatrix} 0.8 \\ -0.25 \end{bmatrix} \tag{9.34}$$

该函数的梯度为

$$\nabla F(\mathbf{x}) = \begin{bmatrix} 2x_1 + x_2 \\ x_1 + 2x_2 \end{bmatrix} \tag{9.35}$$

最速下降法的搜索方向是梯度的反向。对第一次迭代，有

$$\mathbf{p}_0 = -\mathbf{g}_0 = -\nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_0} = \begin{bmatrix} -1.35 \\ -0.3 \end{bmatrix} \tag{9.36}$$

由式(9.31)，第一次迭代的学习速度为

$$\alpha_0 = \frac{[1.35 \quad 0.3]\begin{bmatrix} -1.35 \\ -0.3 \end{bmatrix}}{[-1.35 \quad -0.3]\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} -1.35 \\ -0.3 \end{bmatrix}} = 0.413 \tag{9.37}$$

第一次迭代为

$$\mathbf{x}_1 = \mathbf{x}_0 - \alpha_0 \mathbf{g}_0 = \begin{bmatrix} 0.8 \\ -0.25 \end{bmatrix} - 0.413 \begin{bmatrix} 1.35 \\ 0.3 \end{bmatrix} = \begin{bmatrix} 0.24 \\ -0.37 \end{bmatrix} \tag{9.38}$$

图 9-4 显示该算法的前 5 次迭代。

注意：算法的逐次迭代都是正交的。为什么如此？首先，沿直线的最小化总会在轮廓线的切线上一点停止。其次，由于梯度正交于轮廓线，沿梯度相反方向的下一步就与前一步正交。

用式(9.30)的链规则(chain rule)来分析：

$$\begin{aligned} \frac{d}{d\alpha_k} F(\mathbf{x}_k + \alpha_k \mathbf{p}_k) &= \frac{d}{d\alpha_k} F(\mathbf{x}_{k+1}) = \nabla F(\mathbf{x})^T \Big|_{\mathbf{x}=\mathbf{x}_{k+1}} \frac{d}{d\alpha_k}[\mathbf{x}_k + \alpha_k \mathbf{p}_k] \\ &= \nabla F(\mathbf{x})^T \Big|_{\mathbf{x}=\mathbf{x}_{k+1}} \mathbf{p}_k = \mathbf{g}_{k+1}^T \mathbf{p}_k \end{aligned} \tag{9.39}$$

9-9

所以在极小点，该导数为零，梯度与前一步搜索方向正交。由于下一次搜索方向与梯度方向相反，后面依次进行的搜索方向都是正交的。(这个结果说明在任何方向上的最小化，哪怕未用最速下降法，极小点的梯度都与搜索方向正交。在后面关于共轭方向的讨论中还要用到这个结果。)

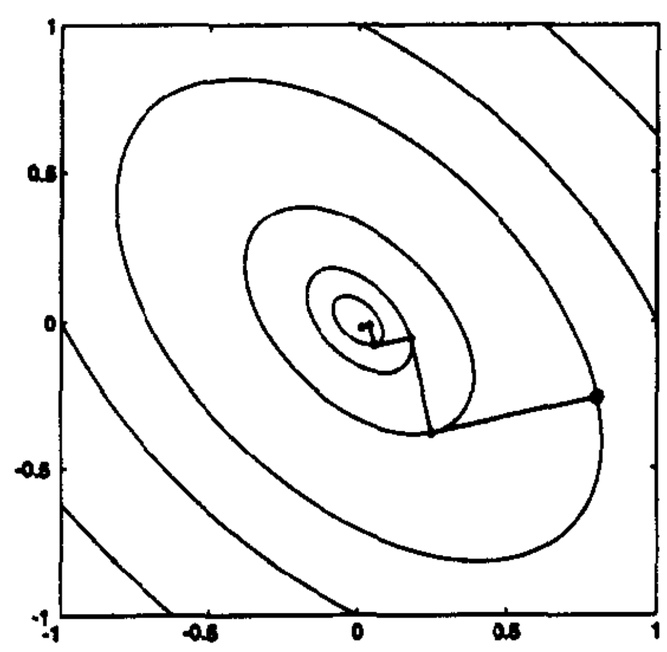



图 9-4 沿直线最小化的最速下降法

试验沿直线最小化的最速下降法请用 *Neural Network Design Demonstration Method Comparison* (**nnd9mc**)。

本章后面我们将发现调整搜索方向(用共轭代替正交)可以提高性能。如果使用共轭方向，函数最多能在 $n$ 步的迭代中被最小化($n$ 为 $\mathbf{x}$ 的维数)。(实际上存在某些类型的二次函数，用最速下降算法一步就能最小化。你能否想像出这样一个函数？其赫森矩阵的特性是什么？)

### 9.2.2 牛顿法

最速下降算法的导数是以一阶泰勒级数展开为基础的(式(9.4))。牛顿法则基于二阶泰勒级数：

$$F(\mathbf{x}_{k+1}) = F(\mathbf{x}_k + \Delta\mathbf{x}_k) \approx F(\mathbf{x}_k) + \mathbf{g}_k^T \Delta\mathbf{x}_k + \frac{1}{2} \Delta\mathbf{x}_k^T \mathbf{A}_k \Delta\mathbf{x}_k \tag{9.40}$$

**牛顿法** 牛顿法的原理是求 $F(\mathbf{x})$ 的二次近似的驻点。用式(8.38)求这个二次函数对 $\Delta\mathbf{x}_k$ 的梯度并设它为零，则有 9-10

$$\mathbf{g}_k + \mathbf{A}_k\Delta\mathbf{x}_k = \mathbf{0} \tag{9.41}$$

求解 $\Delta\mathbf{x}_k$ 得

$$\Delta\mathbf{x}_k = -\mathbf{A}_k^{-1}\mathbf{g}_k \tag{9.42}$$

于是可将牛顿法定义为

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1}\mathbf{g}_k \tag{9.43}$$

为了说明牛顿法的步骤，将它用于前面式(9.12)的例子：

$$F(\mathbf{x}) = x_1^2 + 25x_2^2 \tag{9.44}$$

其梯度和赫森矩阵为

$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1}F(\mathbf{x}) \\ \frac{\partial}{\partial x_2}F(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 2x_1 \\ 50x_2 \end{bmatrix}, \nabla^2 F(\mathbf{x}) = \begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix} \tag{9.45}$$

如果从同一个初始点

$$\mathbf{x}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \tag{9.46}$$

开始，牛顿法的第一步为

$$\mathbf{x}_1 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - \begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix}^{-1}\begin{bmatrix} 1 \\ 25 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{9.47}$$

这个方法总能一步找到二次函数的极小点。因为牛顿法总是用一个二次函数逼近 $F(x)$，然后求其驻点。如果原函数为二次函数(有强极小点)，它就能够实现一步极小化。图 9-5 所示为这个问题的牛顿法的迭代轨迹。

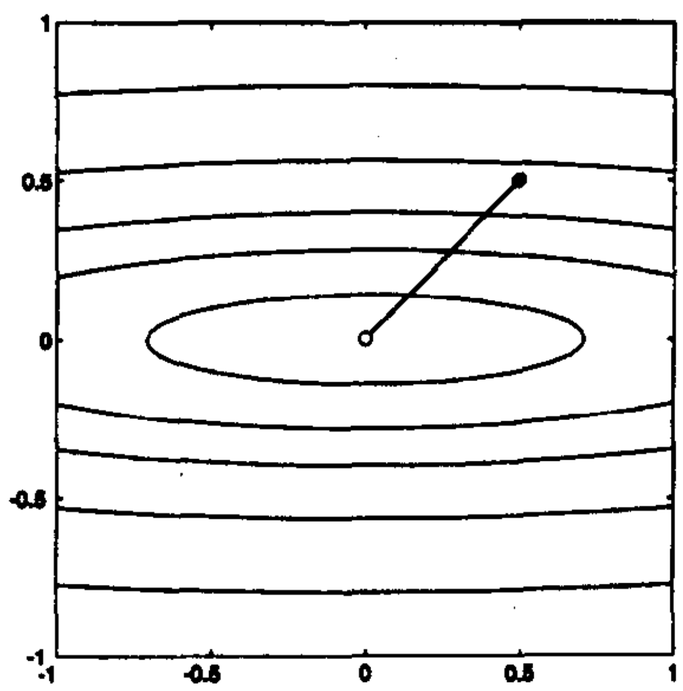



图 9-5 牛顿法的轨迹

如果函数 $F(\mathbf{x})$ 不是二次函数，则牛顿法一般不能在一步内收敛。实际上根本无法确定它是否收敛，因为这取决于具体的函数和初始点。 9-11

回忆式(8.8)的函数：

$$F(\mathbf{x}) = (x_2 - x_1)^4 + 8x_1x_2 - x_1 + x_2 + 3 \tag{9.48}$$

由第 8 章知(见例题 P8.5)该函数有 3 个驻点：

$$\mathbf{x}^1 = \begin{bmatrix} -0.42 \\ 0.42 \end{bmatrix}, \quad \mathbf{x}^2 = \begin{bmatrix} -0.13 \\ 0.13 \end{bmatrix}, \quad \mathbf{x}^3 = \begin{bmatrix} 0.55 \\ -0.55 \end{bmatrix} \tag{9.49}$$

第一点是一个强局部极小点，第二点是一个鞍点，第三点是一个强全局极小点。

将牛顿法用于这个问题，初始点为 $\mathbf{x}_0 = [1.5 \quad 0]^T$，第一次迭代如图 9-6 所示。左边的图是原函数的轮廓线图，右边的图是该函数在初始点的二次近似。

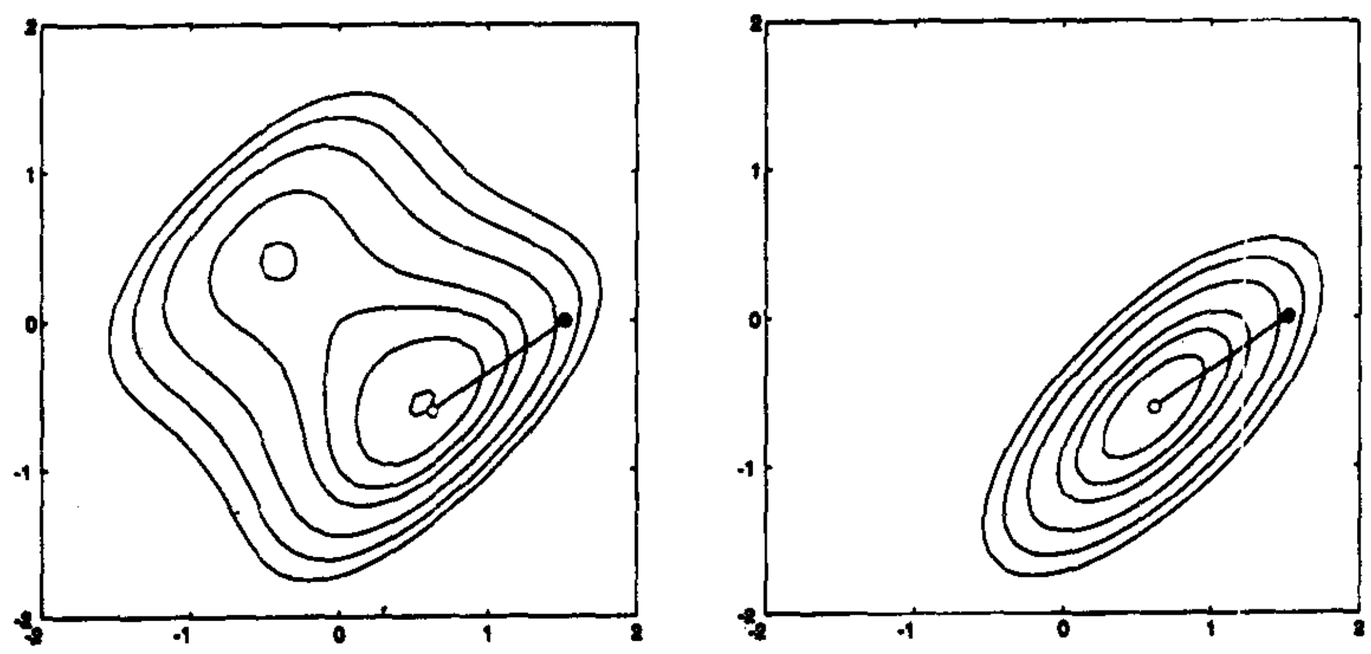


图 9-6 牛顿法从 $\mathbf{x}_0 = [-1.5 \quad 0]^T$ 的一次迭代

该函数不能实现一步最小化，因为这不是二次函数。然而，迭代是朝全局极小方向进行的，如果再迭代两次，算法就能收敛到全局极小点的 0.01 的范围之内。牛顿法在许多应用中都能快速收敛。这是因为在一个强极小点的较小的邻域内，解析函数能够被二次函数精确近似。离极小点越近，牛顿法越能精确标识该极小点。从本例中可以发现在初始点附近，二次近似的轮廓线图同原函数轮廓线图很相似。

9-12

图 9-7 所示为以 $\mathbf{x}_0 = [-1.5 \quad 0]^T$ 为初始点的牛顿一次迭代。本例中收敛到局部极小点。显然，牛顿法不能区别局部极小和全局极小，因为它将函数近似为二次函数，而二次函数只有一个极小点。同最速下降法一样，牛顿法也依赖于曲面的特征(一阶和二阶导数)。它无法弄清函数的全局特征。

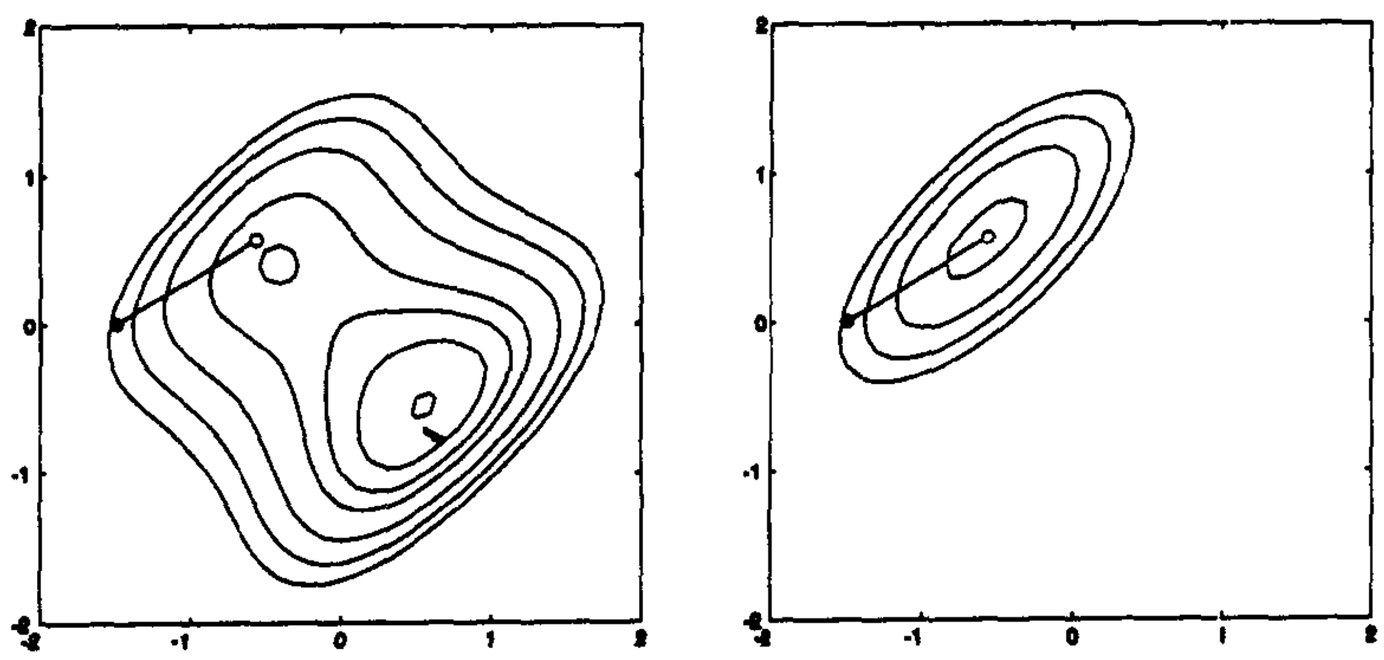


图 9-7 从 $\mathbf{x}_0 = [-1.5 \quad 0]^T$ 的牛顿法的一次迭代

图 9-8 所示为以 $\mathbf{x}_0=[0.75 \quad 0.75]^T$ 为初始点的牛顿法的一次迭代。这次收敛到了函数的鞍点。注意牛顿法是在当前初始点确定函数的二次近似的驻点，它并不区别极小点、极大点和鞍点。本例中的二次近似有一个鞍点(不定型赫森矩阵)，在原函数鞍点附近。如果继续迭代，算法就会收敛到 $F(\mathbf{x})$的鞍点。 9-13

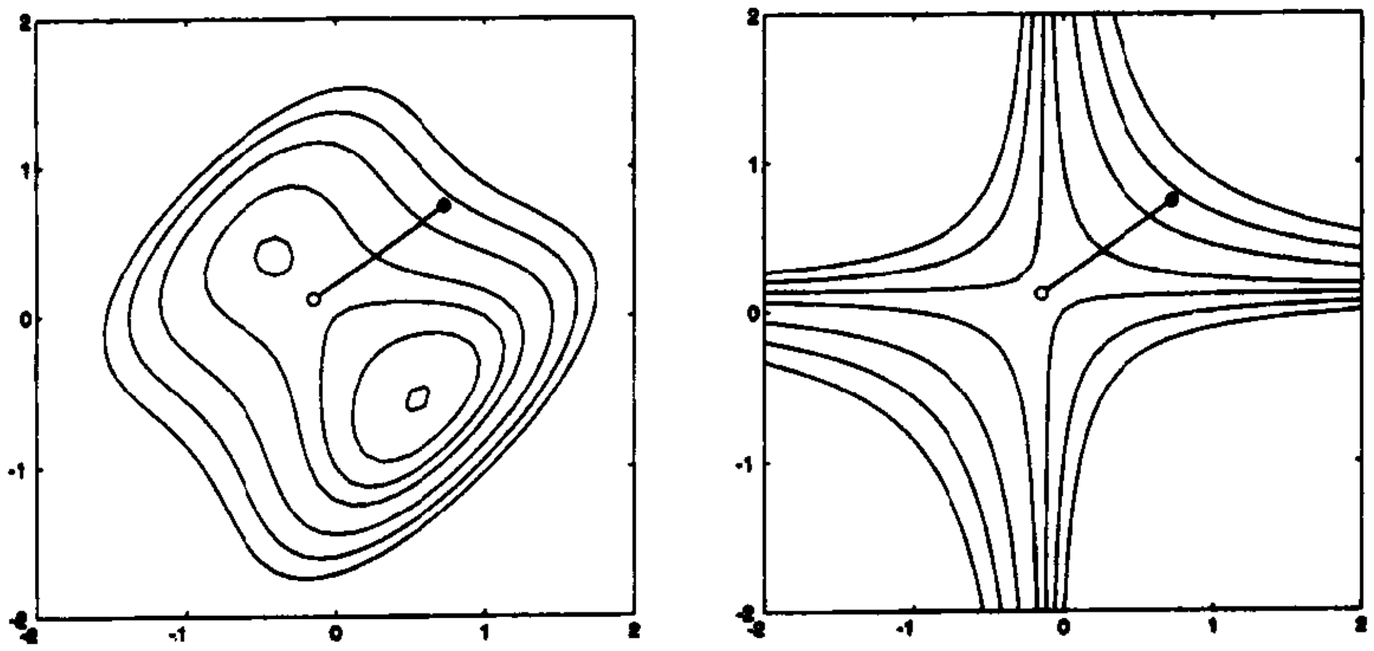


图 9-8 以 $\mathbf{x}_0=[0.75 \quad 0.75]^T$ 为初始点的牛顿法一次迭代

以上各例中二次近似的驻点总在 $F(\mathbf{x})$相应的驻点附近。实际情况并不总是这样。实际上，牛顿法可以产生难以预料的结果。

图 9-9 所示为 $\mathbf{x}_0=[1.15 \quad 0.75]^T$ 为初始点的牛顿法一次迭代。这里，二次近似预期会有一个鞍点，但是鞍点离 $F(\mathbf{x})$的局部极小点很近。如果连续迭代下去，算法将收敛到局部极小点。注意这里初始点离该局部极小点比上例中更远，而上例中却收敛到鞍点。

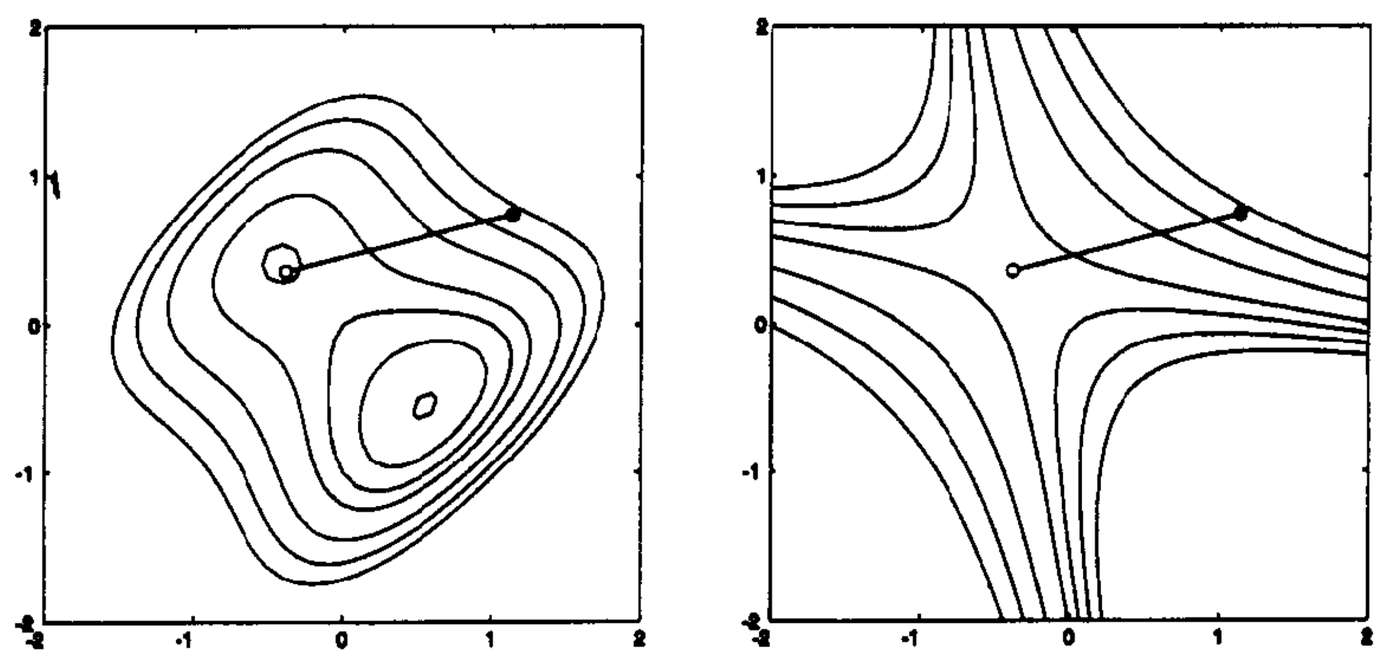


图 9-9 以 $\mathbf{x}_0=[1.15 \quad 0.75]^T$ 为初始点的牛顿法一次迭代 9-14

试验该函数的牛顿法和最速下降法请用 *Neural Network Design Demonstration Newton's Method* (**nnd9nm**)和 *Steepest Descent* (**nnd9sd**)。

牛顿法的特点总结如下：

尽管牛顿法的收敛速度通常比最速下降法更快，但其表现很复杂，除了收敛到鞍点的问题(同最速下降法不同)外，算法还可能振荡和发散。如果学习速度不太快或每步都实现线性极小化，最速下降法能够确保收敛。

第 12 章将讨论适合网络训练的牛顿法的一种变型，它能够解决当发散开始出现时最速下降法的发散问题。

牛顿法的另一个问题是需要对赫森矩阵及其逆阵的计算和存储。将最速下降法的式(9.10)与牛顿法的式(9.43)相比，可以发现当下式成立时，它们的搜索方向将相同：

$$\mathbf{A}_k = \mathbf{A}_k^{-1} = \mathbf{I} \tag{9.50}$$

由此可以导出称之为类牛顿法或单步正割法的一类优化算法。这类方法用一个正定矩阵 $\mathbf{H}_k$ 代替 $\mathbf{A}_k$，该矩阵不需转置，每次迭代都刷新。这类算法通常能使二次函数 $\mathbf{H}_k$ 收敛于 $\mathbf{A}^{-1}$。(二次函数的赫森矩阵为一常数矩阵。)有关这类方法的讨论见[Gill81]、[Scal85]或[Batl92]。

### 9.2.3 共轭梯度法

**二次终结法** 牛顿法有一个性质称为二次终结法(quadratic termination)，即它能在有限的迭代次数内使二次函数极小化。但这需要计算和存储二阶导数。当参数个数 $n$ 很大时，计算所有二阶导数是很困难的。(若梯度有 $n$ 个元素，则赫森矩阵有 $n^2$ 个元素)。在神经网络中这个问题尤其严重，因为这里的实际应用往往需要几百个甚至上千个权值。所以我们希望找到只需要一阶导数但是仍具有二次终结性质的方法。

9-15 回忆最速下降法在每次迭代用线性搜索时的性能。相继迭代的搜索方向相互正交(见图 9-4)。对于轮廓线为椭圆的二次函数，这将产生短步长的锯齿形轨迹。也许二次搜索方向并非最好的选择。那么存在一个确保二次终结法的搜索方向的集合吗？一个可能便是共轭方向。

假定对下述二次函数确定极小点：

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{d}^T\mathbf{x} + c \tag{9.51}$$

**共轭** 当且仅当

$$\mathbf{p}_k^T\mathbf{A}\mathbf{p}_j = 0, k \neq j \tag{9.52}$$

时，称向量集合$\{\mathbf{p}_k\}$对于一个正定赫森矩阵 $\mathbf{A}$ 两两共轭。对于正交向量，存在无穷个能张成一个 $n$ 维空间的两两共轭向量集。由 $\mathbf{A}$ 的特征向量组成的共轭向量集也是其中之一。设$\{\lambda_1, \lambda_2, \cdots, \lambda_n\}$和$\{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n\}$分别为赫森矩阵的特征值和特征向量。为了验证特征向量是共轭的，用 $\mathbf{z}_k$ 代替式(9.52)的 $\mathbf{p}_k$，有

$$\mathbf{z}_k^T\mathbf{A}\mathbf{z}_j = \lambda_j\mathbf{z}_k^T\mathbf{z}_j = 0, \quad k \neq j \tag{9.53}$$

后一等式成立是因为对称矩阵的特征向量两两正交。所以特征向量既是共轭的也是正交的。(你能否找出所有的正交向量都共轭的二次函数?)

沿赫森矩阵的特征向量搜索就能准确地使二次函数极小化。这点并不奇怪，因为特征向量构成函数轮廓线的主轴。(参见 8.2.5 节“赫森的特征系统”中的讨论。)然而这对于实际运用没有多少帮助，因为要知道特征向量必须先求出赫森矩阵。我们希望找到一种不需要计算二阶导数的算法。

已经证明(见[Scal85]或[Gill81])，如果存在沿一个共轭方向集$\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_n\}$的准确线性搜索序列，就能在最多 $n$ 次搜索内实现具有 $n$ 个参数的二次函数的准确极小化。问题在于如何构造这些共轭搜索方向。首先来看式(9.52)中不用赫森矩阵的共轭条件。注意到对

于二次函数，有

$$\nabla F(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{d} \tag{9.54}$$ 9-16

$$\nabla^2 F(\mathbf{x}) = \mathbf{A} \tag{9.55}$$

将这些等式组合起来，能发现在 $k+1$ 次迭代时梯度的变化：

$$\Delta\mathbf{g}_k = \mathbf{g}_{k+1} - \mathbf{g}_k = (\mathbf{A}\mathbf{x}_{k+1} + \mathbf{d}) - (\mathbf{A}\mathbf{x}_k + \mathbf{d}) = \mathbf{A}\Delta\mathbf{x}_k \tag{9.56}$$

又由式(9.2)有

$$\Delta\mathbf{x}_k = (\mathbf{x}_{k+1} - \mathbf{x}_k) = \alpha_k\mathbf{p}_k \tag{9.57}$$

选择 $\alpha_k$ 使函数 $F(\mathbf{x})$ 在 $\mathbf{p}_k$ 方向上极小化。

所以式(9.52)的共轭条件可重写成

$$\alpha_k\mathbf{p}_k^T\mathbf{A}\mathbf{p}_j = \Delta\mathbf{x}_k^T\mathbf{A}\mathbf{p}_j = \Delta\mathbf{g}_k^T\mathbf{p}_j = 0, k \neq j \tag{9.58}$$

注意现在不需要求赫森矩阵了。我们已经把共轭条件表示成算法相继迭代的梯度变化的形式。如果搜索方向与梯度变化方向垂直，则它们共轭。

注意第一次搜索方向 $\mathbf{p}_0$ 是任意的，而 $\mathbf{p}_1$ 可以是与 $\Delta\mathbf{g}_0$ 垂直的任意向量。所以共轭向量集的数量是无限的。通常从最速下降法的方向开始搜索：

$$\mathbf{p}_0 = -\mathbf{g}_0 \tag{9.59}$$

每次迭代都要构选一个与 $\{\Delta\mathbf{g}_0, \Delta\mathbf{g}_1, \cdots, \Delta\mathbf{g}_{k-1}\}$ 正交的向量 $\mathbf{p}_k$。这与第 5 章讨论的 Gram-Schmidt 正交化过程类似。可将迭代形式简化为(见[Scal85])：

$$\mathbf{p}_k = -\mathbf{g}_k + \beta_k\mathbf{p}_{k-1} \tag{9.60}$$

确定系数 $\beta_k$ 的方法有许多种，对二次函数产生的结果相同。通常选择(见[Scal85])：

$$\beta_k = \frac{\Delta\mathbf{g}_{k-1}^T\mathbf{g}_k}{\Delta\mathbf{g}_{k-1}^T\mathbf{p}_{k-1}} \tag{9.61}$$

(由 Hestenes 和 Steifel 确定) 9-17

$$\beta_k = \frac{\mathbf{g}_k^T\mathbf{g}_k}{\mathbf{g}_{k-1}^T\mathbf{g}_{k-1}} \tag{9.62}$$

(由 Fletcher 和 Reeves 确定)

$$\beta_k = \frac{\Delta\mathbf{g}_{k-1}^T\mathbf{g}_k}{\mathbf{g}_{k-1}^T\mathbf{g}_{k-1}} \tag{9.63}$$

(由 Polak 和 Ribiére 确定)

**共轭梯度** 上述关于共轭梯度的讨论可归纳如下：

1) 选择如式(9.59)所示的与梯度相反的方向作为第一次搜索方向。

2) 根据式(9.57)进行下一步搜索，确定 $\alpha_k$ 以使函数沿搜索方向极小化。

第 12 章将讨论通用的线性极小化技术。对于二次函数，可使用式(9.31)。

3) 根据式(9.60)确定下一个搜索方向，用式(9.61)，(9.62)或(9.63)式计算 $\beta_k$。

4) 如果算法不收敛，回到第 2 步。

为了说明这个算法的性能，再使用前面用于说明线性极小化的最速下降法的例子：

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}\mathbf{x} \tag{9.64}$$

初始点为

$$\mathbf{x}_0 = \begin{bmatrix} 0.8 \\ -0.25 \end{bmatrix} \tag{9.65}$$

该函数的梯度为

$$\nabla F(\mathbf{x}) = \begin{bmatrix} 2x_1 + x_2 \\ x_1 + 2x_2 \end{bmatrix} \tag{9.66}$$

9-18 对于最速下降法，第一次搜索方向与梯度相反：

$$\mathbf{p}_0 = -\mathbf{g}_0 = -\nabla F(\mathbf{x})^T\Big|_{\mathbf{x}=\mathbf{x}_0} = \begin{bmatrix} -1.35 \\ -0.3 \end{bmatrix} \tag{9.67}$$

由式(9.31)，第一次迭代的学习速度为

$$\alpha_0 = \frac{[1.35 \quad 0.3]\begin{bmatrix} -1.35 \\ -0.3 \end{bmatrix}}{[-1.35 \quad -0.3]\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} -1.35 \\ -0.3 \end{bmatrix}} = -0.413 \tag{9.68}$$

所以共轭梯度法的第一步为

$$\mathbf{x}_1 = \mathbf{x}_0 + \alpha_0\mathbf{p}_0 = \begin{bmatrix} 0.8 \\ -0.25 \end{bmatrix} + 0.413\begin{bmatrix} -1.35 \\ -0.3 \end{bmatrix} = \begin{bmatrix} 0.24 \\ -0.37 \end{bmatrix} \tag{9.69}$$

这个结果与沿直线的最速下降极小化的结果相同。

现在用式(9.60)找第二次搜索方向。先求出在 $\mathbf{x}_1$ 的梯度：

$$\mathbf{g}_1 = \nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_1} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} 0.24 \\ -0.37 \end{bmatrix} = \begin{bmatrix} 0.11 \\ -0.5 \end{bmatrix} \tag{9.70}$$

现在求 $\beta_1$：

$$\beta_1 = \frac{\mathbf{g}_1^T\mathbf{g}_1}{\mathbf{g}_0^T\mathbf{g}_0} = \frac{[0.11 \quad -0.5]\begin{bmatrix} 0.11 \\ -0.5 \end{bmatrix}}{[1.35 \quad 0.3]\begin{bmatrix} 1.35 \\ 0.3 \end{bmatrix}} = \frac{0.2621}{1.9125} = 0.137 \tag{9.71}$$

这是式(9.62)的 Fletcher 和 Reeves 的方法。于是由式(9.60)得第二次搜索方向：

$$\mathbf{p}_1 = -\mathbf{g}_1 + \beta_1\mathbf{p}_0 = \begin{bmatrix} -0.11 \\ 0.5 \end{bmatrix} + 0.137\begin{bmatrix} -1.35 \\ -0.3 \end{bmatrix} = \begin{bmatrix} -0.295 \\ 0.459 \end{bmatrix} \tag{9.72}$$

9-19 由式(9.31)，第二次迭代的学习速度为

$$\alpha_1 = \frac{[0.11 \quad -0.5]\begin{bmatrix} -0.295 \\ 0.459 \end{bmatrix}}{[-0.295 \quad 0.459]\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} -0.295 \\ 0.459 \end{bmatrix}} = \frac{0.262}{0.325} = 0.807 \tag{9.73}$$

因此共轭梯度法的第二步为

$$\mathbf{x}_2 = \mathbf{x}_1 + \alpha_1\mathbf{p}_1 = \begin{bmatrix} 0.24 \\ -0.37 \end{bmatrix} + 0.807\begin{bmatrix} -0.295 \\ 0.459 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{9.74}$$

该算法和预期的一样，两次迭代就精确收敛到极小点(因为这是一个二维的二次函数)，如图 9-10 所示。把这个结果与图 9-4 的最速下降算法的结果相比较可知，同最速下降法使用正交的搜索方向的方法不同，共轭梯度算法调节第二次搜索方向以使它通过函数极小点(函数轮廓线的中心)。

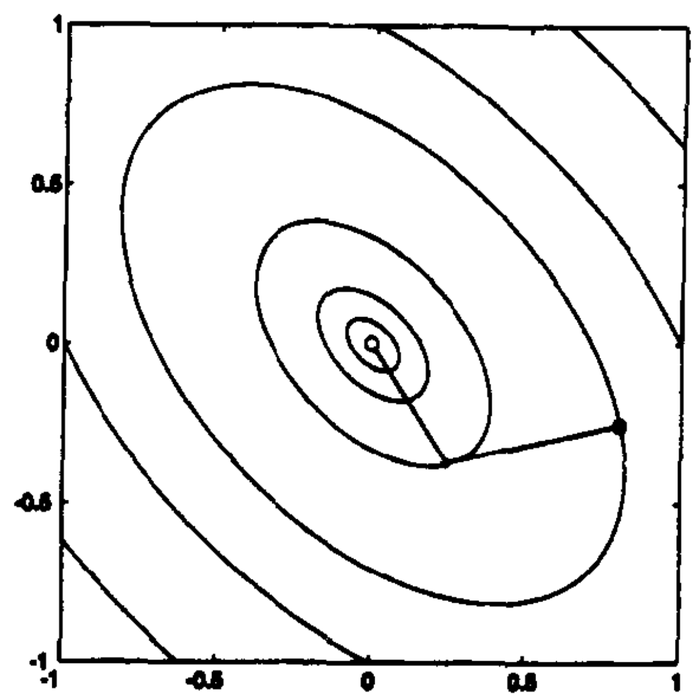



图 9-10 共轭梯度算法

第 12 章还会讨论共轭梯度算法用于非二次函数的情况。

试验共轭梯度算法与最速下降法的比较请用 *Neural Network Design Demonstration Method Comparison* (**nnd9mc**)。 9-20

## 9.3 小结

**通用最小化算法**

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$$

或

$$\Delta\mathbf{x}_k = (\mathbf{x}_{k+1} - \mathbf{x}_k) = \alpha_k \mathbf{p}_k$$

**最速下降算法**

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k$$

其中 $\mathbf{g}_k \equiv \nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_k}$

**稳定学习速度** ($\alpha_k = \alpha$，**常数**)

$$\alpha < \frac{2}{\lambda_{max}}$$

$\{\lambda_1, \lambda_2, \cdots, \lambda_n\}$为赫森矩阵 $\mathbf{A}$ 的特征值

**沿直线$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$ 的最小化的学习速度**

$$\alpha_k = -\frac{\mathbf{g}_k^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k} \text{(用于二次函数)}$$

**沿直线$\mathbf{x}_{k+1}=\mathbf{x}_k+\alpha_k\mathbf{p}_k$的最小化之后**

$$\mathbf{g}_{k+1}^T\mathbf{p}_k = 0$$

**牛顿法**

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1}\mathbf{g}_k$$

其中

9-21

$$\mathbf{A}_k \equiv \nabla^2 F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_k}$$

**共轭梯度算法**

$$\Delta\mathbf{x}_k = \alpha_k\mathbf{p}_k$$

沿直线 $\mathbf{x}_{k+1}=\mathbf{x}_k+\alpha_k\mathbf{p}_k$ 的最小化确定学习速度 $\alpha_k$

$$\mathbf{p}_0 = -\mathbf{g}_0$$

$$\mathbf{p}_k = -\mathbf{g}_k + \beta_k\mathbf{p}_{k-1}$$

$$\beta_k = \frac{\Delta\mathbf{g}_{k-1}^T\mathbf{g}_k}{\Delta\mathbf{g}_{k-1}^T\mathbf{p}_{k-1}} \text{ 或 } \beta_k = \frac{\mathbf{g}_{k-1}^T\mathbf{g}_k}{\mathbf{g}_{k-1}^T\mathbf{p}_{k-1}} \text{ 或 } \beta_k = \frac{\Delta\mathbf{g}_{k-1}^T\mathbf{g}_k}{\mathbf{g}_{k-1}^T\mathbf{p}_{k-1}}$$

9-22

其中 $\mathbf{g}_k \equiv \nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_k}$ 且$\nabla\mathbf{g}_k=\mathbf{g}_{k+1}-\mathbf{g}_k$

## 9.4 例题

**P9.1** 求下列函数的极小点：

$$F(\mathbf{x}) = 5x_1^2 - 6x_1x_2 + 5x_2^2 + 4x_1 + 4x_2$$

（i）画出该函数的轮廓线图。

（ii）设学习速度很小，起始点为 $\mathbf{x}_0=[-1 \quad -2.5]^T$，画出(i)中轮廓线的最速下降算法的轨迹。

（iii）最大的稳定学习速度是多少？

**解**

（i）要画出轮廓线图必须先求出赫森矩阵。对于二次函数，只要将函数化成标准形式(见(8.35))就能得到赫森矩阵：

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{d}^T\mathbf{x} + c = \frac{1}{2}\mathbf{x}^T\begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix}\mathbf{x} + [4 \quad 4]\mathbf{x}$$

由式(8.39)赫森矩阵为

$$\nabla^2 F(\mathbf{x}) = \mathbf{A} = \begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix}$$

该矩阵的特征值和特征向量为

$$\lambda_1 = 4, \quad \mathbf{z}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \lambda_2 = 16, \quad \mathbf{z}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

由第8章(见8.2.5节"赫森的特征系统")关于二次函数的讨论知该函数轮廓线是椭圆。由于 $\lambda_2$ 大于 $\lambda_1$，所以 $F(\mathbf{x})$ 的最大曲率在 $\mathbf{z}_2$ 方向上。其最小曲率在 $\mathbf{z}_1$ 方向上(椭圆的长轴)。

下面求轮廓线的中心(驻点)，即使梯度为零的点。由式(8.38)有

$$\nabla F(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{d} = \begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix}\mathbf{x} + \begin{bmatrix} 4 \\ 4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

9-23

故有

$$\mathbf{x}^* = -\begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix}^{-1}\begin{bmatrix} 4 \\ 4 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

轮廓线是椭圆，中心点为 $\mathbf{x}^*$，长轴在 $\mathbf{z}_1$ 方向。轮廓线图如图9-11所示。

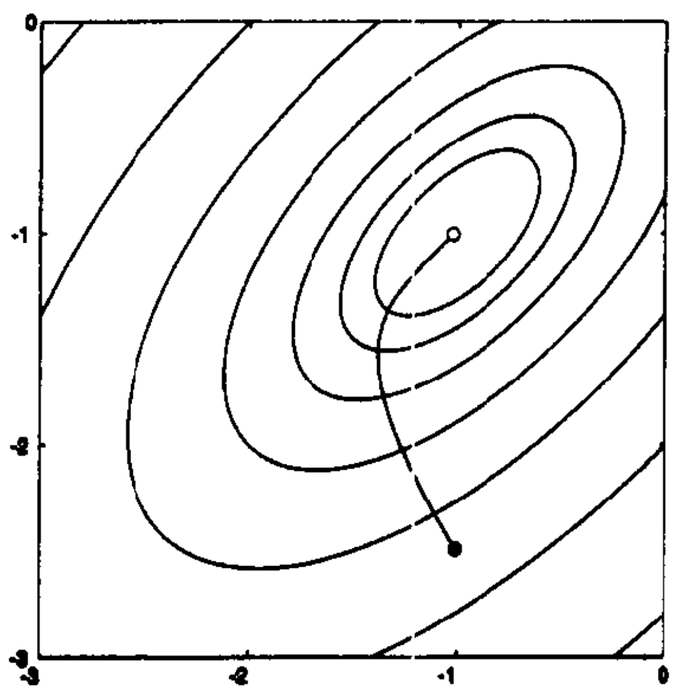


图9-11 例题9.1的轮廓线图及最速下降轨迹

(ii) 梯度总是与轮廓线相垂直，如果步长足够小，最速下降轨迹将与每条相交的轮廓线垂直。所以不需任何计算就可画出这一轨迹，如图9-11所示。

(iii) 由式(9.25)知赫森矩阵的最大特征值决定了二次函数的最大的稳定学习速度：

$$\alpha < \frac{2}{\lambda_{max}}$$

本例的最大特征值为 $\lambda_2 = 16$，所以

$$\alpha < \frac{2}{16} = 0.125$$

图9-12所示验证了这一结果，图中分别画出了学习速度略低于($\alpha = 0.12$)和略大于($\alpha = 0.13$)最大稳定学习速度时的最速下降轨迹。

9-24

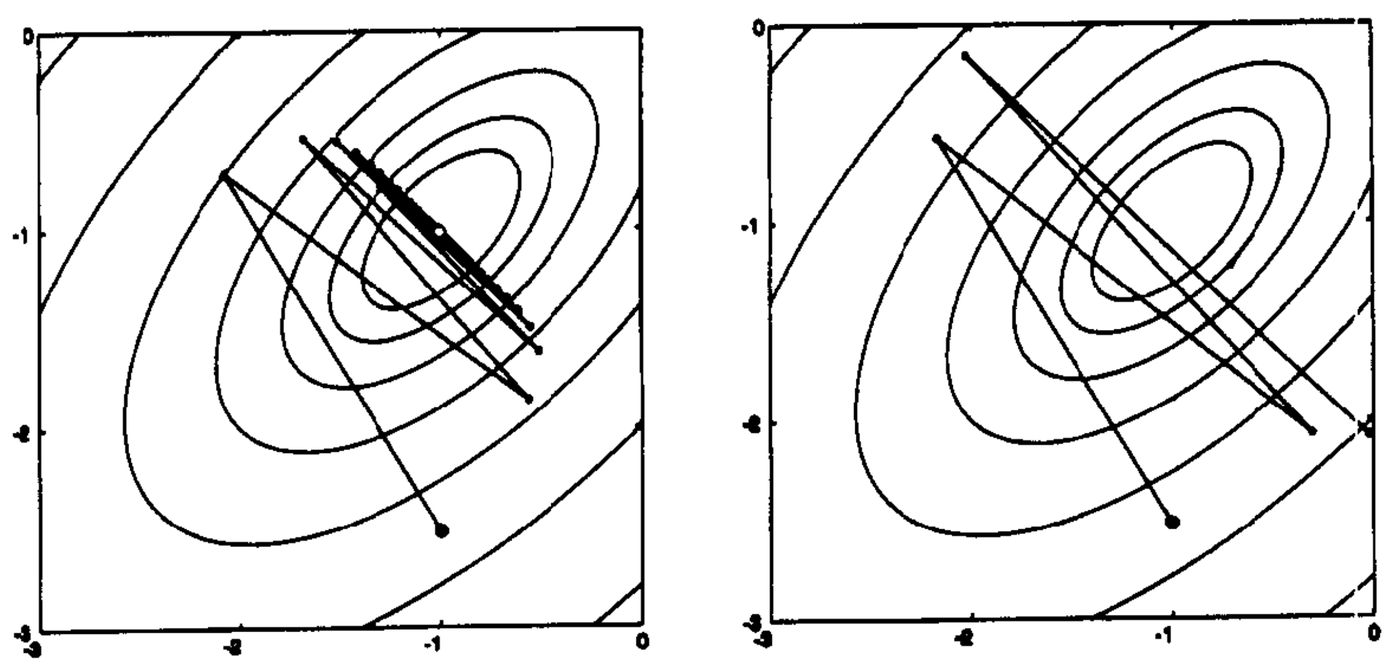


图9-12 $\alpha = 0.12$(左)和 $\alpha = 0.13$(右)的轨迹

**P9.2** 采用沿直线最小化的两步最速下降算法处理例题P9.1中的二次函数。起始条件为：

$$\mathbf{x}_0 = [0 \quad -2]^T$$

**解**

由例题P9.1知函数的梯度为

$$\nabla F(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{d} = \begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix}\mathbf{x} + \begin{bmatrix} 4 \\ 4 \end{bmatrix}$$

在 $\mathbf{x}_0$ 点计算函数的梯度，有

$$\mathbf{g}_0 = \nabla F(\mathbf{x}_0) = \mathbf{A}\mathbf{x}_0 + \mathbf{d} = \begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix}\begin{bmatrix} 0 \\ -2 \end{bmatrix} + \begin{bmatrix} 4 \\ 4 \end{bmatrix} = \begin{bmatrix} 16 \\ -16 \end{bmatrix}$$

所以第一次搜索方向是

$$\mathbf{p}_0 = -\mathbf{g}_0 = \begin{bmatrix} -16 \\ 16 \end{bmatrix}$$

二次函数沿直线的最小化可用式(9.31)：

$$\alpha_0 = \frac{\mathbf{g}_0^T\mathbf{p}_0}{\mathbf{p}_0^T\mathbf{A}\mathbf{p}_0} = \frac{[16 \quad -16]\begin{bmatrix} -16 \\ 16 \end{bmatrix}}{[-16 \quad 16]\begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix}\begin{bmatrix} -16 \\ 16 \end{bmatrix}} = -\frac{-512}{8192} = 0.0625$$

9-25 所以最速下降法的第一次迭代为

$$\mathbf{x}_1 = \mathbf{x}_0 - \alpha_0\mathbf{g}_0 = \begin{bmatrix} 0 \\ -2 \end{bmatrix} - 0.0625\begin{bmatrix} 16 \\ -16 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

第二次迭代要先求 $\mathbf{x}_1$ 点的梯度：

$$\mathbf{g}_1 = \nabla F(\mathbf{x}_1) = \mathbf{A}\mathbf{x}_1 + \mathbf{d} = \begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix}\begin{bmatrix} -1 \\ -1 \end{bmatrix} + \begin{bmatrix} 4 \\ 4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

所以已到达一个驻点；算法也已经收敛。由例题 P9.1 知该驻点的确是这个二次函数的极小点。图 9-13 所示为下降轨迹。

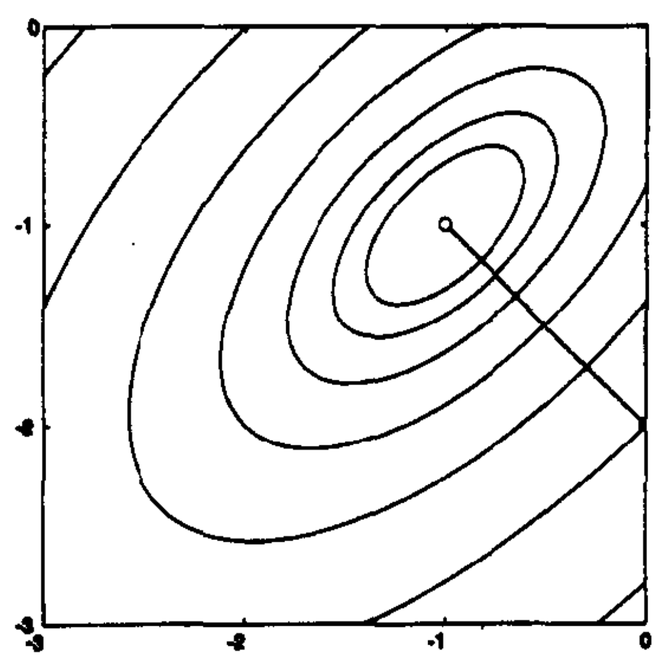



图 9-13 例题 P9.2 采用线性极小化的最速下降法

这是一个最速下降算法一次迭代到极小点的特例。注意：这里的初始点位于赫森矩阵指向极小点的特征向量的方向上。只要搜索方向在特征向量方向上，最速下降法就能一次迭代到极小点。这对赫森矩阵的特征值意味着什么呢？

**P9.3** 我们在例题 P8.6 推出了一个线性神经网络的性能指数。图 9-14 给出该网络的结构图，网络用下述输入/输出对进行训练：

$$\{(p_1 = 2),(t_1 = 0.5)\},\{(p_2 = -1),(t_2 = 0)\}$$

9-26 网络性能指数定义为

$$F(\mathbf{x}) = (t_1 - \alpha_1(\mathbf{x}))^2 + (t_2 - \alpha_2(\mathbf{x}))^2$$

如图 8-18 所示。

（i）设初始点为 $\mathbf{x}_0 = [1 \quad 1]^T$，学习速度 $\alpha = 0.05$。用最速下降法求该网络的最优参数（$\mathbf{x} = [w \quad b]^T$）。

（ii）最大的稳定学习速度是多少？

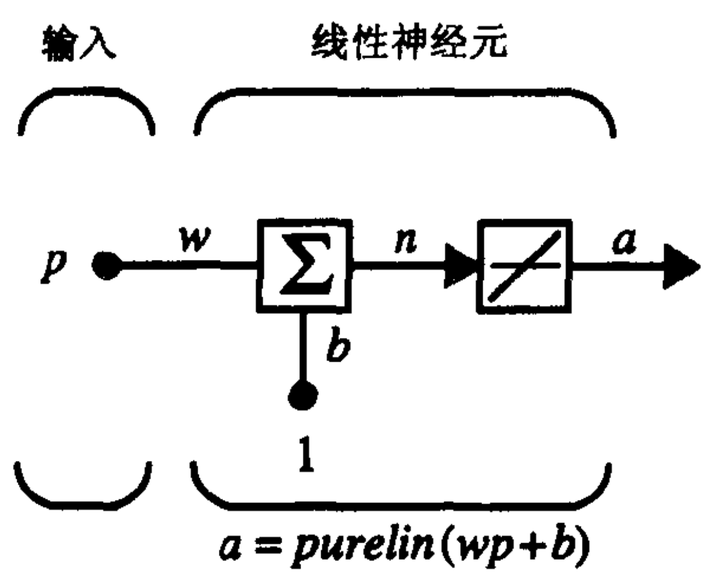



图 9-14 例题 P9.3 和 P8.6 的线性网络

**解**

（i）由例题 P8.6 知该性能指数可以写成二次形式：

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{d}^T\mathbf{x} + c$$

这里

$$c = \mathbf{t}^T\mathbf{t} = [0.5 \quad 0]\begin{bmatrix} 0.5 \\ 0 \end{bmatrix} = 0.25$$

$$\mathbf{d} = -2\mathbf{G}^T\mathbf{t} = -2\begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 0.5 \\ 0 \end{bmatrix} = \begin{bmatrix} -2 \\ -1 \end{bmatrix}$$

$$\mathbf{A} = 2\mathbf{G}^T\mathbf{G} = \begin{bmatrix} 10 & 2 \\ 2 & 4 \end{bmatrix}$$

9-27

$\mathbf{x}_0$ 点的梯度为

$$\mathbf{g}_0 = \nabla F(\mathbf{x}_0) = \mathbf{A}\mathbf{x}_0 + \mathbf{d} = \begin{bmatrix} 10 & 2 \\ 2 & 4 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} -2 \\ -1 \end{bmatrix} = \begin{bmatrix} 10 \\ 5 \end{bmatrix}$$

最速下降法的第一次迭代为

$$\mathbf{x}_1 = \mathbf{x}_0 - \alpha\mathbf{g}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} - 0.05\begin{bmatrix} 10 \\ 5 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.75 \end{bmatrix}$$

第二次迭代为

$$\mathbf{x}_2 = \mathbf{x}_1 - \alpha\mathbf{g}_1 = \begin{bmatrix} 0.7 \\ 0.75 \end{bmatrix} - 0.05\begin{bmatrix} 4.5 \\ 3 \end{bmatrix} = \begin{bmatrix} 0.275 \\ 0.6 \end{bmatrix}$$

后面的迭代见图 9-15。算法收敛于极小点 $\mathbf{x}^* = [0.167 \quad 0.167]^T$。所以该网络的权值和偏置值的最优值都是 0.167。

要训练该网络必须知道所有的输入/输出对，然后进行最速下降算法的迭代直到达到收

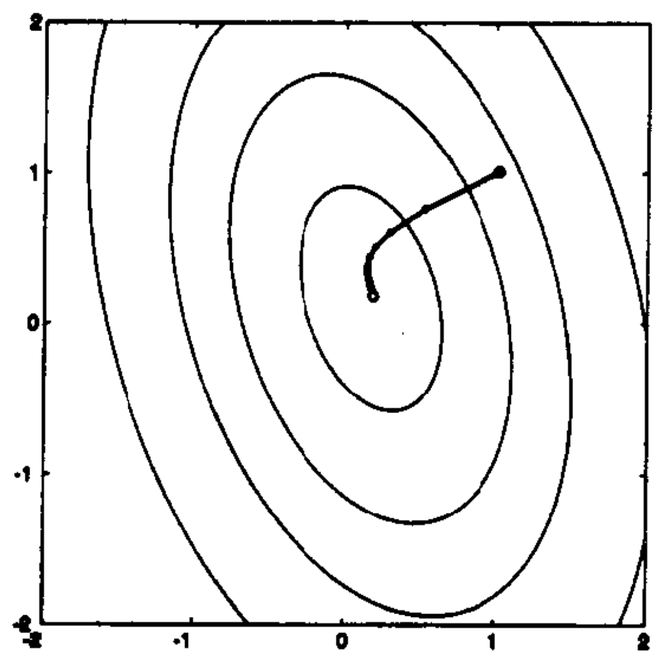



图 9-15 例题 P9.3 中 $\alpha = 0.05$ 的最速下降法轨迹

敛。第 10 章我们将介绍一个用于训练线性网络的自适应最速下降算法。在这种自适应算法中，每次输入/输出对都使网络参数被更新。这样，网络能够适应环境的变化。

（ii）本例题中赫森矩阵的最大特征值为 $\lambda_1 = 10.6$(见 P8.6)，故最大的稳定学习速度为

9-28

$$\alpha < \frac{2}{10.6} = 0.1887$$

**P9.4** 求下列函数的以 $\mathbf{x}_0 = [1 \quad -2]^T$ 为初始点的牛顿法一次迭代。本题结果离 $F(\mathbf{x})$ 极小点有多近？试予以说明。

$$F(\mathbf{x}) = e^{(x_1^2 - x_1 + 2x_2^2 + 4)}$$

**解**

首先求梯度和赫森矩阵。梯度为

$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} = e^{(x_1^2 - x_1 + 2x_2^2 + 4)} \begin{bmatrix} (2x_1 - 1) \\ (4x_2) \end{bmatrix}$$

赫森矩阵为

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2}{\partial x_1^2} F(\mathbf{x}) & \frac{\partial^2}{\partial x_1 \partial x_2} F(\mathbf{x}) \\ \frac{\partial^2}{\partial x_2 \partial x_1} F(\mathbf{x}) & \frac{\partial^2}{\partial x_2^2} F(\mathbf{x}) \end{bmatrix}$$

$$= e^{(x_1^2 - x_1 + 2x_2^2 + 4)} \begin{bmatrix} 4x_1^2 - 4x_1 + 3 & (2x_1 - 1)(4x_2) \\ (2x_1 - 1)(4x_2) & 16x_2^2 + 4 \end{bmatrix}$$

在初始点 $\mathbf{x}_0$，有

$$\mathbf{g}_0 = \nabla F(\mathbf{x}) \Big|_{\mathbf{x} = \mathbf{x}_0} = \begin{bmatrix} 0.163 \times 10^6 \\ -1.302 \times 10^6 \end{bmatrix}$$

和

$$\mathbf{A}_0 = \nabla^2 F(\mathbf{x}) \Big|_{\mathbf{x} = \mathbf{x}_0} = \begin{bmatrix} 0.049 \times 10^7 & -0.130 \times 10^7 \\ -0.130 \times 10^7 & 1.107 \times 10^7 \end{bmatrix}$$

9-29

所以由式(9.43)得牛顿法的第一次迭代为

$$\mathbf{x}_1 = \mathbf{x}_0 - \mathbf{A}_0^{-1}\mathbf{g}_0 = \begin{bmatrix} 1 \\ -2 \end{bmatrix} - \begin{bmatrix} 0.049\times10^7 & -0.130\times10^7 \\ -0.130\times10^7 & 1.107\times10^7 \end{bmatrix}^{-1} \begin{bmatrix} 0.163\times10^6 \\ -1.302\times10^6 \end{bmatrix} = \begin{bmatrix} 0.971 \\ -1.886 \end{bmatrix}$$

这一点离 $F(\mathbf{x})$的极小点有多近呢？首先注意到 $F(\mathbf{x})$的指数部分是一个二次函数：

$$x_1^2 - x_1 + 2x_2^2 + 4 = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{d}^T\mathbf{x} + c = \frac{1}{2}\mathbf{x}^T\begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}\mathbf{x} + [-1 \quad 0]\mathbf{x} + 4$$

$F(\mathbf{x})$的极小点即指数部分的极小点，即

$$\mathbf{x}^* = -\mathbf{A}^{-1}\mathbf{d} = -\begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}^{-1}\begin{bmatrix} -1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}$$

所以牛顿法只是向真正的极小点收敛了一小步。这是因为 $F(\mathbf{x})$无法由一个二次函数在 $\mathbf{x}_0 = [1 \quad -2]^T$ 的邻域内精确近似。

本例中牛顿法可以收敛到真正的极小点，但要经过多次迭代。图 9-16 所示为牛顿法的轨迹。

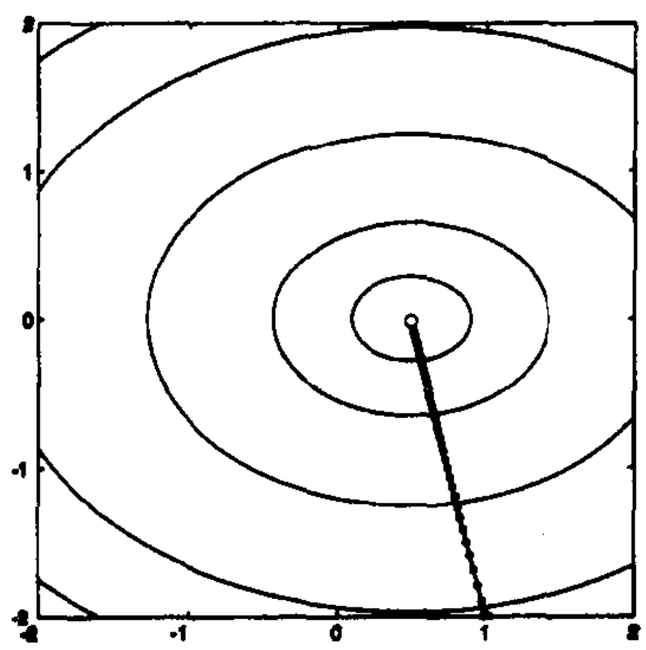



图 9-16 例题 P9.4 的牛顿法轨迹

9-30

**P9.5** 已知函数

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\mathbf{x}$$

起始点为

$$\mathbf{x}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

试比较牛顿法和最快速下降法的性能。

**解**

回忆这个函数是关于一个驻点凹槽的例子(见式(8.59)和图 8-19)。其梯度是

$$\nabla F(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{d} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\mathbf{x}$$

赫森矩阵是

$$\nabla^2 F(\mathbf{x}) = \mathbf{A} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

牛顿法为

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1}\mathbf{g}_k$$

注意：由于赫森矩阵是奇异矩阵，该算法无法实际运行。从第 8 章的讨论我们知道，该函数没有强极小点，但沿直线 $x_1 = x_2$ 有一个弱极小点。

用最速下降算法会出现什么情况呢？如果学习速度为 $\alpha = 0.1$，从初始点出发，前两步迭代为

$$\mathbf{x}_1 = \mathbf{x}_0 - \alpha\mathbf{g}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} - 0.1\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0.9 \\ -0.1 \end{bmatrix}$$

$$\mathbf{x}_2 = \mathbf{x}_1 - \alpha\mathbf{g}_1 = \begin{bmatrix} 0.9 \\ -0.1 \end{bmatrix} - 0.1\begin{bmatrix} 2 \\ -2 \end{bmatrix} = \begin{bmatrix} 0.8 \\ -0.2 \end{bmatrix}$$

9-31

图 9-17 为完整的轨迹。本例中最速下降算法比牛顿法性能要好。最速下降算法收敛到一个极小点(弱极小点)，而牛顿法不收敛。第 12 章我们要讨论一种将牛顿法与最速下降法相结合的技术，以克服赫森矩阵的奇异性(或类奇异性)的影响。

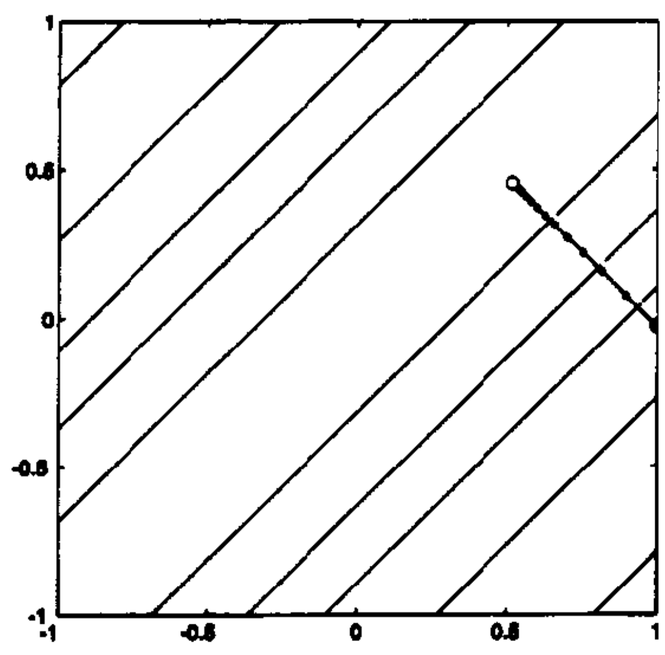



图 9-17 例题 P9.5 在 $\alpha = 0.1$ 时的最速下降轨迹

**P9.6** 已知函数

$$F(\mathbf{x}) = x_1^3 + x_1 x_2 - x_1^2 x_2^2$$

(i) 初始点为 $\mathbf{x}_0 = [1 \quad 1]^T$，求牛顿法的一次迭代。

(ii) 求 $F(\mathbf{x})$关于 $\mathbf{x}_0$ 的二阶泰勒级数展开。这个二次函数在(i)中的 $\mathbf{x}_1$ 点能达到极小值吗？试解释。

**解**

(i) $F(\mathbf{x})$的梯度为

$$\nabla F(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial}{\partial x_1} F(\mathbf{x}) \\ \dfrac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 3x_1^2 + x_2 - 2x_1 x_2^2 \\ x_1 - 2x_1^2 x_2 \end{bmatrix}$$

9-32 赫森矩阵为

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} 6x_1 - 2x_2^2 & 1 - 4x_1x_2 \\ 1 - 4x_1x_2 & -2x_1^2 \end{bmatrix}$$

在初始点有

$$\mathbf{g}_0 = \nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_0} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

$$\mathbf{A}_0 = \nabla^2 F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_0} = \begin{bmatrix} 4 & -3 \\ -3 & -2 \end{bmatrix}$$

牛顿法的第一次迭代为

$$\mathbf{x}_1 = \mathbf{x}_0 - \mathbf{A}_0^{-1}\mathbf{g}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 4 & -2 \\ -3 & -2 \end{bmatrix}^{-1} \begin{bmatrix} 2 \\ -1 \end{bmatrix} = \begin{bmatrix} 0.5882 \\ 1.1176 \end{bmatrix}$$

（ii）由式(9.40)，$F(\mathbf{x})$在 $\mathbf{x}_0$ 的二阶泰勒级数展开式为

$$F(\mathbf{x}) = F(\mathbf{x}_0 + \Delta\mathbf{x}_0) = F(\mathbf{x}_0) + \mathbf{g}_0^T\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}_0^T\mathbf{A}_0\Delta\mathbf{x}_0$$

将 $\mathbf{x}_0$，$\mathbf{g}_0$ 和 $\mathbf{A}_0$ 代入上式，得

$$F(\mathbf{x}) \approx 1 + [2 \quad -1]\left\{\mathbf{x} - \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right\} + \frac{1}{2}\left\{\mathbf{x} - \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right\}^T \begin{bmatrix} 4 & -3 \\ -3 & -2 \end{bmatrix}\left\{\mathbf{x} - \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right\}$$

化简得

$$F(\mathbf{x}) \approx -2 + [1 \quad 4]\mathbf{x} + \frac{1}{2}\mathbf{x}^T\begin{bmatrix} 4 & -3 \\ -3 & -2 \end{bmatrix}\mathbf{x}$$

该函数在 $\mathbf{x}_1$ 有一个驻点。问题在于该驻点是否是一个强极小点。这可由赫森矩阵的特征值确定。如果两个特征值都为正，则它是一个强极小点。如果两个特征值都为负，则它是一个强极大点。如果两个特征值符号相反，则它是一个鞍点。本例中 $\mathbf{A}_0$ 的特征值为

$$\lambda_1 = 5.24, \lambda_2 = -3.24$$ 9-33

由于这是一个鞍点，所以 $F(\mathbf{x})$在 $\mathbf{x}_0$ 的二次近似在 $\mathbf{x}_1$ 点没有极小化。图 9-18 所示为 $F(\mathbf{x})$的轮廓线图及其二次近似。

这种问题也在图 9-18 和图 9-19 中有说明。牛顿法无法确定当前点是否为函数的二次近似的驻点，它不能区分极小点、极大点和鞍点。

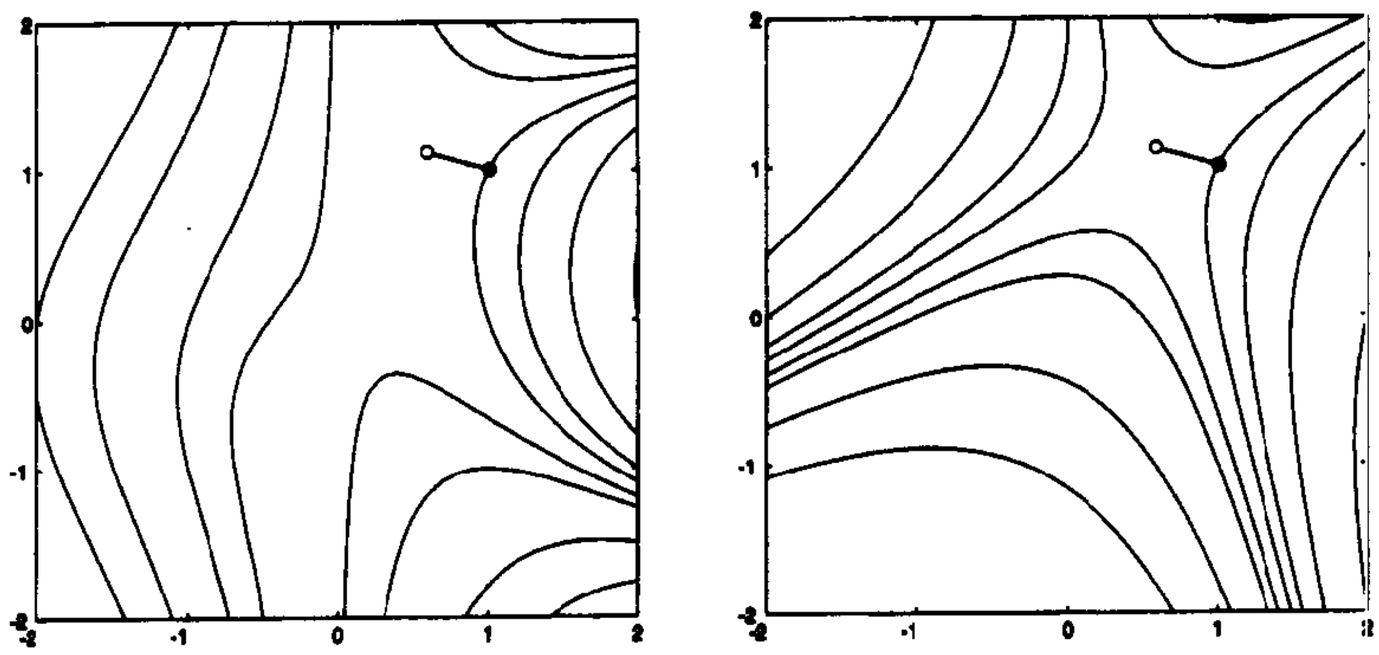



图 9-18 牛顿法在 $\mathbf{x}_0 = [1 \quad 1]^T$ 的一次迭代

**P9.7** 用共轭梯度法重做例题 P9.3 的(i)题。

**解**

已知函数为

$$F(\mathbf{x}) = 0.25 + [-2 \quad -1]\mathbf{x} + \frac{1}{2}\mathbf{x}^T\begin{bmatrix} 10 & 2 \\ 2 & 4 \end{bmatrix}\mathbf{x}$$

在 $\mathbf{x}_0$ 梯度为

$$\mathbf{g}_0 = \nabla F(\mathbf{x}_0) = \mathbf{A}\mathbf{x}_0 + \mathbf{d} = \begin{bmatrix} 10 & 2 \\ 2 & 4 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} -2 \\ -1 \end{bmatrix} = \begin{bmatrix} 10 \\ 5 \end{bmatrix}$$

于是第一次搜索方向为

$$\mathbf{p}_0 = -\mathbf{g}_0 = \begin{bmatrix} -10 \\ -5 \end{bmatrix}$$

9-34 要使二次函数沿直线极小化，可以用式(9.31)：

$$\alpha_0 = \frac{\mathbf{g}_0^T\mathbf{p}_0}{\mathbf{p}_0^T\mathbf{A}\mathbf{p}_0} = -\frac{[10 \quad 5]\begin{bmatrix} -10 \\ -5 \end{bmatrix}}{[-10 \quad -5]\begin{bmatrix} 10 & 2 \\ 2 & 4 \end{bmatrix}\begin{bmatrix} -10 \\ -5 \end{bmatrix}} = \frac{-125}{1300} = 0.0962$$

因此共轭梯度法的第一次迭代为

$$\mathbf{x}_1 = \mathbf{x}_0 + \alpha_0\mathbf{p}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 0.0962\begin{bmatrix} -10 \\ -5 \end{bmatrix} = \begin{bmatrix} 0.038 \\ 0.519 \end{bmatrix}$$

现在用式(9.60)求第二个搜索方向。首先求 $\mathbf{x}_1$ 点的梯度：

$$\mathbf{g}_1 = \nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_1} = \begin{bmatrix} 10 & 2 \\ 2 & 4 \end{bmatrix}\begin{bmatrix} 0.038 \\ 0.519 \end{bmatrix} + \begin{bmatrix} -2 \\ -1 \end{bmatrix} = \begin{bmatrix} -0.577 \\ 1.154 \end{bmatrix}$$

现在求 $\beta_1$：

$$\beta_1 = \frac{\Delta\mathbf{g}_0^T\mathbf{g}_1}{\mathbf{g}_0^T\mathbf{g}_0} = \frac{[-10.577 \quad -3.846]\begin{bmatrix} -0.577 \\ 1.154 \end{bmatrix}}{[10 \quad 5]\begin{bmatrix} 10 \\ 5 \end{bmatrix}} = \frac{1.665}{125} = 0.0133$$

这里使用了 PoLak 和 Ribiére(式(9.63))的方法。(求 $\beta_1$ 的另外两种方法对二次函数的结果不变。可试一下。)于是第二次搜索方向为：

$$\mathbf{p}_1 = -\mathbf{g}_1 + \beta_1\mathbf{p}_0 = \begin{bmatrix} 0.577 \\ -1.154 \end{bmatrix} + 0.0133\begin{bmatrix} -10 \\ -5 \end{bmatrix} = \begin{bmatrix} 0.444 \\ -1.220 \end{bmatrix}$$

由式(9.31)求第二次迭代的学习速度为

$$\alpha_1 = \frac{[-0.577 \quad 1.154]\begin{bmatrix} 0.444 \\ -1.220 \end{bmatrix}}{[0.444 \quad -1.220]\begin{bmatrix} 10 & 2 \\ 2 & 4 \end{bmatrix}\begin{bmatrix} 0.444 \\ -1.220 \end{bmatrix}} = \frac{-1.664}{5.758} = 0.2889$$

9-35 因此共轭梯度法的第二次迭代为

$$\mathbf{x}_2 = \mathbf{x}_1 + \alpha_1 \mathbf{p}_1 = \begin{bmatrix} 0.038 \\ 0.519 \end{bmatrix} + 0.2889 \begin{bmatrix} 0.444 \\ -1.220 \end{bmatrix} = \begin{bmatrix} 0.1667 \\ 0.1667 \end{bmatrix}$$

经过两次迭代如期到达极小点，轨迹如图 9-19 所示。

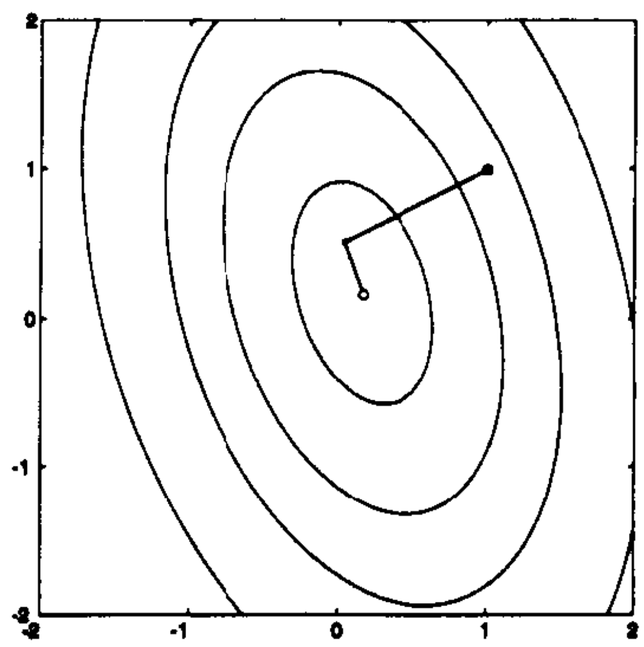



图 9-19 例题 P9.7 的共轭梯度轨迹

**P9.8** 证明共轭向量线性无关。

**证**

设向量集$\{\mathbf{p}_0, \mathbf{p}_1, \cdots, \mathbf{p}_{n-1}\}$对赫森矩阵 **A** 共轭。如果这些向量线性相关，则由式(5.4)有

$$\sum_{j=0}^{n-1} a_j \mathbf{p}_j = \mathbf{0}$$

$a_0, a_1, \cdots, a_{n-1}$为不全为零的常数。

用 $\mathbf{p}_k^T\mathbf{A}$ 乘以上式两边，则有

$$\mathbf{p}_k^T\mathbf{A}\sum_{j=0}^{n-1} a_j \mathbf{p}_j = \sum_{j=0}^{n-1} a_j \mathbf{p}_k^T\mathbf{A}\mathbf{p}_j = a_k \mathbf{p}_k^T\mathbf{A}\mathbf{p}_k = 0$$

这里用到了式(9.52)关于共轭向量的定义。如果 **A** 是正定的(存在一个惟一的强极小点)，则 $\mathbf{p}_k^T\mathbf{A}\mathbf{p}_k$ 为严格正的。这表明对所有的 $k$，$a_k$ 必为零，与假设矛盾。所以共轭向量一定线性无关。 9-36

## 9.5 结束语

本章介绍了三个不同的优化算法：最速下降法，牛顿法，共轭梯度法。这三种算法的基础是泰勒级数展开。最速下降法由一阶泰勒展开导出，而牛顿法和共轭梯度法则用于二阶(二次)函数。

最速下降法的优点是简单且只要计算梯度。如果学习速度足够小，它还能保证收敛到一个驻点。其缺点是训练时间通常比其他算法长，当二次函数赫森矩阵的特征值相差很大时尤其如此。

牛顿法通常比最速下降法快得多。对于二次函数，它能够一次迭代收敛到一个驻点。它的一个缺点是需要计算和存储赫森矩阵及其逆矩阵。另外，牛顿法的收敛特性也很复杂。第 12 章我们将介绍经过修正的牛顿法，它克服了原标准算法的缺点。

共轭梯度算法是最速下降法与牛顿法折中的产物。它能在有限的迭代步数内收敛到二次

函数的极小点，且不需要计算和存储赫森矩阵。它最适合于解决参数量很大且赫森矩阵的计算和存储不可行时的问题。

后面的几章我们将把这些优化算法用于训练神经网络。第 10 章将介绍一种最速下降的近似算法，即 Widrow-Hoff 学习，可用于训练线性网络。第 11 章推广用于训练多层网络 Widrow-Hoff 学习。在第 12 章，共轭梯度算法和牛顿法的一个变形将用于加速多层网络的训练。

9-37

## 参考文献

[Batt92] R. Battiti, "First and Second Order Methods for Learning: Between Steepest Descent and Newton's Method," *Neural Computation*, Vo. 4, No. 2, PP. 141 - 166, 1992.

本文评述运用一阶和二阶导数的非约束优化的最新发展，讨论最适于神经网络应用的技术。

[Brog91] W. L. Brogan, *Modern Control Theory*, 3nd Ed., Englewood Cliffs, NJ: Prentice-Hall, 1991.

这是一本写得很好的线性系统学科的书。前半部分讲述线性代数。它还对线性微分方程及线性非线性系统的稳定性作了精采的讨论，并收入许多已解决的问题。

[Gill81] P. E. Gill, W. Murray and M. H. Wright, *Practical Optimization*, New York: Academic Press, 1981.

本书着重讨论优化算法的实际应用问题。它提出了一种优化算法的激励机制，并讨论了影响算法性能的实现细节问题。

[Himm72] D. M. Himmelblau, *Applied Non Linear Programming*, New York: McGraw-Hill, 1972.

本书是内容广泛的非线性优化的课本。它讨论了约束和非约束的优化问题。该书非常全面，包含许多例题详解。

[Scal85] C. E. Scales, *Introduction to Non-Linear Optimization*, New York: Spinger-Verlag, 1985.

本书是一本可读性很强的关于主要的优化算法的书，本书强调的重点是优化的方法而不是存在定理和收敛性证明。它通过图形和例子来直观地解释算法。多数算法都给出了伪码。

9-38

## 习题

**E9.1** 在例题 P9.1 中我们讨论了应用于典型的二次函数的最速下降算法的最大稳定学习速度的求解。如果采用较大的学习速度，算法是否一定发散？或者说是否存在保证算法收敛的条件？

**E9.2** 求下列函数的极小点：

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\begin{bmatrix} 6 & -2 \\ -2 & 6 \end{bmatrix}\mathbf{x} + \begin{bmatrix} -1 & -1 \end{bmatrix}\mathbf{x}$$

（i）画出该函数的轮廓线图。

(ii) 在(i)的轮廓线图上画出最速下降算法的轨迹，设初始点为 $\mathbf{x}_0=[0\quad 0]^T$，学习速度足够小。

(iii) 若学习速度 $\alpha=0.1$，进行最速下降算法两次迭代。

(iv) 最大的稳定学习速度是多少?

(v) 对于(ii)中给定的初始点求最大的稳定学习速度。(见习题 E9.1。)

(vi) 写出 MATLAB 的 M-file 文件，实现本题中的最速下降算法，并用以检验从(i)到(v)的答案。

**E9.3** 已知二次函数

$$F(\mathbf{x}) = x_1^2 + 2x_2^2$$

(i) 求函数沿下列直线的极小点:

$$\mathbf{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \alpha \begin{bmatrix} -1 \\ -2 \end{bmatrix}$$

(ii) 证明 $F(\mathbf{x})$在(i)题中的极小点处的梯度垂直于最小化的搜索方向。 9-39

**E9.4** 对习题 E8.3 中的函数，从初始的估计值 $\mathbf{x}_0=[1\quad 1]^T$ 开始，用线性最小化的最速下降法迭代两次。写出 MATLAB 的 M-file 文件，检查答案。

**E9.5** 考虑下面的函数:

$$F(\mathbf{x}) = [1+(x_1+x_2-5)^2][1+(3x_1-2x_2)^2]$$

(i) 从初始估计值 $\mathbf{x}_0=[10\quad 10]^T$ 开始，用牛顿法迭代一次。

(ii) 从初始估计值 $\mathbf{x}_0=[2\quad 2]^T$ 开始，重复(i)题中的操作。

(iii) 求函数的极小点，并与前两部分的结果比较。

**E9.6** 考虑习题 E8.5 中的函数。写出求此函数的最速下降法和牛顿法的 MATLAB M-file文件。对不同的初始值，测试算法的性能。

**E9.7** 使用共轭梯度算法重做习题 E9.4。对式(9.61)~(9.63)中的三种方法，每种方法至少做一次。

**E9.8** 证明或反驳下面的断言:

若 $\mathbf{p}_1$ 共轭于 $\mathbf{p}_2$ 且 $\mathbf{p}_2$ 共轭于 $\mathbf{p}_3$，则 $\mathbf{p}_1$ 共轭于 $\mathbf{p}_3$ 9-40

# 第10章 Widrow-Hoff学习算法

## 10.1 目的

在前面两章的学习中，我们打下了性能学习的基础。可以看到，网络通过训练来优化其性能。在本章中，将会把性能学习的原理用于单层线性神经网络。

Widrow-Hoff学习算法是一个近似最速下降法，其中性能指标是均方误差。这个算法很重要，原因有两个：第一，它被广泛使用于现在的信号处理应用中，其中有几个应用将在本章介绍；第二，它是多层网络中BP算法的先驱(BP算法将在第11章中讲述)。

10-1

## 10.2 理论和实例

Bernard Widrow早在20世纪五十年代末便开始了神经网络的研究工作，几乎在同一时期，Frank Rosenblatt设计了感知机学习规则。在1960年，Widrow和他的研究生Marcian Hoff引入了ADALINE(ADAptive LInear NEuron，自适应线性神经元)网络和一个称为LMS(Least Mean Square，最小均方)算法的学习规则[WiHo60]。

他们的ADALINE网络与感知机非常相似，不同之处在于它的传输函数是线性函数而不是硬极限函数。ADALINE和感知机均受同样的局限性的影响：它们只能解决线性可分问题。但是，LMS算法比感知机学习规则要强大得多。感知机规则能保证将训练模式收敛到一个可正确分类的解上，但得到的网络对噪声敏感，因为训练模式常接近网络的判定边界。而LMS算法使均方误差最小化，从而使网络的判定边界尽量远离训练模式。

LMS算法在实际中的应用比感知机学习规则多，尤其是在数字信号处理领域。例如，大多数长距离电话线路使用ADALINE网络来消除回声。本章将在后面详细讨论这些应用。

因为LMS算法在信号处理应用中取得了巨大成功，而该算法在多层网络中则不太成功，所以在20世纪60年代早期，Widrow中止了他在神经网络方面的工作，而开始全力研究自适应的信号处理。直到80年代，他才重返神经网络领域，并开始研究自适应控制中神经网络的使用。在研究中使用了由他最初的LMS算法得到的时间反向传播法。

### 10.2.1 ADALINE网络

ADALINE网络如图10-1所示。注意，它具有与第4章中所讨论的感知机网络相同的基本结构。惟一的不同点是它使用了一个线性传输函数。

10-2

网络输出由下式给出：

$$\mathbf{a} = \mathbf{purelin}(\mathbf{Wp} + \mathbf{b}) = \mathbf{Wp} + \mathbf{b} \tag{10.1}$$

回忆过去对感知机网络的讨论可得到网络输出向量的第 $i$ 行元素为：

$$a_i = purelin(n_i) = purelin({}_i\mathbf{w}^T\mathbf{p} + b_i) = {}_i\mathbf{w}^T\mathbf{p} + b_i \tag{10.2}$$

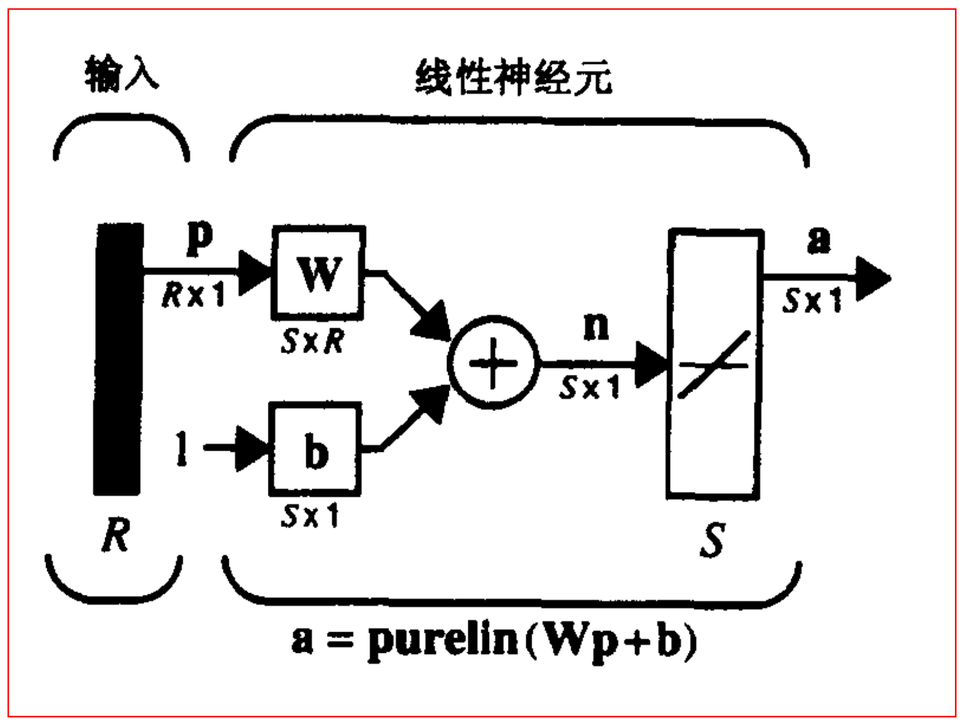



图 10-1　ADALINE 网络

这里，$_i\mathbf{w}$ 由 $\mathbf{W}$ 的第 $i$ 行元素组成：

$$_i\mathbf{w} = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,R} \end{bmatrix} \tag{10.3}$$

*单层 ADALINE 网络*

为了简化讨论，这里考虑一个两输入的单层 ADALINE 网络，如图 10-2 所示。网络的输出由下式给出：

$$\begin{aligned} a &= purelin(n) = purelin(_1\mathbf{w}^T\mathbf{p} + b) = {}_1\mathbf{w}^T\mathbf{p} + b \\ &= {}_1\mathbf{w}^T\mathbf{p} + b = w_{1,1}p_1 + w_{1,2}p_2 + b \end{aligned} \tag{10.4}$$

10-3

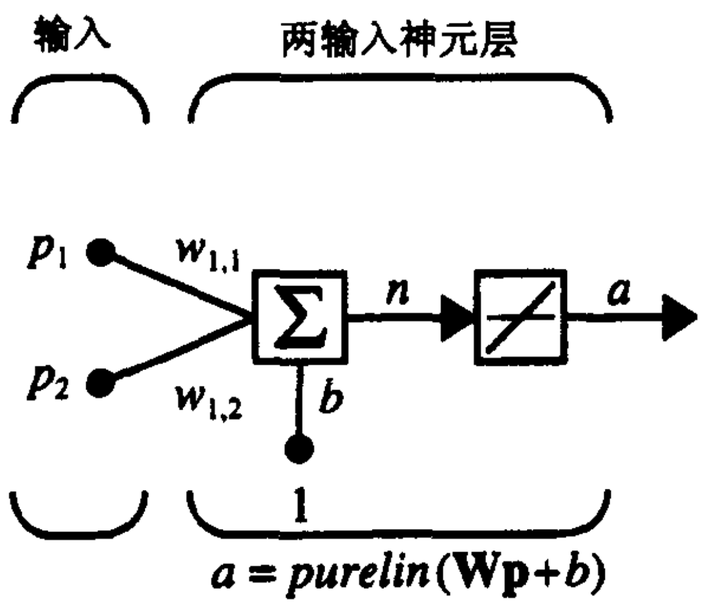



图 10-2　两输入的线性神经元

由第 4 章知道，感知机有一个判定边界，它由净输入为 0 的输入向量所决定。那么，ADALINE 是否也有这样一个边界呢？显然是这样的。若设 $n = 0$，则 $_1\mathbf{w}^T\mathbf{p} + b = 0$，它定义了如图 10-3 中的一条线。

图中灰色区域对应的神经元输出大于 0，白的区域中神经元输出小于 0。那么对 ADALINE 这意味着什么呢？它说明 ADALINE 网络可将对象分为两类。然而，只有对象是线性可分时它才能做到这一点。因此，在这一点上，ADALINE 网络具有和感知机同样的限制。

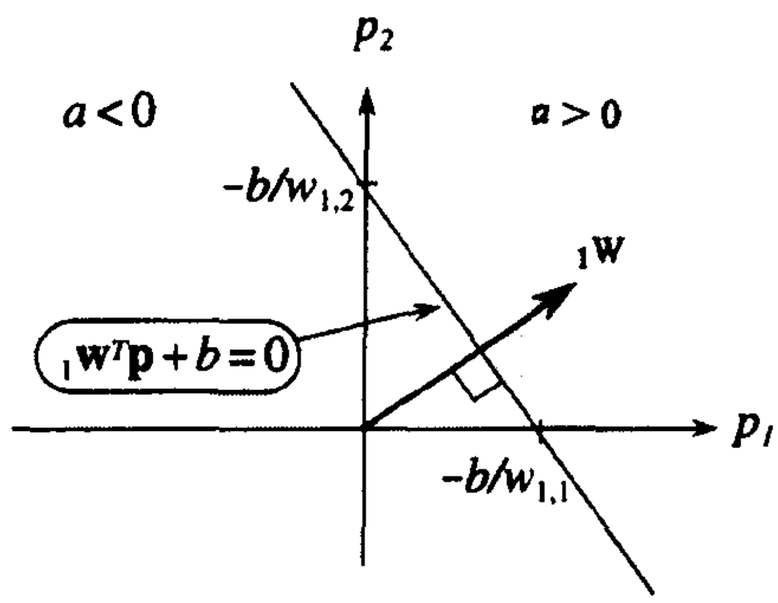



图 10-3 两输入 ADALINE 的判定边界

### 10.2.2 均方误差

前面已看到了 ADALINE 网络的性质，下面开始 LMS 算法的讨论。与感知机规则一样，
10-4 LMS 算法也是有监督训练的一个例子，其中，学习规则将使用一个正确的行为样本的集合：

$$\{\mathbf{p}_1,\mathbf{t}_1\},\{\mathbf{p}_2,\mathbf{t}_2\},\cdots,\{\mathbf{p}_Q,\mathbf{t}_Q\} \tag{10.5}$$

这里，$\mathbf{p}_q$ 是网络的一个输入，$\mathbf{t}_q$ 是对应的目标输出。网络每输入一个数据，便将网络输出与目标输出相比较一次。

为使均方误差最小化，LMS 算法将调整 ADALINE 网络的权值和偏置值。这里的误差指的是目标输出和网络输出之差。本节中我们要讨论这个性能指数。首先考虑单神经元的情况。

为简化讨论，我们将所有要调整的参数，包括偏置值，组成一个向量：

$$\mathbf{x}=\begin{bmatrix}{}_1\mathbf{w}\\ b\end{bmatrix} \tag{10.6}$$

类似地，我们将偏置值输入“1”作为输入向量的一部分：

$$\mathbf{z}=\begin{bmatrix}\mathbf{p}\\ 1\end{bmatrix} \tag{10.7}$$

对网络输出，我们通常用下式来表示：

$$a={}_1\mathbf{w}^T\mathbf{p}+b \tag{10.8}$$

现在，可以将它写作

$$a=\mathbf{x}^T\mathbf{z} \tag{10.9}$$

**均方误差** 这样，我们可以方便地写出 ADALINE 网络的均方误差的表达式：

$$F(\mathbf{x})=E[e^2]=E[(t-a)^2]=E[(t-\mathbf{x}^T\mathbf{z})^2] \tag{10.10}$$

其中，期望值在所有输入/输出对上求得。(这里使用 $E[\quad]$ 来表示期望值，并使用期望的广义定义，即确定性信号的时间平均值。参见[WiSt85]。)上式可扩展为：

10-5

$$\begin{aligned}F(\mathbf{x})&=E[t^2-2t\mathbf{x}^T\mathbf{z}+\mathbf{x}^T\mathbf{z}\mathbf{z}^T\mathbf{x}]\\&=E[t^2]-2\mathbf{x}^TE[t\mathbf{z}]+\mathbf{x}^TE[\mathbf{z}\mathbf{z}^T]\mathbf{x}\end{aligned} \tag{10.11}$$

这可以表示成下面更方便的形式：

$$F(\mathbf{x})=c-2\mathbf{x}^T\mathbf{h}+\mathbf{x}^T\mathbf{R}\mathbf{x} \tag{10.12}$$

其中

$$c=E[t^2],\mathbf{h}=E[t\mathbf{z}]\quad 且\ \mathbf{R}=E[\mathbf{z}\mathbf{z}^T] \tag{10.13}$$

**相关矩阵**　在这里，向量 $\mathbf{h}$ 给出输入向量和对应目标输出之间的相关系数，$\mathbf{R}$ 是输入的相关矩阵。矩阵的对角线元素等于输入向量元素的均方值。

等式(8.35)中的二次函数为

$$F(\mathbf{x}) = c + \mathbf{d}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} \tag{10.14}$$

将它与等式(10.12)相比较，我们可以看到，ADALINE 网络的均方差性能指数是一个二次函数，其中

$$\mathbf{d} = -2\mathbf{h} \text{ 且 } \mathbf{A} = 2\mathbf{R} \tag{10.15}$$

这是一个很重要的结果。从第 8 章中我们知道，二次函数的性质主要取决于赫森矩阵 $\mathbf{A}$。例如，若赫森矩阵的特征值全是正的，则函数有一个惟一的全局最小点。

这里，赫森矩阵是相关矩阵 $\mathbf{R}$ 的两倍，并且所有相关矩阵是正定的或半正定的，这意味着它们决不会有负的特征值。但是还有两种可能：若相关矩阵只有正的特征值，性能指数将有一个惟一的全局极小点(见图 8-8)；若相关矩阵有一些特征值为 0，性能指数将有一个弱极小点(见图 8-10)或没有极小点(见问题 8-8)，这取决于是否有向量 $\mathbf{d} = -2\mathbf{h}$。

现在来确定性能指数的驻点。从前面对二次函数的讨论我们知道，梯度为：

$$\nabla F(\mathbf{x}) = \nabla\left(c + \mathbf{d}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}\right) = \mathbf{d} + \mathbf{A}\mathbf{x} = -2\mathbf{h} + 2\mathbf{R}\mathbf{x} \tag{10.16}$$

$F(\mathbf{x})$的驻点可以通过令梯度等于 0 来求得：

$$-2\mathbf{h} + 2\mathbf{R}\mathbf{x} = 0 \tag{10.17}$$

10-6

因此，若相关矩阵是正定的，则将有一个惟一的驻点，它是一个强极小点：

$$\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h} \tag{10.18}$$

值得注意的是，惟一解的存在只依赖于相关矩阵 $\mathbf{R}$。因此，输入向量的性质决定了是否存在惟一解。

### 10.2.3　LMS 算法

前面已分析了性能指数，下一步是设计一个确定极小点的算法。若能计算出统计量 $\mathbf{h}$ 和 $\mathbf{R}$，就能从式(10.18)直接求出极小点。若不想计算 $\mathbf{R}^{-1}$，可以对由式(10.16)计算得来的梯度使用最速下降法。然而，通常并不希望或不方便计算 $\mathbf{h}$ 和 $\mathbf{R}$。因而，我们将使用一个近似的最速下降法，其中使用一个估计的梯度值。

Widrow 和 Hoff 的主要观点是用下式来估计 $F(\mathbf{x})$的均方误差：

$$\hat{F}(\mathbf{x}) = (t(k) - a(k))^2 = e^2(k) \tag{10.19}$$

其中均方误差的期望被第 $k$ 次迭代时的均方误差所代替。因而，每次迭代中，梯度估计值为：

$$\hat{\nabla} F(\mathbf{x}) = \nabla e^2(k) \tag{10.20}$$

$\nabla e^2(k)$的前 $R$ 个元素是关于网络权值的导数值，第$(R+1)$个元素则是关于偏置值的导数值。于是有

$$[\nabla e^2(k)]_j = \frac{\partial e^2(k)}{\partial w_{1,j}} = 2e(k)\frac{\partial e(k)}{\partial w_{1,j}}, \quad j = 1,2,\cdots,R \tag{10.21}$$

及

$$[\nabla e^2(k)]_{R+1} = \frac{\partial e^2(k)}{\partial b} = 2e(k)\frac{\partial e(k)}{\partial b} \tag{10.22}$$

10-7 下面考虑上述等式中后面的偏导数项。首先计算 $e(k)$对网络权值 $w_{1,j}$的偏导数：

$$\begin{aligned}\frac{\partial e(k)}{\partial w_{1,j}} &= \frac{\partial[t(k)-a(k)]}{\partial w_{1,j}} = \frac{\partial}{\partial w_{1,j}}[t(k)-({}_1\mathbf{w}^T\mathbf{p}(k)+b)] \\ &= \frac{\partial}{\partial w_{1,j}}\left[t(k)-\left(\sum_{i=1}^{R} w_{1,i}p_i(k)+b\right)\right]\end{aligned} \tag{10.23}$$

其中 $p_i(k)$是第 $k$ 次迭代中输入向量的第 $i$ 个元素。上式可简化为：

$$\frac{\partial e(k)}{\partial w_{1,j}} = -p_j(k) \tag{10.24}$$

类似地可以得到梯度的最后一项：

$$\frac{\partial e(k)}{\partial b} = -1 \tag{10.25}$$

注意 $p_j(k)$和 1 是输入向量 $\mathbf{z}$ 的元素，因此第 $k$ 次迭代时均方误差的梯度为：

$$\hat{\nabla}F(\mathbf{x}) = \nabla e^2(k) = -2e(k)\mathbf{z}(k) \tag{10.26}$$

现在我们可以看到用 $k$ 次迭代时的单个误差来近似均方误差的好处，如在式(10.19)中。要计算这个梯度的近似值，我们只需用误差乘输入。

$\nabla F(\mathbf{x})$的近似量可被用于最速下降法。根据式(9.10)，具有固定的学习速度的最速下降法为

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha\nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_k} \tag{10.27}$$

用式(10.26)中的$\hat{\nabla}F(\mathbf{x})$代替$\nabla F(\mathbf{x})$，可以得到

$$\mathbf{x}_{k+1} = \mathbf{x}_k + 2\alpha e(k)\mathbf{z}(k) \tag{10.28}$$

或

$${}_1\mathbf{w}(k+1) = {}_1\mathbf{w}(k) + 2\alpha e(k)\mathbf{p}(k) \tag{10.29}$$

和

$$b(k+1) = b(k) + 2\alpha e(k) \tag{10.30}$$

10-8 最后两个等式构成了最小均方(LMS)算法，它称为 δ 规则或 Widrow-Hoff 学习算法。

前面的结果可加以修改用来处理有多个输出的情况，即有多个神经元，如图 10-1。更新权值矩阵的第 $i$ 行时使用：

$${}_i\mathbf{w}(k+1) = {}_i\mathbf{w}(k) + 2\alpha e_i(k)\mathbf{p}(k) \tag{10.31}$$

其中，$e_i(k)$是第 $k$ 次迭代时的第 $i$ 个元素。更新偏置值的第 $i$ 个元素使用

$$b_i(k+1) = b_i(k) + 2\alpha e_i(k) \tag{10.32}$$

**LMS 算法** LMS 算法可以方便地用矩阵记号表示：

$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha\mathbf{e}(k)\mathbf{p}^T(k) \tag{10.33}$$

和

$$\mathbf{b}(k+1) = \mathbf{b}(k) + 2\alpha\mathbf{e}(k) \tag{10.34}$$

注意这里误差 $\mathbf{e}$ 和偏置值 $\mathbf{b}$ 是向量。

### 10.2.4 收敛性分析

第 9 章中已分析过最速下降法的稳定性。那里，我们发现二次函数的最大稳定学习速度为 $\alpha < 2/\lambda_{max}$，其中 $\lambda_{max}$ 是赫森矩阵的最大特征值。下面我们分析 LMS 算法的收敛性，它与最速下降法近似。我们将发现结果是一样的。

首先注意 LMS 算法式(10.28)中，$\mathbf{x}_k$ 只是 $\mathbf{z}(k-1)$，$\mathbf{z}(k-2)$，…，$\mathbf{z}(0)$ 的函数。若假定后继的输入向量是统计独立的，则 $\mathbf{x}_k$ 独立于 $\mathbf{z}(k)$。下面我们将说明，对满是这个条件的稳态输入过程，权向量的期望值将收敛于

$$\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h} \tag{10.35}$$

这就上最小均方误差 $\{E[e_k^2]\}$ 的解(如在式(10.18)中所见的那样)。

回忆 LMS 算法(式(10.28))：

$$\mathbf{x}_{k+1} = \mathbf{x}_k + 2\alpha e(k)\mathbf{z}(k) \tag{10.36}$$

两边求期望得：

$$E[\mathbf{x}_{k+1}] = E[\mathbf{x}_k] + 2\alpha E[e(k)\mathbf{z}(k)] \tag{10.37}$$ 10-9

将误差用 $t(k) - \mathbf{x}_k^T\mathbf{z}(k)$ 代入得：

$$E[\mathbf{x}_{k+1}] = E[\mathbf{x}_k] + 2\alpha\{E[t(k)\mathbf{z}(k)] - E[(\mathbf{x}_k^T\mathbf{z}(k))\mathbf{z}(k)]\} \tag{10.38}$$

最后，用 $\mathbf{z}^T(k)\mathbf{x}_k$ 替换 $\mathbf{x}_k^T\mathbf{z}(k)$，整理后得：

$$E[\mathbf{x}_{k+1}] = E[\mathbf{x}_k] + 2\alpha\{E[t_k\mathbf{z}(k)] - E[(\mathbf{z}(k)\mathbf{z}^T(k))\mathbf{x}_k]\} \tag{10.39}$$

由于 $\mathbf{x}_k$ 独立于 $\mathbf{z}(k)$，从而得：

$$E[\mathbf{x}_{k+1}] = E[\mathbf{x}_k] + 2\alpha\{\mathbf{h} - \mathbf{R}E[\mathbf{x}_k]\} \tag{10.40}$$

即

$$E[\mathbf{x}_{k+1}] = [\mathbf{I} - 2\alpha\mathbf{R}]E(\mathbf{x}_k) + 2\alpha\mathbf{h} \tag{10.41}$$

当 $[\mathbf{I} - 2\alpha\mathbf{R}]$ 的所有特征值落在单位圆内时，此动态系统趋于稳定(见[Brog91]。从第 9 章中知道，$[\mathbf{I} - 2\alpha\mathbf{R}]$ 的特征值将为 $1 - 2\alpha\lambda_i$，其中 $\lambda_i$ 是 $\mathbf{R}$ 的特征值，因此系统稳定的条件为：

$$1 - 2\alpha\lambda_i > -1 \tag{10.42}$$

由于 $\lambda_i > 0$，$1 - 2\alpha\lambda_i$ 总是小于 1。因此，稳定的条件为：

$$\alpha < 1/\lambda_i, \text{对所有 } i \tag{10.43}$$

或

$$0 < \alpha < 1/\lambda_{max} \tag{10.44}$$

注意此条件等价于我们在第 9 章中推导出的最速下降法的条件，不过在那里使用的是赫森矩阵 $\mathbf{A}$ 的特征值。这里我们用的是输入相关矩阵 $\mathbf{R}$ 的特征值。(回忆 $\mathbf{A} = 2\mathbf{R}$。)若此稳定性条件满足，则稳态解为：

$$E[\mathbf{x}_{ss}] = [\mathbf{I} - 2\alpha\mathbf{R}]E[\mathbf{x}_{ss}] + 2\alpha\mathbf{h} \tag{10.45}$$

或

$$E[\mathbf{x}_{ss}] = \mathbf{R}^{-1}\mathbf{h} = \mathbf{x}^* \tag{10.46}$$ 10-10

因此，每次输入一个输入向量得到的 LMS 的解，与式(10.18)中最小均方误差的解是相同的。

为测试 ADALINE 网络和 LMS 算法，再考虑在第 3 章中讨论过的苹果/橘子区分问题。为简单起见，我们假定 ADALINE 网络的偏置值为 0。

式(10.29)中的 LMS 权值更新算法被用来在网络训练的每一步中计算新的权值：

$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha e(k)\mathbf{p}^T(k) \tag{10.47}$$

首先计算最大稳态学习速度 $\alpha$。通过求解输入相关矩阵的特征值可以得到它。橘子和苹果向量以及它们相应的目标输出为：

$$\left\{\mathbf{p}_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, t_1 = [-1]\right\}, \quad \left\{\mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, t_2 = [1]\right\} \tag{10.48}$$

若假定输入向量是以相等概率随机产生的，则可以如下计算输入相关矩阵：

$$\begin{aligned}\mathbf{R} &= E[\mathbf{p}\mathbf{p}^T] = \frac{1}{2}\mathbf{p}_1\mathbf{p}_1^T + \frac{1}{2}\mathbf{p}_2\mathbf{p}_2^T \\ &= \frac{1}{2}\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}\begin{bmatrix} 1 & -1 & -1 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}\begin{bmatrix} 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}\end{aligned} \tag{10.49}$$

**R** 的特征值为：

$$\lambda_1 = 1.0, \quad \lambda_2 = 0.0, \quad \lambda_3 = 2.0 \tag{10.50}$$

因此，最大稳态学习速度为

$$\alpha < \frac{1}{\lambda_{max}} = \frac{1}{2.0} = 0.5 \tag{10.51}$$

若保守些，可以取 $\alpha = 0.2$。(注意，在实际应用中，计算 **R** 可能是不实际的，这时可通过试错的办法来选择 $\alpha$ 的值。选择 $\alpha$ 的其他方法可参见[WiSt85]。)

10-11 开始时，我们可将所有权值设为 0，然后应用输入 $\mathbf{p}_1$，$\mathbf{p}_2$，$\mathbf{p}_1$，$\mathbf{p}_2$ 等，每次输入给出后便计算新的权值。(不必以交替的顺序给出权值，一个随机的顺序就行了。)给出 $\mathbf{p}_1$(橘子)和其目标输出 $-1$，我们得到

$$a(0) = \mathbf{W}(0)\mathbf{p}(0) = \mathbf{W}(0)\mathbf{p}_1 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} = 0 \tag{10.52}$$

和

$$e(0) = t(0) - a(0) = t_1 - a(0) = -1 - 0 = -1 \tag{10.53}$$

现在我们可以计算新的权矩阵：

$$\begin{aligned}\mathbf{W}(1) &= \mathbf{W}(0) + 2\alpha e(0)\mathbf{p}^T(0) \\ &= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} + 2(0.2)(-1)\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}^T = \begin{bmatrix} -0.4 & 0.4 & 0.4 \end{bmatrix}\end{aligned} \tag{10.54}$$

下一次给出 $\mathbf{p}_2$(苹果)和它的目标输出 1：

$$a(1) = \mathbf{W}(1)\mathbf{p}(1) = \mathbf{W}(1)\mathbf{p}_2 = \begin{bmatrix} -0.4 & 0.4 & 0.4 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} = -0.4 \tag{10.55}$$

从而，误差为

$$e(1) = t(1) - a(1) = t_2 - a(1) = 1 - (-0.4) = 1.4 \tag{10.56}$$

现在我们计算新的权值：

$$\begin{aligned}\mathbf{W}(2) &= \mathbf{W}(1) + 2\alpha e(1)\mathbf{p}^T(1) \\ &= [-0.4 \quad 0.4 \quad 0.4] + 2(0.2)(1.4)\begin{bmatrix}1\\1\\-1\end{bmatrix}^T = [0.16 \quad 0.96 \quad -0.16]\end{aligned} \tag{10.57}$$

下一步再次给出橘子的值：

$$a(2) = \mathbf{W}(2)\mathbf{p}(2) = \mathbf{W}(2)\mathbf{p}_1 = [0.16 \quad 0.96 \quad -0.16]\begin{bmatrix}1\\-1\\-1\end{bmatrix} = -0.64 \tag{10.58}$$

10-12

误差为

$$e(2) = t(2) - a(2) = t_1 - a(2) = -1 - (-0.64) = -0.36 \tag{10.59}$$

新的权值为：

$$\mathbf{W}(3) = \mathbf{W}(2) + 2\alpha e(2)\mathbf{p}^T(2) = [0.016 \quad 1.1040 \quad -0.0160] \tag{10.60}$$

若继续此过程，算法将收敛于

$$\mathbf{W}(\infty) = [0 \quad 1 \quad 0] \tag{10.61}$$

与第 4 章中由感知机学习规则得到的结果相比较。可以注意到，ADALINE 产生和第 3 章中为苹果/橘子问题设计的相同的判定边界。这个边界处于两个参考模式的中间。感知机规则不产生这样一个边界。这是因为，尽管一些模式可能接近于边界，一旦模式被正确地分类，感知机规则便中止了。LMS 算法使均方误差最小化，因而它尽力使判定边界远离参考模式。

### 10.2.5 自适应滤波

正如我们在本章开始时提到的，ADALINE 网络具有和感知机网络相同的限制；它只能解决线性可分问题。尽管有此缺陷，ADALINE 的应用范围仍比感知机网络广得多。事实上可以有把握地说，它是实际应用中使用最广的神经网络之一。ADALINE 的一个主要应用领域便是自适应滤波，现在它仍被广泛地使用着。本节中我们将介绍一个自适应滤波的例子。

**抽头延迟线** 为了将 ADALINE 网络用作自适应滤波器，我们先介绍一个新的构造块：抽头延迟线。图 10-4 所示为带有 $R$ 个输出的抽头延迟线。

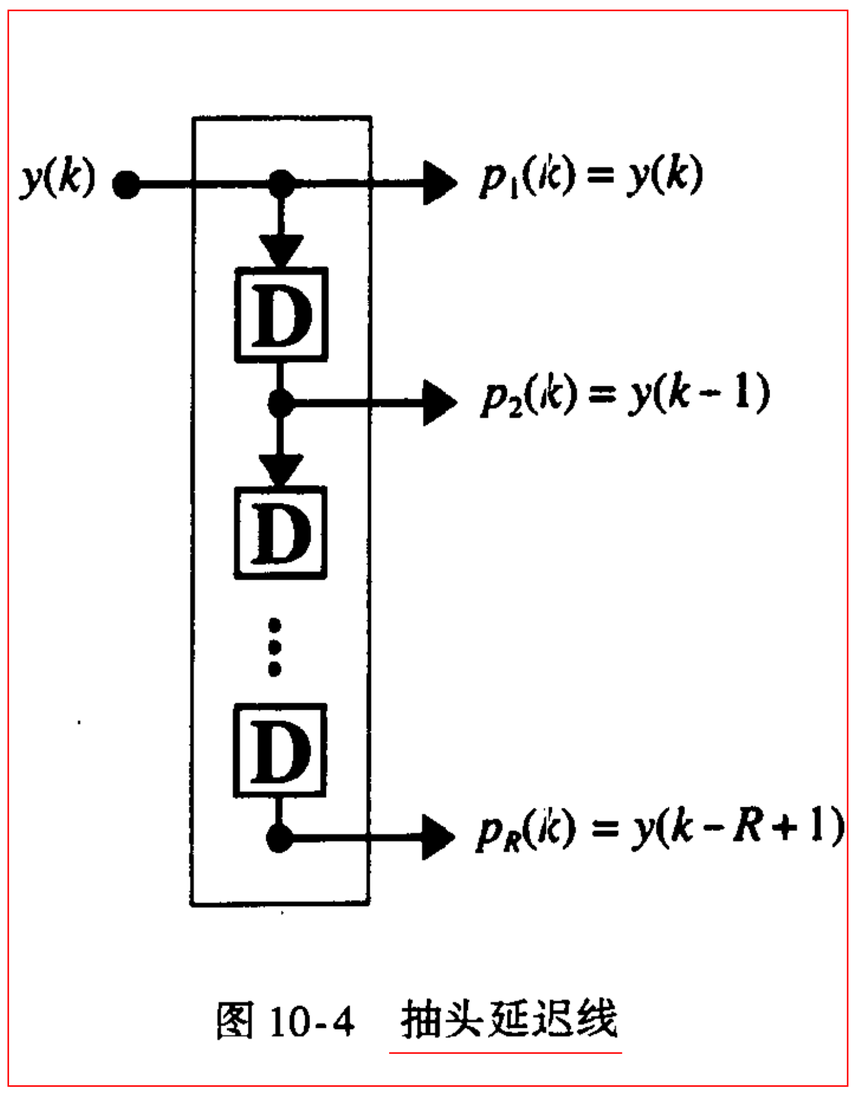



图 10-4 抽头延迟线

信号从左边输入。在延迟线的输出端是一个

10-13 $R$ 维的向量，包含当前时刻的输入信号和分别延迟了 1 到 $R-1$ 时间步的输入信号。

**自适应滤波器** 若将一个延迟线与一个 ADALINE 网络结合起来，我们就能得到一个自适应滤波器，如图 10-5 所示。滤波器的输出为：

$$a(k) = purelin(\mathbf{Wp} + b) = \sum_{i=1}^{R} w_{1,i}y(k-i+1) + b \tag{10.62}$$

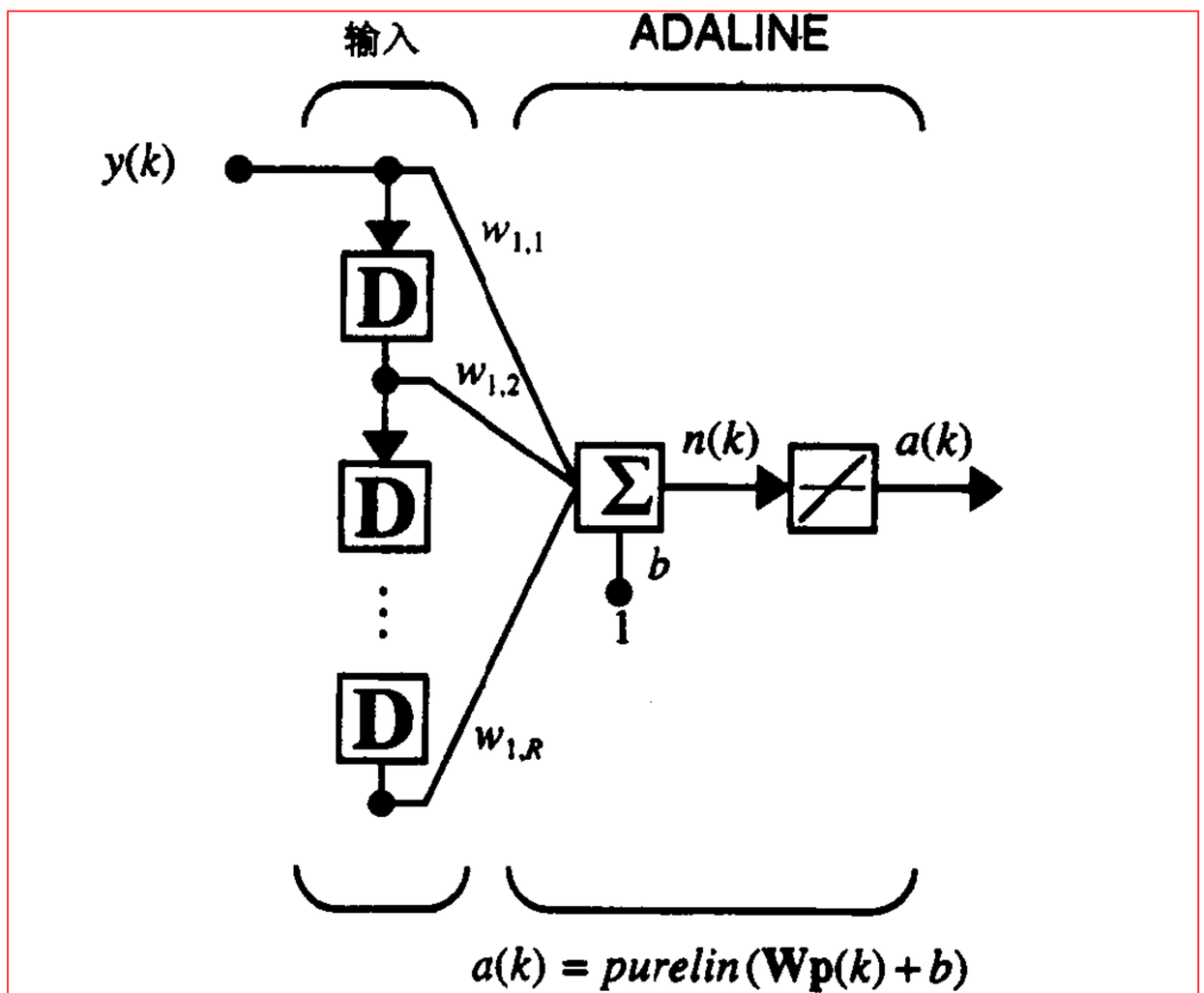



图 10-5 自适应滤波器 ADALINE

10-14 读者要是对数字信号处理熟悉的话，可以看到图 10-5 中的网络就是一个有限脉冲响应(FIR)滤波器[WiSt85]。数字信号处理领域的内容已超出本书的范围，不过我们仍可以通过一个简单但实用的例子来展示这个自适应滤波器的用处。

1. 自适应噪声消除

自适应滤波器可以通过各种各样的办法来使用。下面的例子中，我们用它来消除噪声。你最好花一点时间来看看这个例子，因为它与你所期望的有点不一样。例如，网络力求将其减至最小的输出“误差”，实际上却是我们试图要恢复的信号的近似！

假设没有一个医生正试图检查一个心烦意乱的研究生的脑电图(EEG)。他发现要看的信号混杂了 60Hz 噪声源发出的噪声。他以在线的方式检查病人，想观看到能够得到的最好信号。图 10-6 表示如何用一个自适应滤波器来除去噪声信号。

10-15 所图所示，原始 60Hz 信号样本输入到一个自适应滤波器中，并通过调整它的元件来使“误差”$e$ 达到最小。滤波器的期望输出是被干扰了的 EEG 信号 $t$。滤波器尽量复制这个被干扰了的信号，然而它仅知道初始的噪声源 $v$。因此，它只能复制 $t$ 中与 $v$ 线性相关的部分，即 $m$。结果，自适应滤波器试图模拟噪声路径滤波器，因而滤波器的输出 $a$ 将接近于干扰噪声 $m$。通过这样的途径，误差 $e$ 将接近于未被干扰的初始 EEG 信号 $s$。

在下面这个单正弦波噪声源的简单情况下，一个有两个权值和没有偏置值的神经元就足实现需要的滤波器了。滤波器的输入是噪声源的当前值和前一个值。这样有两输入的滤波器可以使噪声 $v$ 以所期望的方式被削弱和发生相移。滤波器如图 10-7 所示。

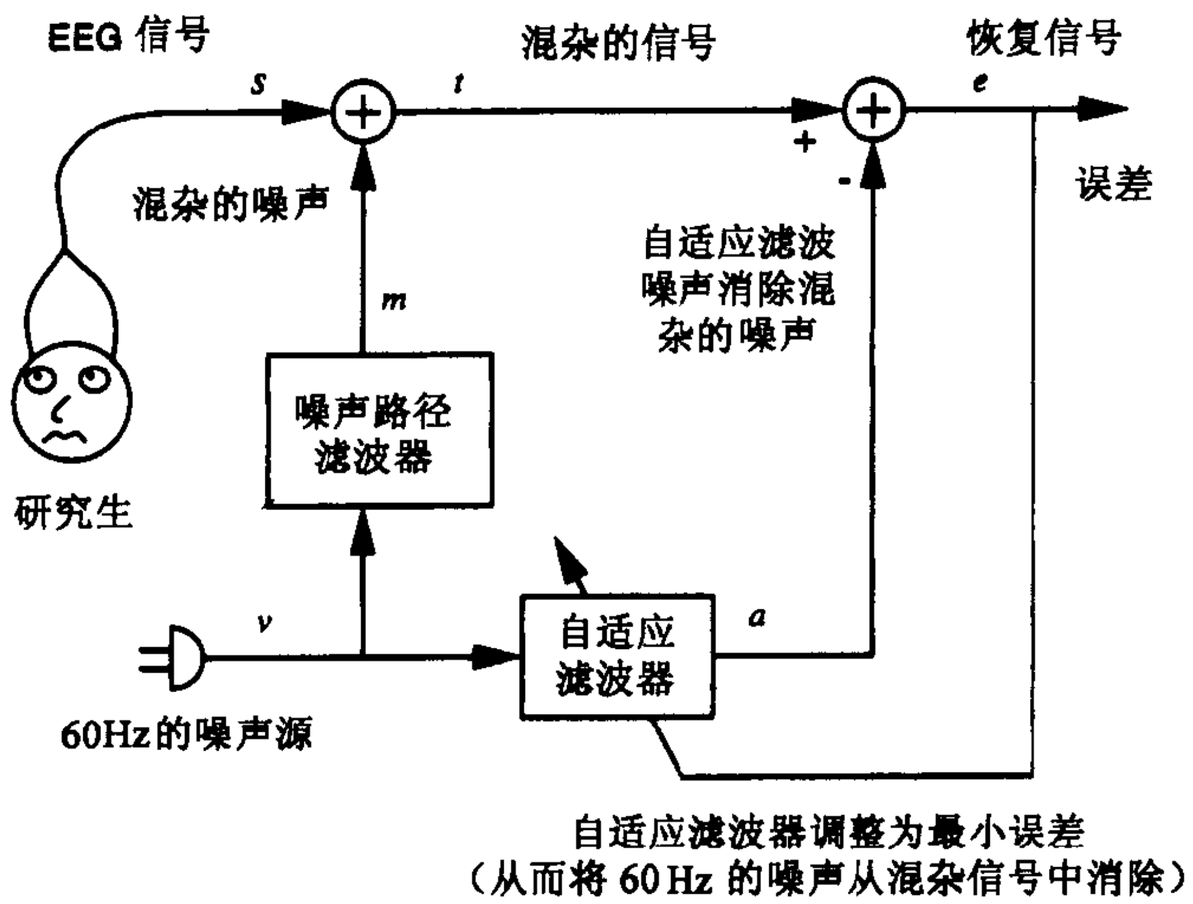



图 10-6 噪声消除系统

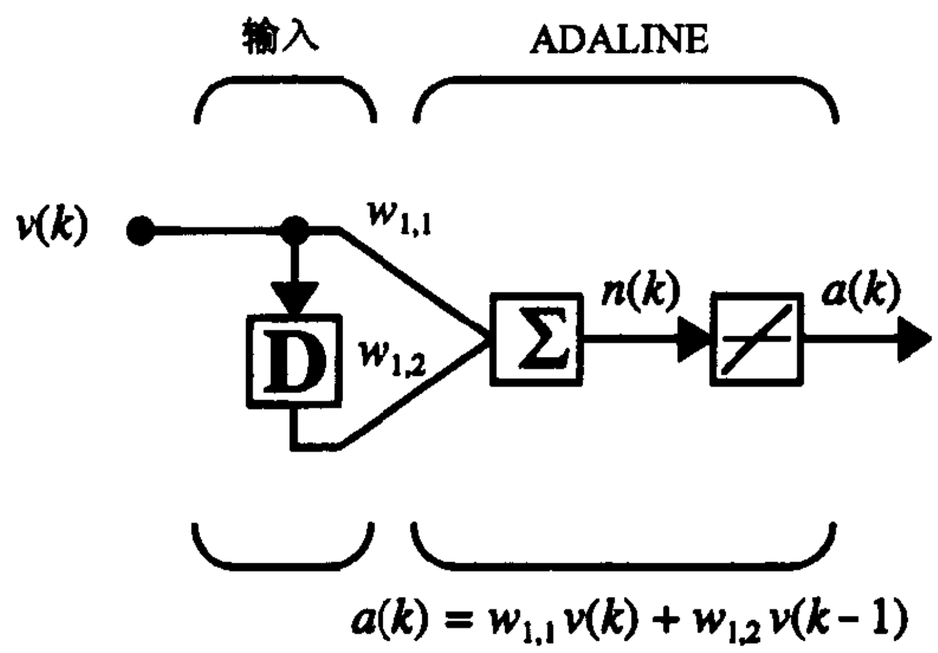



图 10-7 用于噪声消除的自适应滤波器

可以用本章中前面所得到的数学关系式来分析这个系统。首先需要得到输入相关矩阵 **R** 和输入/目标互相关向量 **h**：

$$\mathbf{R} = [\mathbf{z} \quad \mathbf{z}^T], \mathbf{h} = E[t \quad \mathbf{z}] \tag{10.63}$$

本例中，输入向量由噪声源的当前值和前一个值给出：

$$\mathbf{z}(k) = \begin{bmatrix} v(k) \\ v(k-1) \end{bmatrix} \tag{10.64}$$

而目标是当前信号和被过滤的噪声信号之和：

$$t(k) = s(k) + m(k) \tag{10.65}$$

将 **R** 和 **h** 的表达式展开可以得到：

$$\mathbf{R} = \begin{bmatrix} E[v^2(k)] & E[v(k)v(k-1)] \\ E[v(k-1)v(k)] & E[v^2(k-1)] \end{bmatrix} \tag{10.66}$$ 10-16

及

$$\mathbf{h} = \begin{bmatrix} E[(s(k) + m(k))v(k)] \\ E[(s(k) + m(k))v(k-1)] \end{bmatrix} \tag{10.67}$$

要得到这两个量的特定值，必须定义噪声信号 $v$，EEG 信号 $s$ 和被过滤的噪声 $m$。这里我们假定：EEG 信号是白的(一个时间步与下一时间步不相关)随机信号，且均匀分布于 -0.2 和 +0.2 之间，噪声源(以 180Hz 频率采样的 60Hz 正弦波)为：

$$v(k) = 1.2\sin\left(\frac{2\pi k}{3}\right) \tag{10.68}$$

干扰 EEG 信号的要被过滤的噪声是按 1/10 削减的且相移了 $\frac{\pi}{2}$ 的噪声源：

$$m(k) = 0.12\sin\left(\frac{2\pi k}{3} + \frac{\pi}{2}\right) \tag{10.69}$$

现在可计算输入相关矩阵 **R** 的各个元素：

$$E[v^2(k)] = (1.2)^2 \frac{1}{3}\sum_{k=1}^{3}\left(\sin\left(\frac{2\pi k}{3}\right)\right)^2 = (1.2)^2 0.5 = 0.72 \tag{10.70}$$

$$E[v^2(k-1)] = E[v^2(k)] = 0.72 \tag{10.71}$$

$$\begin{aligned} E[v(k)v(k-1)] &= \frac{1}{3}\sum_{k=1}^{3}\left(1.2\sin\frac{2\pi k}{3}\right)\left(1.2\sin\frac{2\pi(k-1)}{3}\right) \\ &= (1.2)^2 0.5\cos\left(\frac{2\pi}{3}\right) = -0.36 \end{aligned} \tag{10.72}$$

(这里我们使用了一些三角恒等式)。

于是 **R** 为

$$\mathbf{R} = \begin{bmatrix} 0.72 & -0.36 \\ -0.36 & 0.72 \end{bmatrix} \tag{10.73}$$

可以用类似的办法求得 **h**。首先考虑式(10.67)中上面一项：

10-17

$$E[(s(k) + m(k))v(k)] = E[s(k)v(k)] + E[m(k)v(k)] \tag{10.74}$$

因为 $s(k)$和 $v(k)$独立且均值为 0，所以右边第一项为 0。第二项也为 0；

$$E[m(k)v(k)] = \frac{1}{3}\sum_{k=1}^{3}\left(0.12\sin\left(\frac{2\pi k}{3} + \frac{\pi}{2}\right)\right)\left(1.2\sin\frac{2\pi k}{3}\right) = 0 \tag{10.75}$$

因此，**h** 的第一个元素为 0。

再考虑 **h** 的第二元素：

$$E[(s(k) + m(k))v(k-1)] = E[s(k)v(k-1)] + E[m(k)v(k-1)] \tag{10.76}$$

如同 **h** 的第一个元素，因为 $s(k)$与 $v(k-1)$独立且均值为 0，故右边第一项为 0。第二项为：

$$E[m(k)v(k-1)] = \frac{1}{3}\sum_{k=1}^{3}\left(0.12\sin\left(\frac{2\pi k}{3} + \frac{\pi}{2}\right)\right)\left(1.2\sin\frac{2\pi(k-1)}{3}\right) = -0.0624 \tag{10.77}$$

因而，**h** 为

$$\mathbf{h} = \begin{bmatrix} 0 \\ -0.0624 \end{bmatrix} \tag{10.78}$$

权值的最小均方误差由式(10.18)给出：

$$\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h} = \begin{bmatrix} 0.72 & -0.36 \\ -0.36 & 0.72 \end{bmatrix}^{-1}\begin{bmatrix} 0 \\ -0.0624 \end{bmatrix} = \begin{bmatrix} -0.0578 \\ -0.1156 \end{bmatrix} \tag{10.79}$$

现在，求得最小值时我们将得到哪种误差呢？为求出这个误差，由式(10.12)得：

$$F(\mathbf{x}) = c - 2\mathbf{x}^T\mathbf{h} + \mathbf{x}^T\mathbf{R}\mathbf{x} \tag{10.80}$$

我们已求得 $\mathbf{x}^*$，$\mathbf{h}$ 和 $\mathbf{R}$，因此只需求 $c$：

$$\begin{aligned} c = E[t^2(k)] &= E[(s(k) + m(k))^2] \\ &= E[s^2(k)] + 2E[s(k)m(k)] + E[m^2(k)] \end{aligned} \tag{10.81}$$

10-18

由于 $s(k)$与 $m(k)$独立且均值为 0，因而中间项为 0。第一项为随机号的期望值，计算如下：

$$E[s^2(k)] = \frac{1}{0.4}\int_{-0.2}^{0.2} s^2 ds = \frac{1}{3(0.4)}s^3\Big|_{-0.2}^{0.2} = 0.0133 \tag{10.82}$$

被过滤的噪声的均方值为

$$E[m^2(k)] = \frac{1}{3}\sum_{k=1}^{3}\left(0.12\sin\left(\frac{2\pi}{3} + \frac{\pi}{2}\right)\right)^2 = 0.0072 \tag{10.83}$$

从而

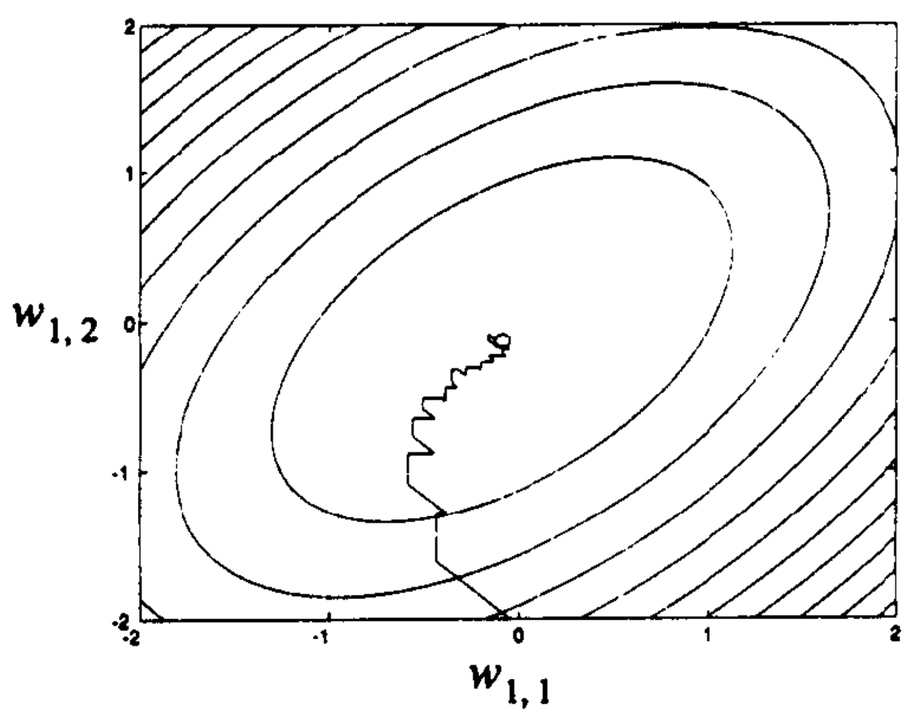
$$c = 0.0133 + 0.0072 = 0.0205 \tag{10.84}$$

将 $\mathbf{x}^*$，$\mathbf{h}$ 和 $\mathbf{R}$ 代入式(10.80)中得到最小均方误差

$$F(\mathbf{x}^*) = 0.0205 - 2(0.0072) + 0.0072 = 0.0133 \tag{10.85}$$

最小均方误差与 EEG 信号的均方值相同。这正是我们所期望的，因为这个自适应噪声消除器的“误差”事实上是被恢复的 EEG 信号。

图 10-8 说明了学习速度 $\alpha = 0.1$ 时 LMS 算法在权值空间中的轨迹。在这个模拟中，初始时系统的权值 $w_{1,1}$和 $w_{1,2}$分别被随意地设为 0 和 $-2$。从图中可以看到，LMS 的轨迹看起来像有噪声时的最速下降法。




10-19

图 10-8　$\alpha = 0.1$ 时 LMS 的轨迹

注意在图中，轮廓线表示了赫森矩阵($\mathbf{A} = 2\mathbf{R}$)的特征值和特征向量为：

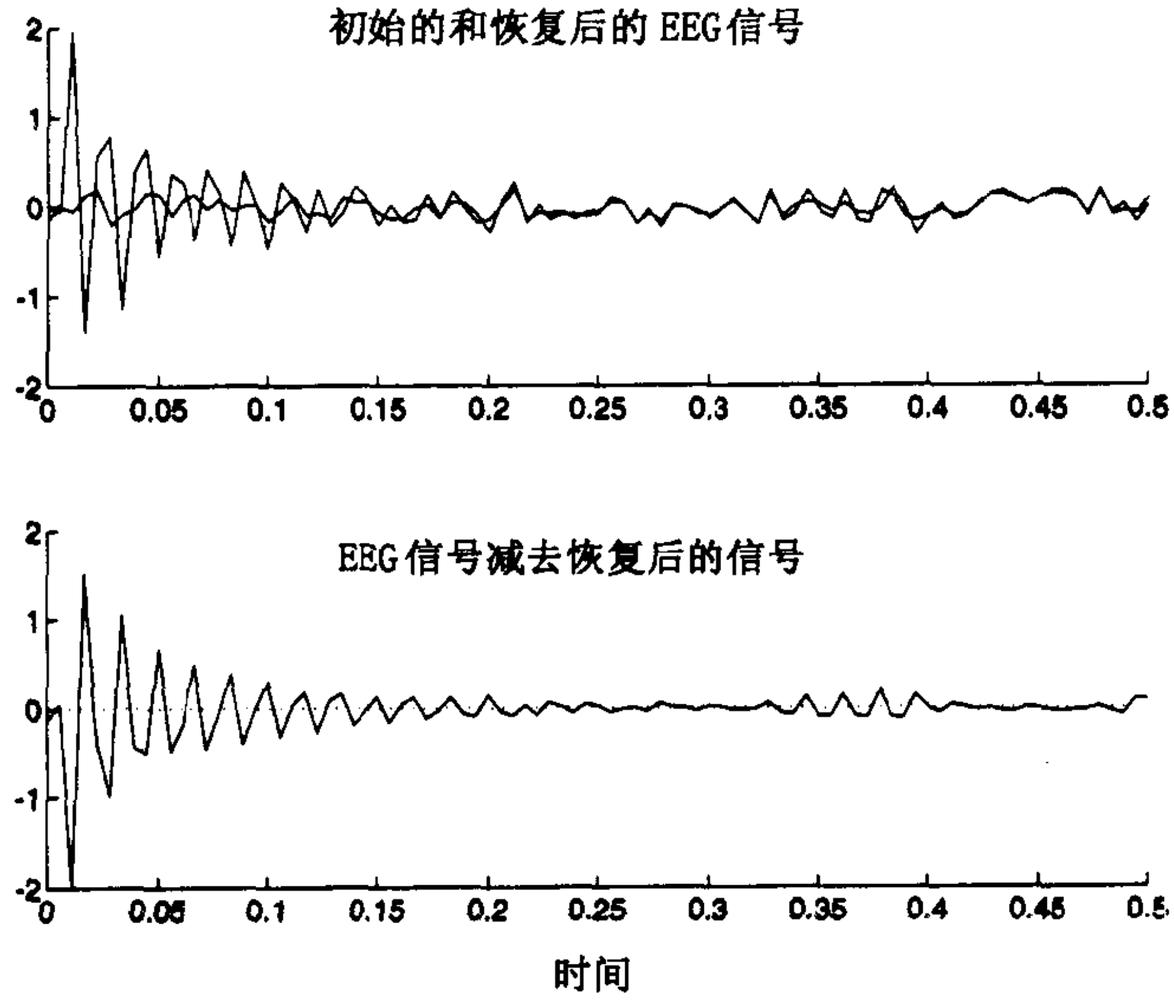
$$\lambda_1 = 2.16, \mathbf{z}_1 = \begin{bmatrix} -0.7071 \\ 0.7071 \end{bmatrix}, \lambda_2 = 0.72, \mathbf{z}_2 = \begin{bmatrix} -0.7071 \\ -0.7071 \end{bmatrix} \tag{10.86}$$

(参考第 8 章中对赫森矩阵的特征系统的讨论。)

若学习速度减小，LMS 的轨迹将比图 10-8 中的更光滑，但学习过程进行得更慢；若学

习速度增加，轨迹将带更多的锯齿状且呈现振荡。事实上，如本章开始时所述，若学习速度增加太大，系统将根本不收敛。最大稳态学习速度为 $\alpha < 2/2.16 = 0.926$。

为了判别噪声消除器的性能，考虑图 10-9。这幅图说明了滤波器如何自适应以消除噪声。上面的图为恢复后和初始时的 EEG 信号。开始时，恢复后的信号与初始的 EEG 信号极不相似，滤波器用了约 0.2 秒($\alpha = 0.1$)的时间作调整，给出一个可接受的恢复信号。实验的后半段中，初始信号和恢复后的信号之间的均方差为 0.002。与信号的均方值 0.0133 相比，这个结果是不错的。初始信号和恢复后的信号之间的差表示了下面的图中。




10-20

图 10-9 干扰噪声的自适应过滤器消除

你也许会奇怪误差为什么不变为 0。这是因为 LMS 算法是一个近似的最速下降法；它使用梯度的估计值而不是真正的梯度值来更新网络权值。梯度的估计值是有噪声的梯度值。这使得即使均方误差达到最小时，权值仍会继续作小小的改变。从图 10-8 中可看到此效应。

试验使用此自适应噪声消除滤波器请用 *Neural Network Design Demonstration Adaptive Noise Cancellation*(**nnd10nc**)。一个更复杂的噪声源和实际的 EEG 数据用在 *Electroencephalogram Noise Cancellation* (**nnd10eeg**)的演示中。

2. 回声消除

自适应噪声消除的另一个更重要的实际应用是回声消除。在“混合”设备中阻抗的不匹配会在长途电话线和用户的本地线之间形成接头，这使得长途电话线上的回声很普遍。在打国际电话时你可能就感觉过这类效应。

图 10-10 说明了如何用一个自适应噪声消除滤波器来减少这些回声[WiWi85]。在长途线的末端，到来的信号被送到一个含有自适应滤波器的混合设备。滤波器的目标输出是混合设备的输出，因而滤波器将消除混合输出中与输入信号相关的那部分信号，即回声。

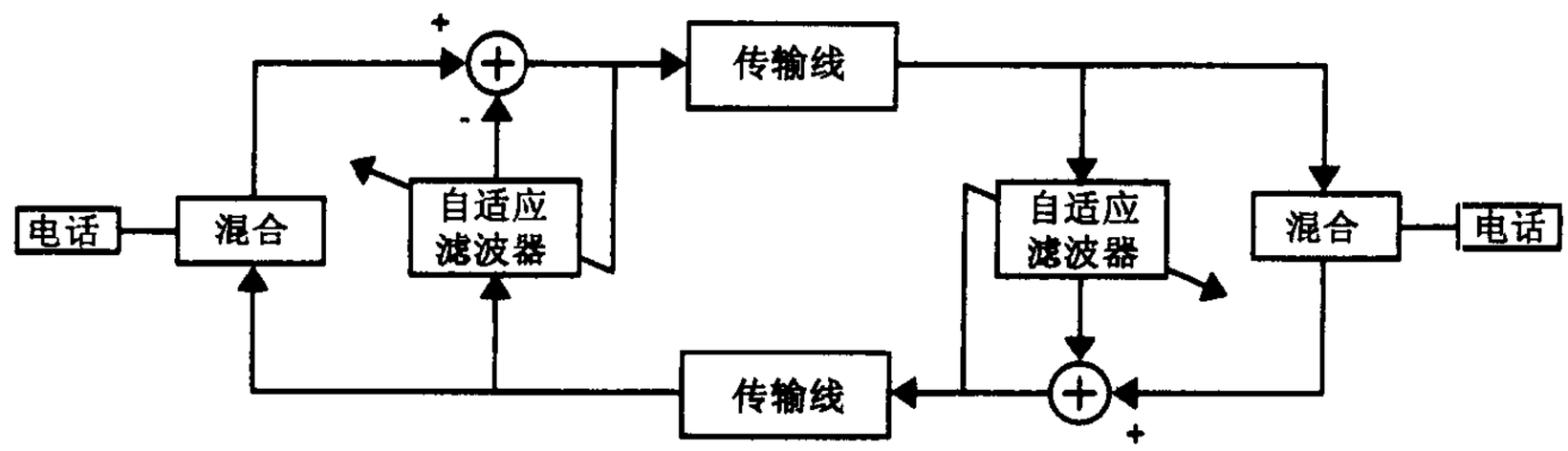
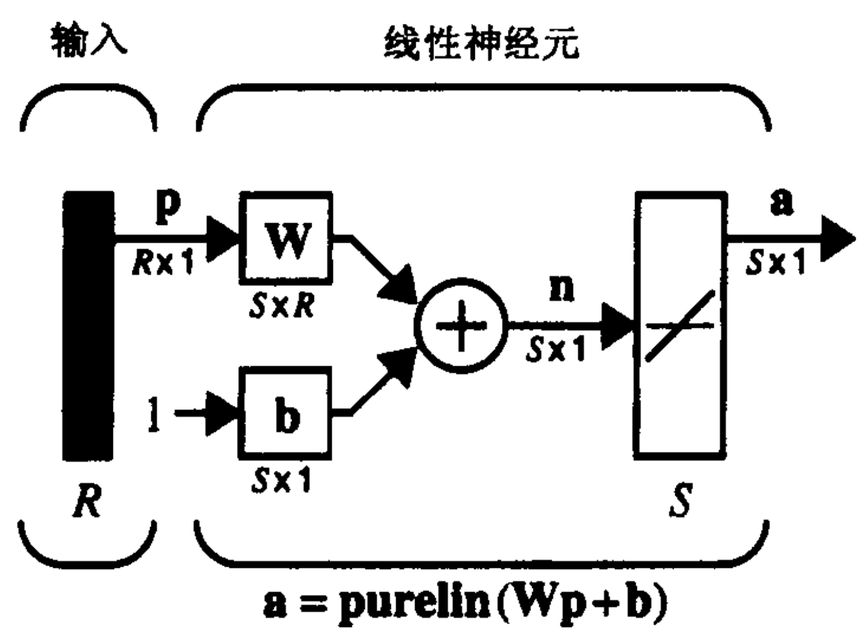



图 10-10 回声消除系统

10-21

## 10.3 小结

### ADALINE




**均方误差**

$$F(\mathbf{x}) = E[e^2] = E[(t-a)^2] = E[(t-\mathbf{x}^T\mathbf{z})^2]$$

$$F(\mathbf{x}) = c - 2\mathbf{x}^T\mathbf{h} + \mathbf{x}^T\mathbf{R}\mathbf{x}$$

$$c = E[t^2], \quad \mathbf{h} = E[t\mathbf{z}] \quad 且 \quad \mathbf{R} = E[\mathbf{z}\mathbf{z}^T]$$

惟一的最小值若存在，则为 $\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h}$。这里 $\mathbf{x} = \begin{bmatrix} {}_1\mathbf{w} \\ b \end{bmatrix}$ 且 $\mathbf{z} = \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}$

**LMS 算法**

$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha\mathbf{e}(k)\mathbf{p}^T(k)$$

$$\mathbf{b}(k+1) = \mathbf{b}(k) + 2\alpha\mathbf{e}(k)$$

**收敛点**

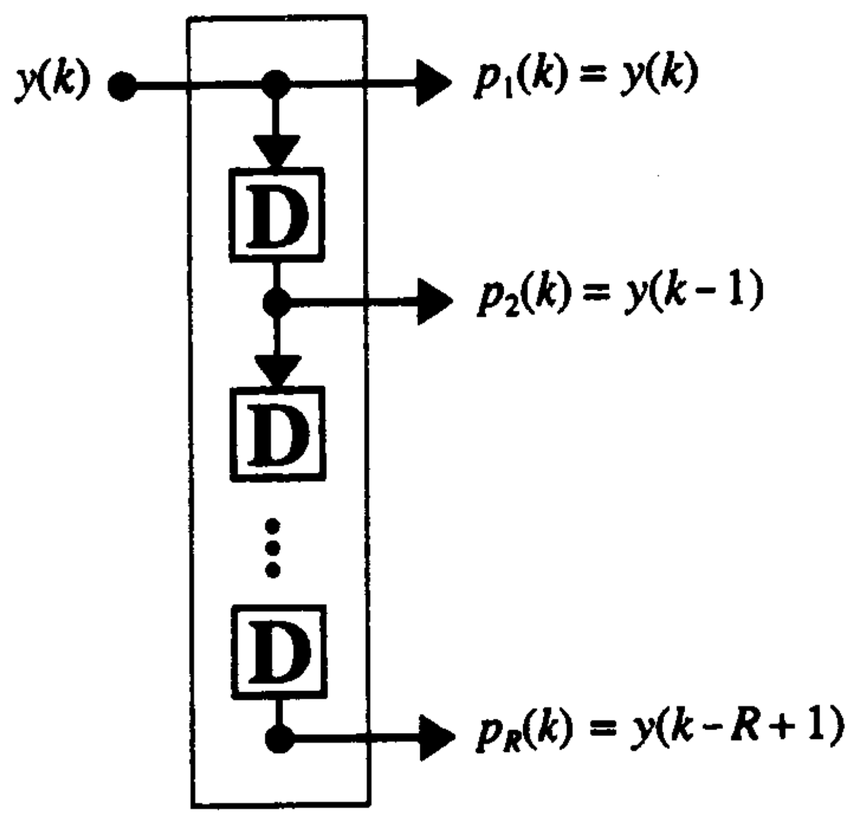
$$\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h}$$

10-22

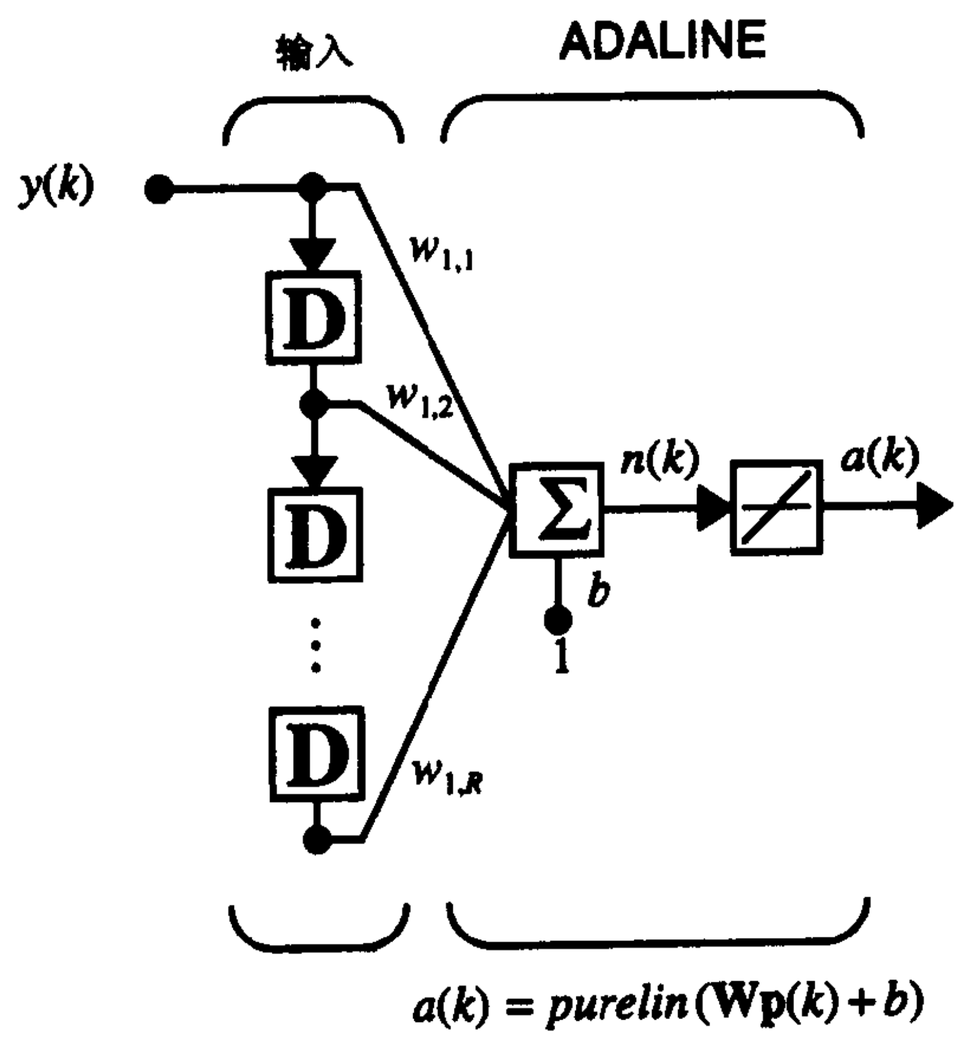
**稳点学习速度**

$$0 < \alpha < 1/\lambda_{max} (\lambda_{max} 是 \mathbf{R} 的最大特征值)$$

**抽头延迟线**




**自适滤波器 ADALINE**




10-23
$$a(k) = purelin(\mathbf{Wp} + b) = \sum_{i=1}^{R} w_{1,i}\, y(k - i + 1) + b$$

## 10.4 例题

**P10.1** 考虑图 10-11 中的 ADALINE 滤波器。

假定

$$w_{1,1} = 2, \quad w_{1,2} = -1, \quad w_{1,3} = 3$$

且输入序列为

$$\{y(k)\} = \{\cdots,0,0,0,5,-4,0,0,0,\cdots\}$$

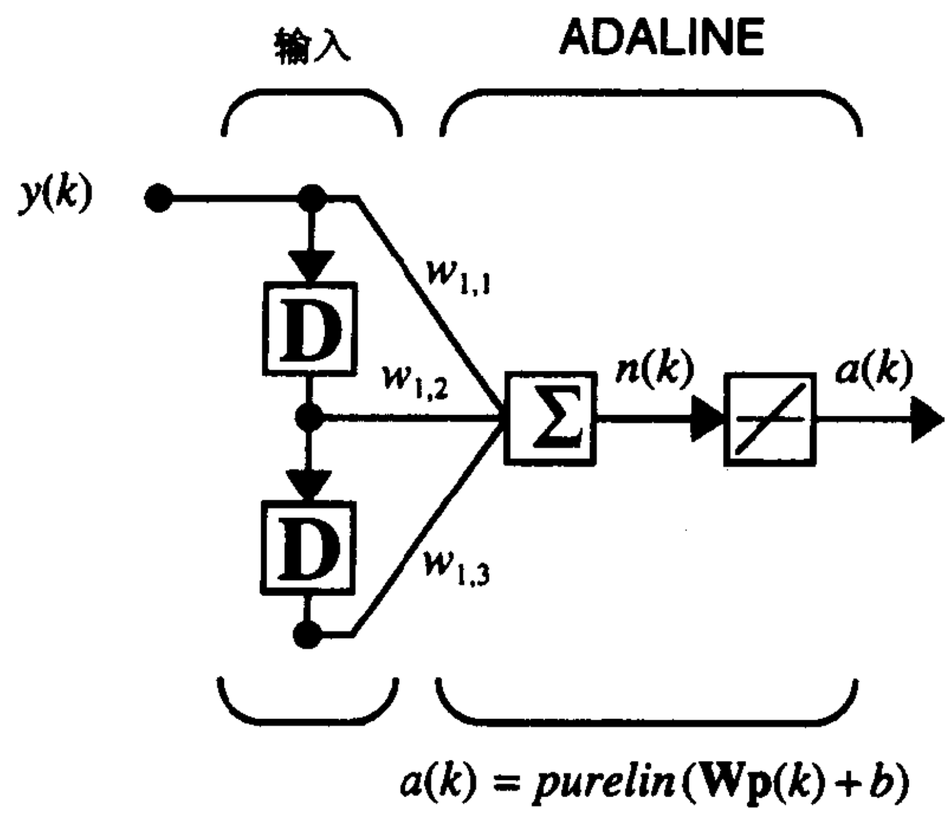



图 10-11 ADALINE 滤波器

其中 $y(0)=5$，$y(1)=-4$，等等。

（i）$k=0$ 以前滤波器的输出是什么？

（ii）从 $k=0$ 到 $k=5$，滤波器的输出是什么？

（iii）$y(0)$对输出的影响有多长时间？

**解**

（i）在 $k=0$ 以前输入了 3 个 0，因而输出为 0。

（ii）在 $k=0$ 时，数字“5”被输入滤波器，它将被乘以 $w_{1,1}$，其值为 2，因而 $a(0)=10$。这可以通过矩阵操作得到： 10-24

$$a(0)=\mathbf{W}\mathbf{p}(0)=\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \end{bmatrix}\begin{bmatrix} y(0) \\ y(-1) \\ y(-2) \end{bmatrix}=\begin{bmatrix} 2 & -1 & 3 \end{bmatrix}\begin{bmatrix} 5 \\ 0 \\ 0 \end{bmatrix}=10$$

类似地，可以计算下一个输出为

$$a(1)=\mathbf{W}\mathbf{p}(1)=\begin{bmatrix} 2 & -1 & 3 \end{bmatrix}\begin{bmatrix} -4 \\ 5 \\ 0 \end{bmatrix}=-13$$

$$a(2)=\mathbf{W}\mathbf{p}(2)=\begin{bmatrix} 2 & -1 & 3 \end{bmatrix}\begin{bmatrix} 0 \\ -4 \\ 5 \end{bmatrix}=19$$

$$a(3)=\mathbf{W}\mathbf{p}(3)=\begin{bmatrix} 2 & -1 & 3 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ -4 \end{bmatrix}=-12$$

$$a(4)=\mathbf{W}\mathbf{p}(4)=\begin{bmatrix} 2 & -1 & 3 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}=0$$

其余的输出将为 0。

（iii）$y(0)$的影响从 $k=0$ 持续到 $k=2$，因此它将影响 3 个时间区间。这对应于这个滤波器的脉冲响应时间长度。

**P10.2** 假定要设计一个 ADALINE 网络来区分输入向量的不同类别。首先使用如下的类别：

类别Ⅰ：$\mathbf{p}_1 = [1 \quad 1]^T$ 且 $\mathbf{p}_2 = [-1 \quad -1]^T$

类别Ⅱ：$\mathbf{p}_3 = [2 \quad 2]^T$

（**i**）能否设计一个 ADALINE 网络来作这样一个区分？

（**ii**）若对(i)题的回答为“是”，那么什么权值和偏差集合可被使用？

再考虑下面不同的类别：

10-25 类别Ⅲ：$\mathbf{p}_1 = [1 \quad 1]^T$ 且 $\mathbf{p}_2 = [1 \quad -1]^T$

类别Ⅳ：$\mathbf{p}_3 = [1 \quad 0]^T$

（**iii**）能否设计一个 ADALINE 网络来作这样一个区分？

（**iv**）若对(iii)题的回答为“是”，则可以使用什么权值和偏置值集合？

**解**

（**i**）输入向量画在图 10-12 中。

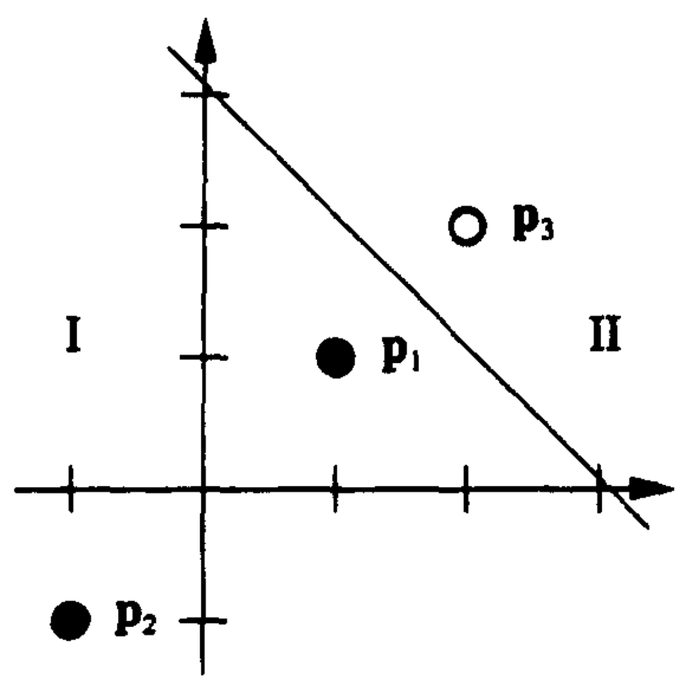



图 10-12 例题 P10.2 (i)的输入向量

图中的直线是成功区分这两个类别的判定边界。由于它们是线性可分的，因而 ADALINE 可以完成此任务。

（**ii**）判定边界经过点(3，0)和(0，3)。这两点就是交点 $-b/w_{1,1}$ 和 $-b/w_{1,2}$。因此，下面的解可满足要求：

$$b = 3, w_{1,1} = -1, w_{1,2} = -1$$

注意，若 ADALINE 的输出为正或零，则输入向量为类别Ⅰ的；若输出为负，则输入向量为类另Ⅱ的。这个解也提供误差，因为判定边界分开 $p_1$ 和 $p_3$ 之间的线。

（**iii**）被区分的输入向量如图 10-13 中所示。图中的向量不是线性可分的，因此 ADALINE 网络不能对它们进行区分。

10-26 （**iv**）如(iii)题中所述，ADALINE 不能完成任务，因此没有满足要求的权值和偏置值集合。

**P10.3** 假定有如下的输入/目标输出对：

$$\left\{\mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_1 = 1\right\}, \left\{\mathbf{p}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, t_2 = -1\right\}$$

这些模式以相等的概率产生，它们可用来训练一个无偏置值的 ADALINE 网络。均方误差的性能曲面大体是什么？

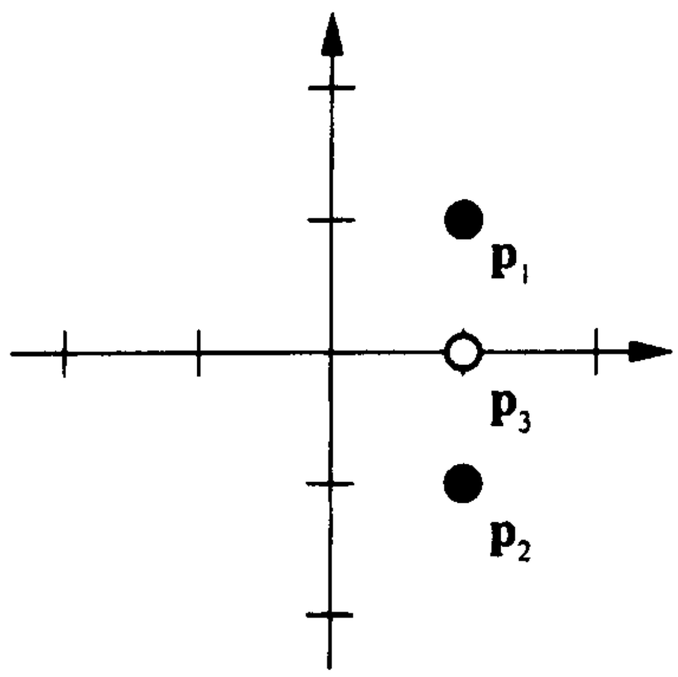



图 10-13　例题 P10.2(iii)的输入向量

**解**

首先我们需要计算二次函数的各项。由式(10.11)，性能指数函数为

$$F(\mathbf{x}) = c - 2\mathbf{x}^T\mathbf{h} + \mathbf{x}^T\mathbf{R}\mathbf{x}$$

因此需要计算 $c$，$\mathbf{h}$ 和 $\mathbf{R}$。

每个输入出现的概率为 0.5，因此每个目标输出的概率也为 0.5。于是，目标输出的平方的期望值为

$$c = E[t^2] = (1)^2(0.5) + (-1)^2(0.5) = 1$$

类似地，输入和目标输出之间的相互关系为

$$\mathbf{h} = E(t\mathbf{z}) = (0.5)(1)\begin{bmatrix}1\\1\end{bmatrix} + (0.5)(-1)\begin{bmatrix}1\\-1\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix}$$

最后，输入相关矩阵 $\mathbf{R}$ 为 10-27

$$\mathbf{R} = E[\mathbf{z}\mathbf{z}^T] = \mathbf{p}_1\mathbf{p}_1^T(0.5) + \mathbf{p}_2\mathbf{p}_2^T(0.5)$$

$$= (0.5)\left[\begin{bmatrix}1\\1\end{bmatrix}[1 \quad 1] + \begin{bmatrix}1\\-1\end{bmatrix}[1 \quad -1]\right] = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$$

因此，均方误差的性能指数函数为

$$F(\mathbf{x}) = c - 2\mathbf{x}^T\mathbf{h} + \mathbf{x}^T\mathbf{R}\mathbf{x}$$

$$= 1 - 2[w_{1,1} \quad w_{1,2}]\begin{bmatrix}0\\1\end{bmatrix} + [w_{1,1} \quad w_{1,2}]\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\begin{bmatrix}w_{1,1}\\w_{1,2}\end{bmatrix}$$

$$= 1 - 2w_{1,2} + w_{1,1}^2 + w_{1,2}^2$$

$F(\mathbf{x})$的赫森矩阵等于 $2\mathbf{R}$，其两个特征值均为 2。因此，性能曲面的轮廓线将是圆。为了找到轮廓线的中心(极小点)，需要解方程(10.18)：

$$\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h} = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}^{-1}\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix}$$

因此，极小点在 $w_{1,1}=0$，$w_{1,2}=1$。结果的均方误差性能曲面如图 10-14 所示。 10-28

**P10.4** 再次考虑例题 P10.3 中系统。使用 LMS 算法对网络进行训练，初始值设为 0，学习速度设为 $\alpha=0.25$。在训练中每个参考模式只使用一次。在每步中画出判定边界。

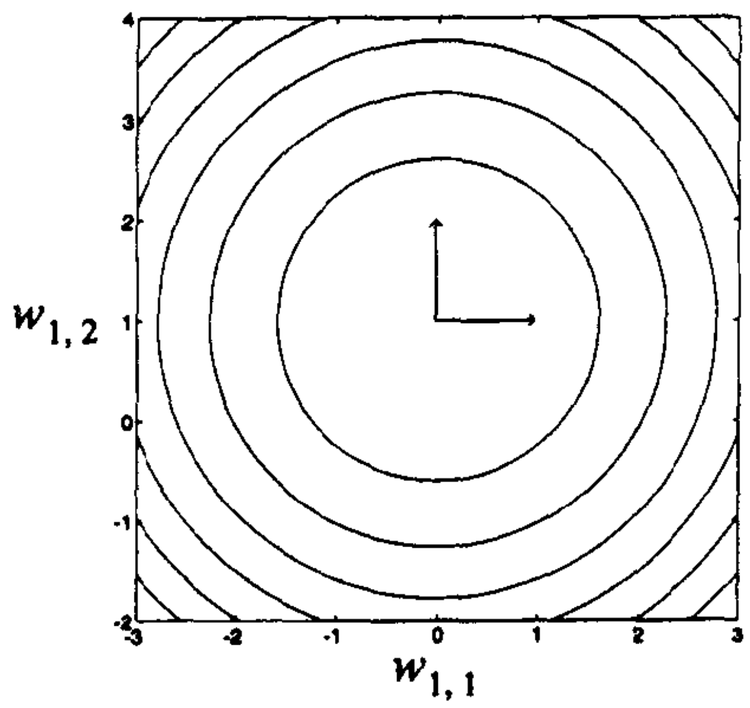



图 10-14 例题 P10.3 中 $F(\mathbf{x})$的轮廓线

**解**

假定首先输入 $\mathbf{p}_1$。网络输出、误差和新的权值计算如下：

$$a(0) = purelin\left[\begin{bmatrix}0 & 0\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\right] = 0$$

$$e(0) = t(0) - a(0) = 1 - 0 = 1$$

$$\mathbf{W}(1) = \mathbf{W}(0) + 2\alpha e(0)\mathbf{p}(0)^T = \begin{bmatrix}0 & 0\end{bmatrix} + 2\left(\frac{1}{4}\right)(1)\begin{bmatrix}1 & 1\end{bmatrix} = \begin{bmatrix}\frac{1}{2} & \frac{1}{2}\end{bmatrix}$$

与这些权值关联的判定边界如图 10-15 所示。下面输入第二个输入向量：

$$a(1) = purelin\left\{\begin{bmatrix}\frac{1}{2} & \frac{1}{2}\end{bmatrix}\begin{bmatrix}1\\-1\end{bmatrix}\right\} = 0$$

$$e(1) = t(1) - a(1) = -1 - 0 = -1$$

$$\mathbf{W}(2) = \mathbf{W}(1) + 2\alpha e(1)\mathbf{p}(1)^T = \begin{bmatrix}\frac{1}{2} & \frac{1}{2}\end{bmatrix} + 2\left(\frac{1}{4}\right)(-1)\begin{bmatrix}1 & -1\end{bmatrix} = \begin{bmatrix}0 & 1\end{bmatrix}$$

与这些权值相关联的判定边界如图 10-16 所示。这个边界表明真正的约定。它正好处于输入向量的中间。可以验证，当输入每一个输入向量，网络将产生正确的目标输出。（若交换与两个输入向量相关联的目标输出，什么权值集合是最优的?）

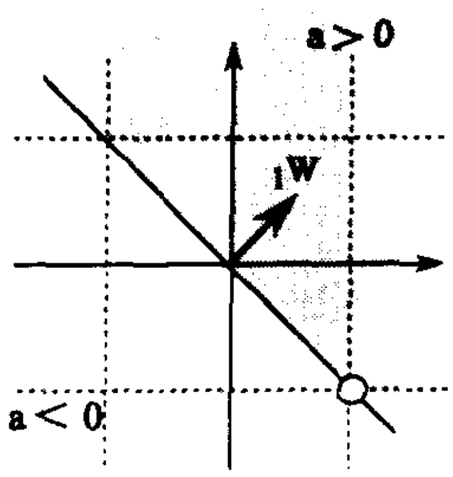



图 10-15

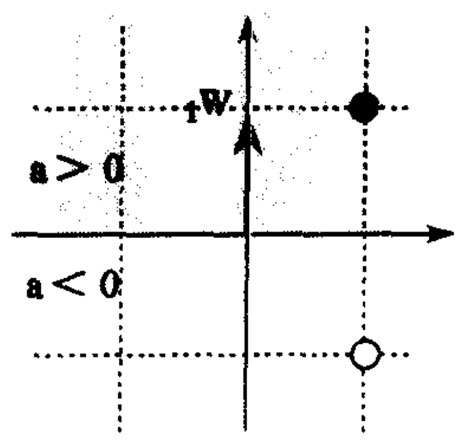



图 10-16

**P10.5** 考虑例题 P10.3 和 P10.4 中系统的收敛性。LMS 算法的最大稳态学习速度是多少?

**解**

LMS 的收敛性由学习速度 $\alpha$ 所决定，它不应超过 **R** 的最大特征值的倒数。我们可用

MATAB 找到这些特征值来确定此速度限制。 10-29

```
[V, D] = eig(R)
V =
        1     0
        0     1
D =
        1     0
        0     1
```

矩阵 D 的对角线元素给出了特征值 1 和 1，矩阵 V 的列为特征向量。顺便注意，特征向量的方向与图 10-14 中所示的相同。

最大特征值为 $\lambda_{max}=1$，它设定了学习速度的上限：

$$\alpha < 1/\lambda_{max} = 1/1 = 1$$

前一例题中建议的学习速度为 0.25，你(也许)可发以发现 LMS 算法收敛得很快。当学习速度为 1.0 甚至更大时会发生什么情况呢?

**P10.6** 考虑图 10-17 中的自适应 ADALINE 自适应滤波器。这个滤波器的目的是从前两个值中预测输入信号的下一个值。假定输入信号是一个稳态随机过程，其自相关函数为：

$$C_y(n) = E[y(k)y(k+n)]$$

$$C_y(0) = 3, \quad C_y(1) = -1, \quad C_y(2) = -1$$

(i) 画出性能指数(均方误差)的轮廓线图。

(ii) LMS 算法的最大稳态学习速度($\alpha$)是多少?

(iii) 假定 $\alpha$ 的值很小。从初始值 $\mathbf{W}(0)=[0.75 \quad 0]^T$ 开始，画出 LMS 算法中权值的变化路径图，解释画出此路径的过程。 10-30

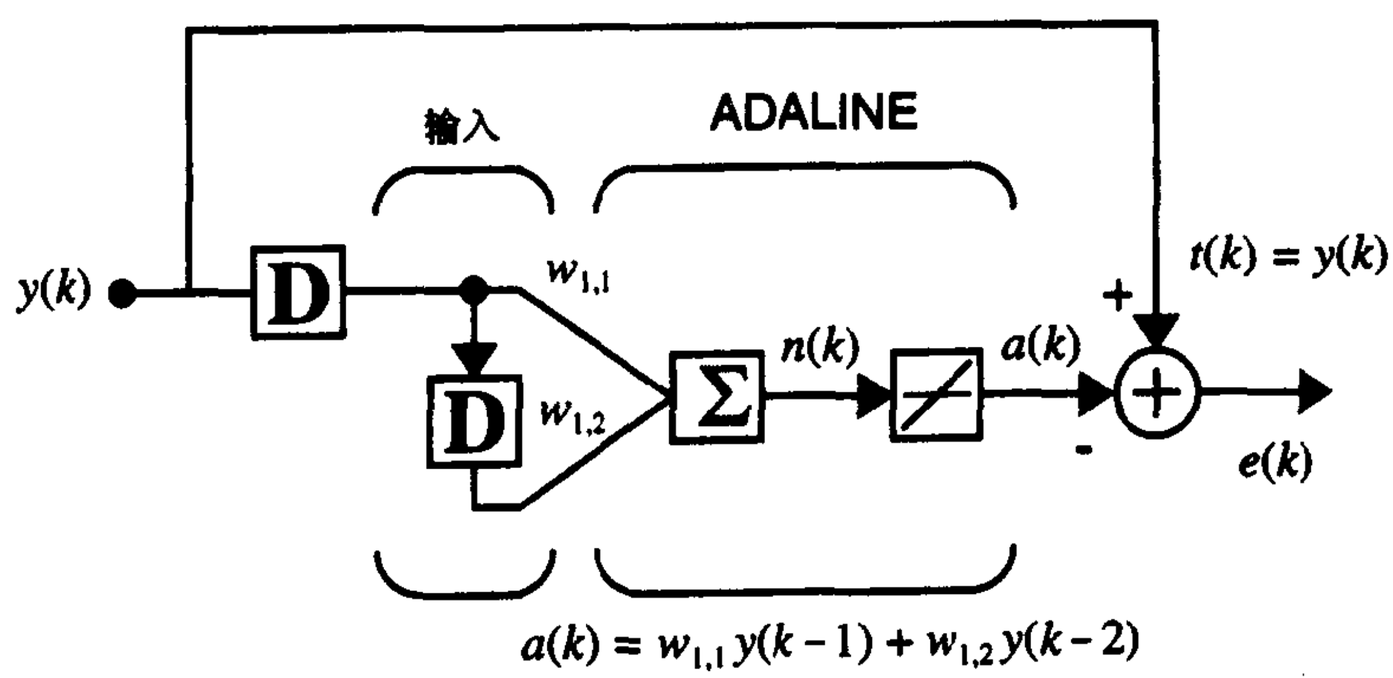



图 10-17 自适应预测器

**解**

(i) 首先需要求解性能指数以及赫森矩阵的特征值和特征向量。注意到输入向量为

$$\mathbf{z}(k) = \mathbf{p}(k) = \begin{bmatrix} y(k-1) \\ y(k-2) \end{bmatrix}$$

考虑性能指数。由(10.12)式得

$$F(\mathbf{x}) = c - 2\mathbf{x}^T\mathbf{h} + \mathbf{x}^T\mathbf{R}\mathbf{x}$$

性能指数中的常量计算如下：

$$c = E[t^2(k)] = E[y^2(k)] = C_y(0) = 3$$

$$\mathbf{R} = E[\mathbf{z}\mathbf{z}^T] = E\begin{bmatrix} y^2(k-1) & y(k-1)y(k-2) \\ y(k-1)y(k-2) & y^2(k-2) \end{bmatrix}$$

$$= \begin{bmatrix} C_y(0) & C_y(1) \\ C_y(1) & C_y(0) \end{bmatrix} = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}$$

$$\mathbf{h} = E[t\ \mathbf{z}] = E\begin{bmatrix} y(k)y(k-1) \\ y(k)y(k-2) \end{bmatrix} = \begin{bmatrix} C_y(1) \\ C_y(2) \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

10-31 最优的权值为

$$\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h} = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}^{-1}\begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 3/8 & 1/8 \\ 1/8 & 3/8 \end{bmatrix}\begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} -1/2 \\ -1/2 \end{bmatrix}$$

赫森矩阵为

$$\nabla^2 F(\mathbf{x}) = \mathbf{A} = 2\mathbf{R} = \begin{bmatrix} 6 & -2 \\ -2 & 6 \end{bmatrix}$$

现在可求得特征值：

$$|\mathbf{A} - \lambda\mathbf{I}| = \begin{bmatrix} 6-\lambda & -2 \\ -2 & 6-\lambda \end{bmatrix} = \lambda^2 - 12\lambda + 32 = (\lambda - 8)(\lambda - 4)$$

于是

$$\lambda_1 = 4, \lambda_2 = 8$$

用

$$[\mathbf{A} - \lambda\mathbf{I}]\mathbf{v} = 0$$

求特征向量对 $\lambda_1 = 4$，

$$\begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix}\mathbf{v}_1 = 0, \quad \mathbf{v}_1 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

对 $\lambda_2 = 8$，

$$\begin{bmatrix} -2 & -2 \\ -2 & -2 \end{bmatrix}\mathbf{v}_2 = 0, \quad \mathbf{v}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

因此，$F(\mathbf{x})$的轮廓线将是椭圆，每个椭圆的长轴沿着第一个特征向量的方向，因为第一个特征值的值最小。椭圆的中心为 $\mathbf{x}^*$，如图 10-18 所示。

10-32

可以写一个 MATLABM M-file 文件画出 $F(\mathbf{x})$的轮廓线图来检验此结果。

（ii）最大稳态学习速度是 $\mathbf{R}$ 的最大特征值的倒数，也是赫森矩阵$\nabla^2 F(\mathbf{x}) = \mathbf{A}$ 的最大特征值的倒数的两倍：

$$\alpha < 2/\lambda_{max} = 2/8 = 0.25$$

（iii）LMS 算法接近于最速下降法，因此，对小的学习速度，轨迹线将与轮廓线垂直，如图 10-19 所示。

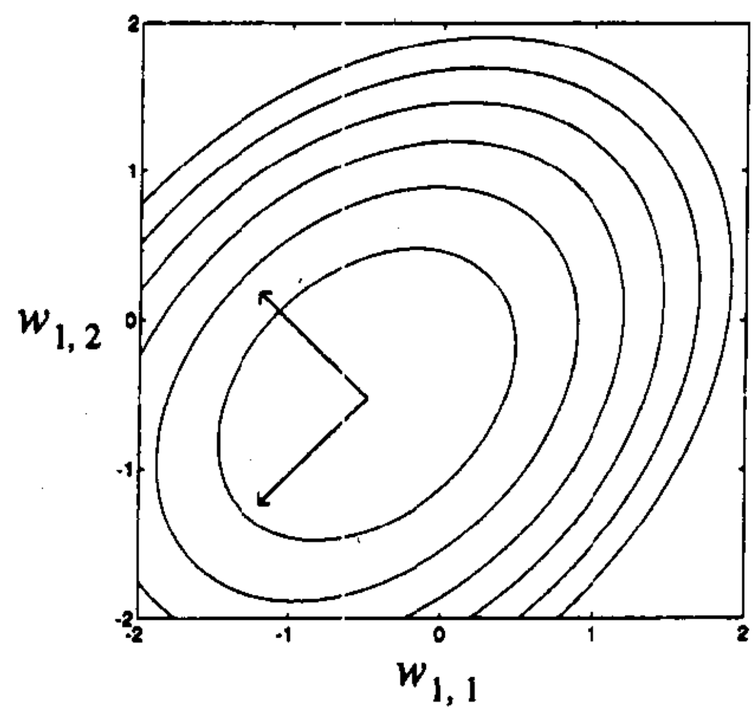



图 10-18 例题 P10.6 中的误差轮廓线

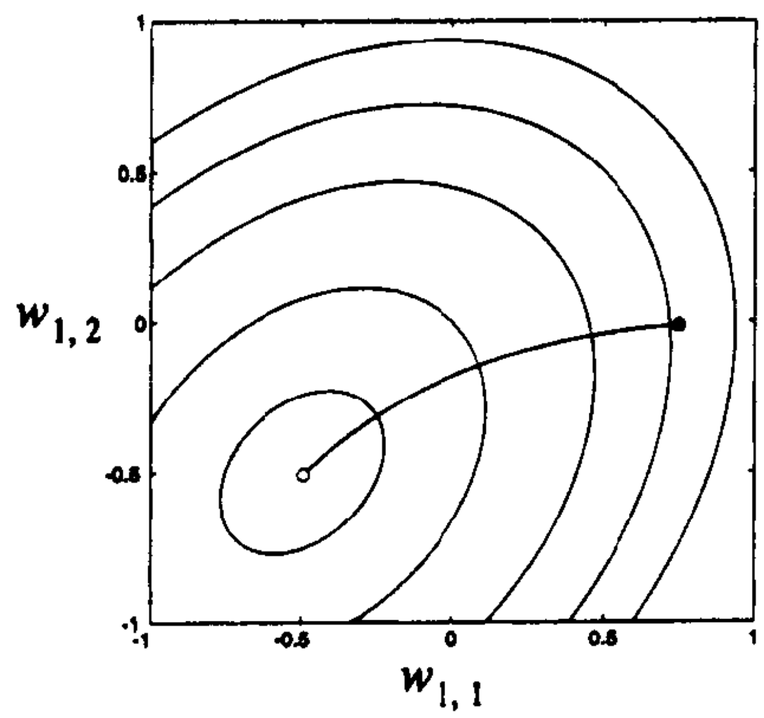



图 10-19 LMS 权值的轨迹

10-33

**P10.7** 一个飞机中的飞行员正通过飞机座舱中的麦克风讲话。由于飞行员的话音信号被到达麦克风的飞机发动机噪声所干扰，控制塔内的空中交通控制员不能接收到正确的话音。你能设计一个自适应的 ADALINE 滤波器，从而帮助减小控制塔收到的信号的噪声吗？解释你的系统。

**解**

输入到麦克风中的发动机噪声可以通过图 10-20 中的自适应过滤系统减小到最低限度。通过座舱中的一个麦克风，发动机噪声的一个样本被输入到一个自适应滤波器中。滤波器期望的输出值是从飞行员的麦克风来的被干扰了的信号。滤波器试图将“误差”信号减至最小。它能做的只是将被干扰了的信号中与发动机噪声线性相关的部分减去(假定发动机噪声和飞行员的话音不相关)。尽管发动机噪声和飞行员的声音信号一起进入飞行员的麦克风，系统的结果却是送到控制塔的清楚的话音信号。(参见[WiSt85]中对类似的噪声消除系统的讨论。)

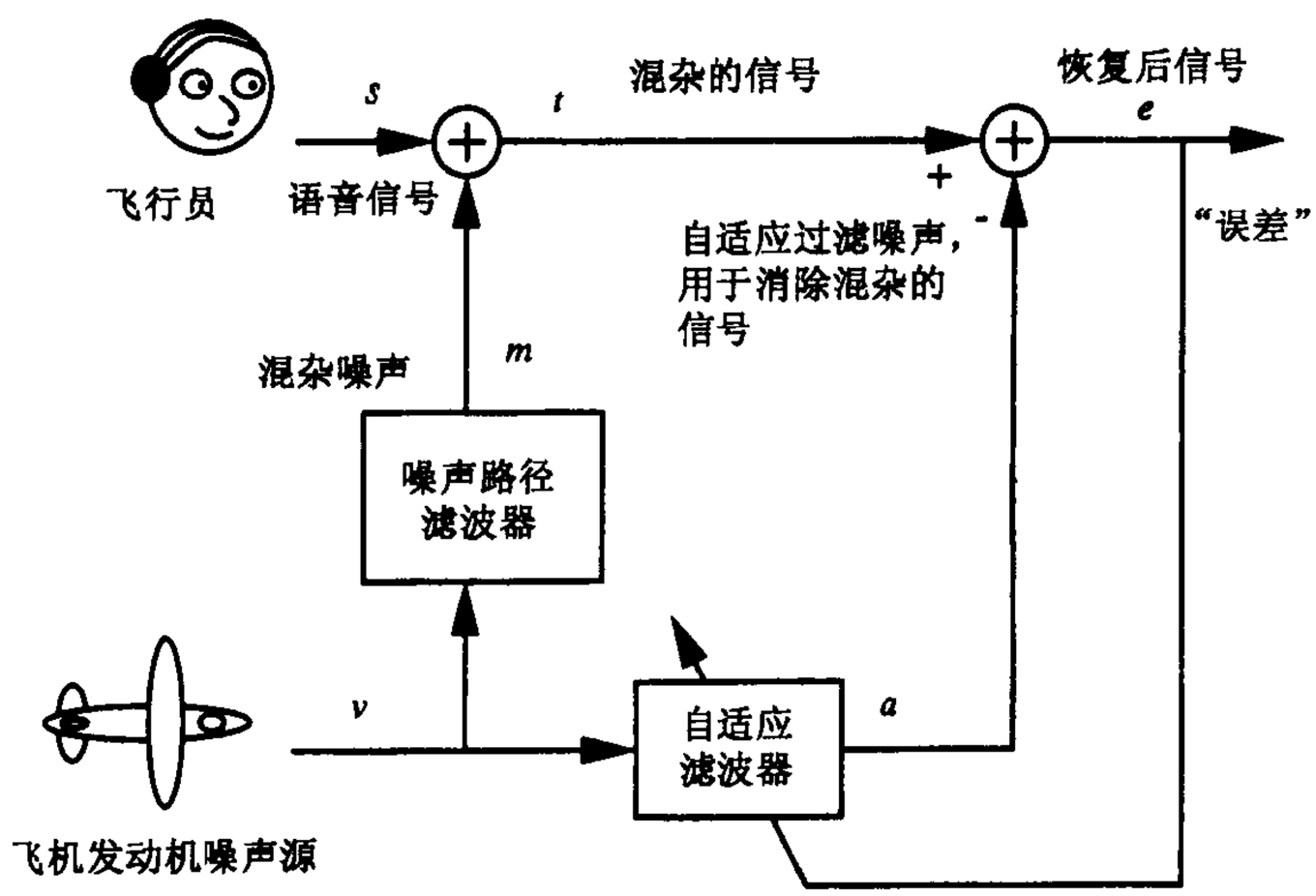



图 10-20 从飞行员的话音信号中过滤发动机噪声

**P10.8** 这是与例题 P4.3 和 P4.5 类似的一个分类问题，但是要使用一个 ADALINE 网络和 LMS 学习规则而不是感知机学习规则。首先描述问题。

10-34 这个分类问题中有 4 类输入向量，分别为：

$$\text{类 1}:\left\{\mathbf{p}_1=\begin{bmatrix}1\\1\end{bmatrix},\mathbf{p}_2=\begin{bmatrix}1\\2\end{bmatrix}\right\}\qquad \text{类 2}:\left\{\mathbf{p}_3=\begin{bmatrix}2\\-1\end{bmatrix},\mathbf{p}_4=\begin{bmatrix}2\\0\end{bmatrix}\right\}$$

$$\text{类 3}:\left\{\mathbf{p}_5=\begin{bmatrix}-1\\2\end{bmatrix},\mathbf{p}_6=\begin{bmatrix}-2\\1\end{bmatrix}\right\}\qquad \text{类 4}:\left\{\mathbf{p}_7=\begin{bmatrix}-1\\-1\end{bmatrix},\mathbf{p}_8=\begin{bmatrix}-2\\-2\end{bmatrix}\right\}$$

使用 LMS 学习规则训练一个 ADALINE 网络，以解决此问题。假定每种模式发生的概率均为 1/8。

**解**

首先画出输入向量，如图 10-21 空心圆表示类 1 的向量，空心方块表示类 2 的向量，实心圆表示类 3 的向量，实心方块表示类 4 的向量。

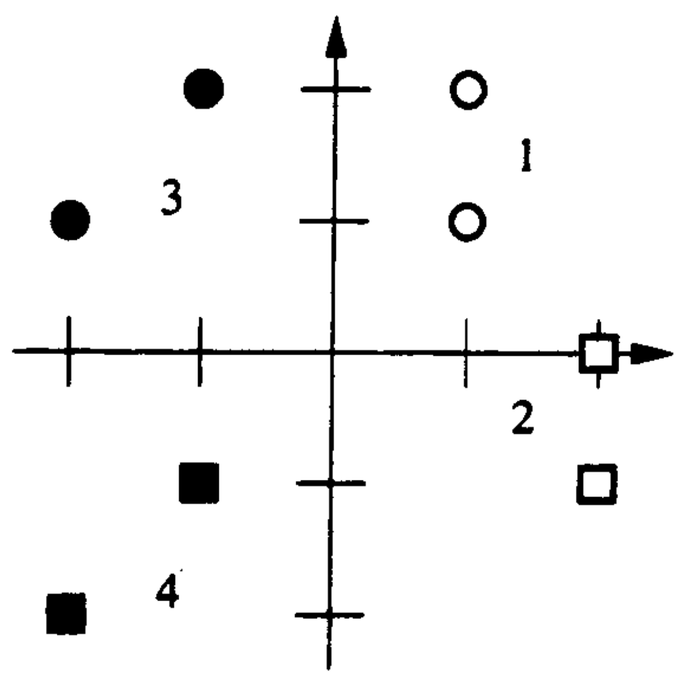



图 10-21 例题 P10.8 中的输入向量

我们将使用与例题 P4.3 中所使用的相类似的目标向量，但是用目标输出 −1 代替目标输出 0。(感知机只能输出 0 和 1)因而，训练集合为

$$\left\{\mathbf{p}_1=\begin{bmatrix}1\\1\end{bmatrix},\mathbf{t}_1=\begin{bmatrix}-1\\-1\end{bmatrix}\right\}\quad\left\{\mathbf{p}_2=\begin{bmatrix}1\\2\end{bmatrix},\mathbf{t}_2=\begin{bmatrix}-1\\-1\end{bmatrix}\right\}$$

$$\left\{\mathbf{p}_3=\begin{bmatrix}2\\-1\end{bmatrix},\mathbf{t}_3=\begin{bmatrix}-1\\1\end{bmatrix}\right\}\quad\left\{\mathbf{p}_4=\begin{bmatrix}2\\0\end{bmatrix},\mathbf{t}_4=\begin{bmatrix}-1\\1\end{bmatrix}\right\}$$

$$\left\{\mathbf{p}_5=\begin{bmatrix}-1\\2\end{bmatrix},\mathbf{t}_5=\begin{bmatrix}1\\-1\end{bmatrix}\right\}\quad\left\{\mathbf{p}_6=\begin{bmatrix}-2\\1\end{bmatrix},\mathbf{t}_6=\begin{bmatrix}1\\-1\end{bmatrix}\right\}$$

$$\left\{\mathbf{p}_7=\begin{bmatrix}-1\\-1\end{bmatrix},\mathbf{t}_7=\begin{bmatrix}1\\1\end{bmatrix}\right\}\quad\left\{\mathbf{p}_8=\begin{bmatrix}-2\\-2\end{bmatrix},\mathbf{t}_8=\begin{bmatrix}1\\1\end{bmatrix}\right\}$$

10-35 与例题 P4.5 中一样，我们也从下面的初始权值和偏置值开始：

$$\mathbf{W}(0)=\begin{bmatrix}1&0\\0&1\end{bmatrix},\quad \mathbf{b}(0)=\begin{bmatrix}1\\1\end{bmatrix}$$

我们已差不多准备好用 LMS 规则训练一个 ADALINE 网络了。学习速度设为 $\alpha=0.04$，根据下标的顺序依次输入各个输入向量。第一次迭代为

$$\mathbf{a}(0) = purelin(\mathbf{W}(0)\mathbf{p}(0) + \mathbf{b}(0)) = purelin\left(\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} + \begin{bmatrix}1\\1\end{bmatrix}\right) = \begin{bmatrix}2\\2\end{bmatrix}$$

$$\mathbf{e}(0) = \mathbf{t}(0) - \mathbf{a}(0) = \begin{bmatrix}-1\\-1\end{bmatrix} - \begin{bmatrix}2\\2\end{bmatrix} = \begin{bmatrix}-3\\-3\end{bmatrix}$$

$$\begin{aligned}\mathbf{W}(1) &= \mathbf{W}(0) + 2\alpha\mathbf{e}(0)\mathbf{p}^T(0)\\ &= \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} + 2(0.04)\begin{bmatrix}-3\\-3\end{bmatrix}\begin{bmatrix}1 & 1\end{bmatrix} = \begin{bmatrix}0.76 & -0.24\\-0.24 & 0.76\end{bmatrix}\end{aligned}$$

$$\mathbf{b}(1) = \mathbf{b}(0) + 2\alpha\mathbf{e}(0) = \begin{bmatrix}1\\1\end{bmatrix} + 2(0.04)\begin{bmatrix}-3\\-3\end{bmatrix} = \begin{bmatrix}0.76\\0.76\end{bmatrix}$$

第二次迭代为

$$\begin{aligned}\mathbf{a}(1) &= purelin(\mathbf{W}(1)\mathbf{p}(1) + \mathbf{b}(1))\\ &= purelin\left(\begin{bmatrix}0.76 & -0.24\\-0.24 & 0.76\end{bmatrix}\begin{bmatrix}1\\2\end{bmatrix} + \begin{bmatrix}0.76\\0.76\end{bmatrix}\right) = \begin{bmatrix}1.04\\2.04\end{bmatrix}\end{aligned}$$

$$\mathbf{e}(1) = \mathbf{t}(1) - \mathbf{a}(1) = \begin{bmatrix}-1\\-1\end{bmatrix} - \begin{bmatrix}1.04\\2.04\end{bmatrix} = \begin{bmatrix}-2.04\\-3.04\end{bmatrix}$$

$$\begin{aligned}\mathbf{W}(2) &= \mathbf{W}(1) + 2\alpha\mathbf{e}(1)\mathbf{p}^T(1)\\ &= \begin{bmatrix}0.76 & -0.24\\-0.24 & 0.76\end{bmatrix} + 2(0.04)\begin{bmatrix}-2.04\\-3.04\end{bmatrix}\begin{bmatrix}1 & 2\end{bmatrix} = \begin{bmatrix}0.5968 & -0.5664\\-0.4832 & 0.2736\end{bmatrix}\end{aligned} \quad \boxed{10\text{-}36}$$

$$\mathbf{b}(2) = \mathbf{b}(1) + 2\alpha\mathbf{e}(1) = \begin{bmatrix}0.76\\0.76\end{bmatrix} + 2(0.04)\begin{bmatrix}-2.04\\-3.04\end{bmatrix} = \begin{bmatrix}0.5968\\0.5168\end{bmatrix}$$

若继续下去直到权值收敛，则可以得到

$$\mathbf{W}(\infty) = \begin{bmatrix}-0.5948 & -0.0523\\0.1667 & -0.6667\end{bmatrix}, \quad \mathbf{b}(\infty) = \begin{bmatrix}0.0131\\0.1667\end{bmatrix}$$

得到的判定边界如图 10-22 所示。将此结果与例题 P4.5 中由感知机学习规则得到的最终判定边界(图 4-25)相比较。当所有的模式被正确分类时，感知机规则便停止了训练。LMS 算法使判定边界尽可能远离被分类的模式。

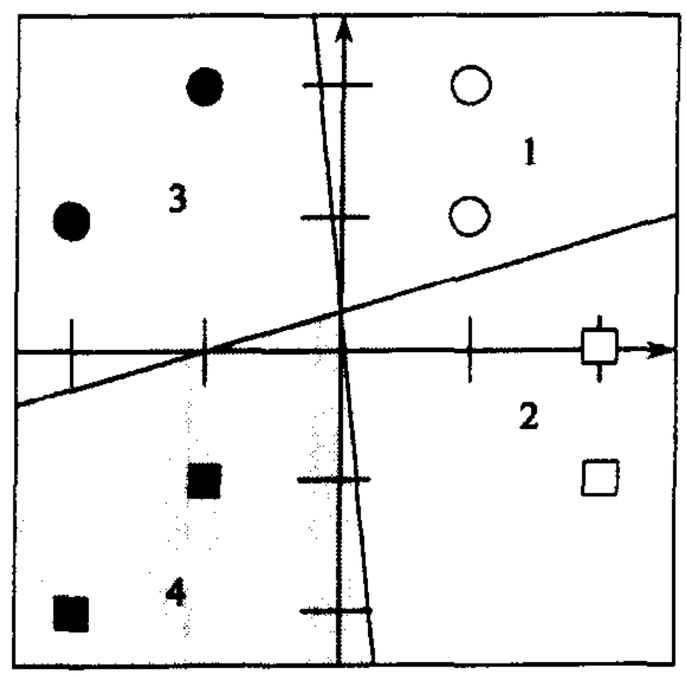
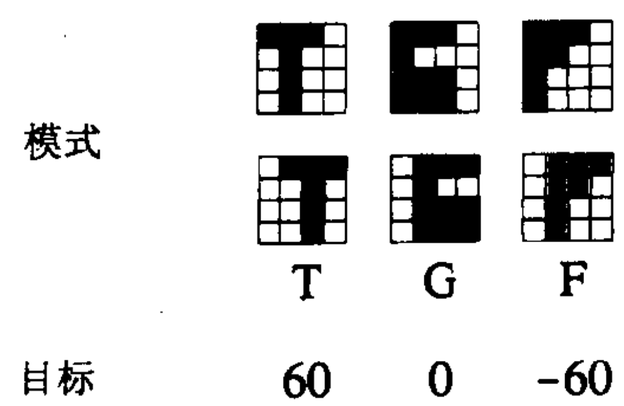



图 10-22 例题 P10.8 的最终判定边界

**P10.9** 重做在 Widrow 和 Hoff 在 1960 年的经典论文中的模式识别问题[WiHo60]。他们想设计一个能将图 10-23 中的 6 个模式分类的识别系统。




10-37

图 10-23 模式和它们的分类目标

这些模式表示字母 T，G 和 F，上面一排是它们的原始形式，下面一排则是将它们移动后的形式。这些字母的分类目标分别为 60，0 和 - 60。(Widrow 和 Hoff 使用 60，0 和 - 60 的原因是为了较好地在他们使用的仪器表面显示他们的网络输出结果。)目标是训练网络，使得它将 6 个模式划分到相应的下 T，G 和 F 组中。

**解**

模式图中对黑色的方块赋值 + 1，白色的方块赋值 - 1。首先我们将每个字母转换为一个 16 元素的向量。转换时从左上角开始，先转换左边第 1 列，接着转换第 2 列，等等。例如，对应于未移动的字母 T，其相应的向量为：

$$\mathbf{p}_1 = [1 \quad -1 \quad -1 \quad -1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1]^T$$

对 6 个字母中的每一个将产生这样的一个输入向量。

将使用的 ADALINE 网络如图 10-24。

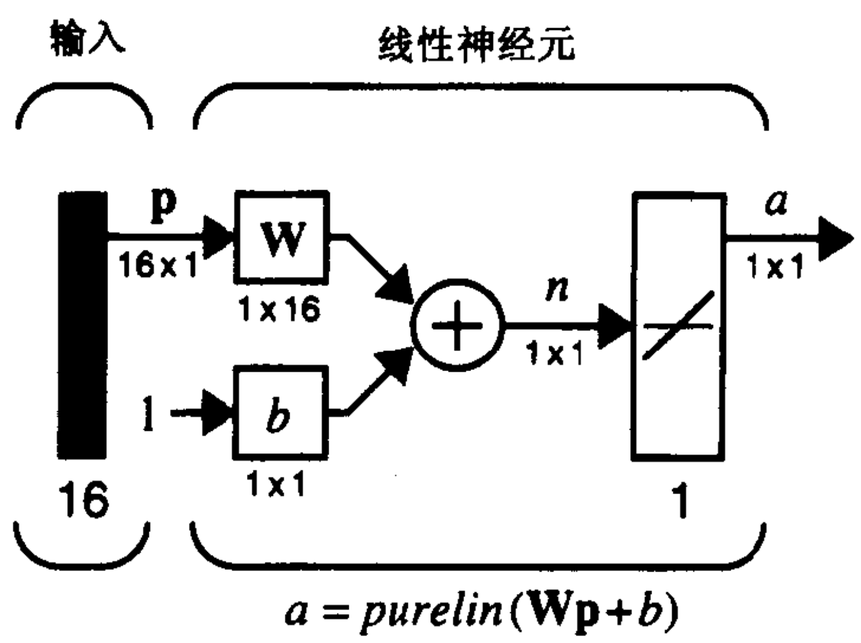



图 10-24 自适应模式分类器

(Widrow 和 Hoff 构造了实现这个 ADALINE 的机器。据他们所述，它“像一个午餐桶那么大”。)

现在将 6 个输入向量以随机的顺序输入到网络中，在每一次输入后用 LMS 算法调整网络的权值，学习速度为 $\alpha = 0.03$。调整权值后，再将 6 个输入向量到输入到网络中，并产生它们的输出结果和相应的误差。用误差的平方和来检测网络的质量。

图 10-25 说明了网络的收敛情况。总共输入约 60 个输入向量，即每个可能的输入向量大约输入 10 次。

10-38

图 10-25 中的结果与 Widrow 和 Hoff 在 35 年前得到的和发表的结果很相似。Widrow 和 Hoff 做了很好的科学工作，甚至几十所后他们的工作都是可重复的(但不必要有一个午餐桶了)。

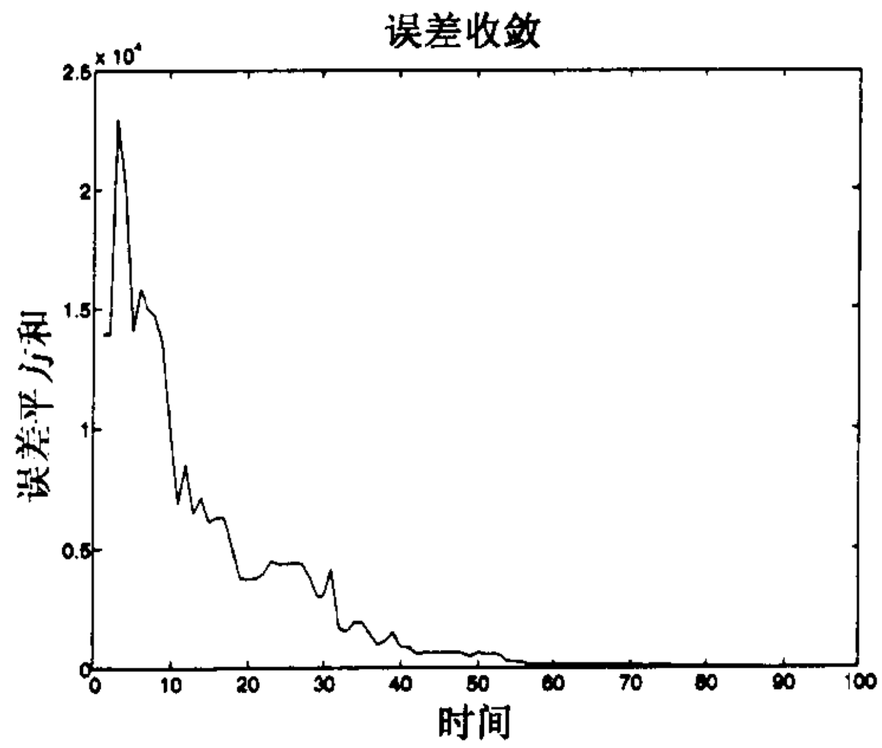



图 10-25 学习速度为 0.03 时的误差收敛曲线

试验这个字母识别问题可以使用 *Neural Network Design Demonstration Linear Pattern Classification*(**nnd101c**)。注意网络对输入模式中噪声的敏感性。 10-39

## 10.5 结束语

在本章中，我们介绍了 ADALINE 神经网络和 LMS 学习规则。ADALINE 网络与第 4 章中的感知器网络很相似，两者具有相同的基本限制：它们只能对线性可分的模或进行分类。尽管对于这个限制，事实上 LMS 算法仍比感知机学习规则更有效。因为它使均方误差最小化，所以算法能产生比感知机学习规则受噪声影响小的判定边界。

ADALINE 网络和 LMS 算法在实践中均有许多应用。尽管它们是在 20 世纪 50 年代末首次被提出来的，它们仍广泛地用于自适应滤波的应用中。当前，在许多长途电话线上安装的回声消除系统就使用了 LMS 算法。

除了作为许多自适应滤波应用的实际解决办法以外，LMS 算法也因为它是反向传播(BP)算法的前驱而显得很重要，BP 算法将在第 11 章和 12 章中讨论。像 LMS 算法一样，反向传播算法也是使均方误差最小化的近似的最速下降算法。两个算法惟一的区别在于导数的计算方式。BP 算法是 LMS 算法的推广，可以用于多层神经网络。这些更复杂的网络不限于解线性可分问题。它们能解决任意的分类问题。 10-40

## 参考文献

[AnRo89] J. A. Anderson, E. Rosenfeld, *Neurocomputing: Foundations of Research*, Cambridge, MA: MIT Press, 1989.

*Neurocomputing*(《神经计算》)是一本基础参考书。它包含了 40 多篇神经计算方面最重要的著述。每篇文章附有一段引言，是对文章结果的综述和在该领域中的历史地位的评价。

[StDo84] W. D. Stanley, G. R. Dougherty, R. Dougherty, *Digital Signal Processing*, Reston VA: Reston, 1984.

[WiHo60] B. Widrow, M. E. Hoff, "Adaptive switching circuits", 1960 *IRE WESCON Convention Record*, NewYork: IRE Part 4, pp. 96 - 104.

这篇重要文章描述一个自适应的类感知机的网络，它能快速准确地学习。作者假定系统有输入，每个输入有一个期望的输出类别，且系统能计算实际输出和期望输出之间的误差。为了使均方误差最小化，网络使用一个最速下降法来调整权值。(最小均方误差或 LMS 算法)。这篇文章在[AnRo89]中被重印。

[WiSt85] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Englewood Cliffs, NJ: Prentice - Hall, 1985.

这本内容丰富型的书叙述了自适应信号处理方面的理论和应用。作者在书中概述了所需要的数学背景知识，给出了他们的详细的自适应算法，并计论了许多实际应用。

[WiWi88] B. Widrow and R. Winter, "Neural nets for adaptive filtering and adaptive pattern recognition, "*IEEE Computer Magazine*, March 1988, pp. 25 - 39.

这篇文章特别可读，它总结了自适应多层神经网络的应用。网络被用于系统建模、统计预测、回声消除、反向建模和模式识别等。

10-41

## 习题

**E10.1** 图 10-26 中所示为一个自适应滤波器 ADALINE。假设网络的权值为：

$$w_{1,1} = 1, w_{1,2} = -4, w_{1,3} = 2$$

滤波器的输入为

$$\{y(k)\} = \{\cdots,0,0,0,1,1,2,0,0,\cdots\}$$

求滤波器的响应 $\{a(k)\}$。

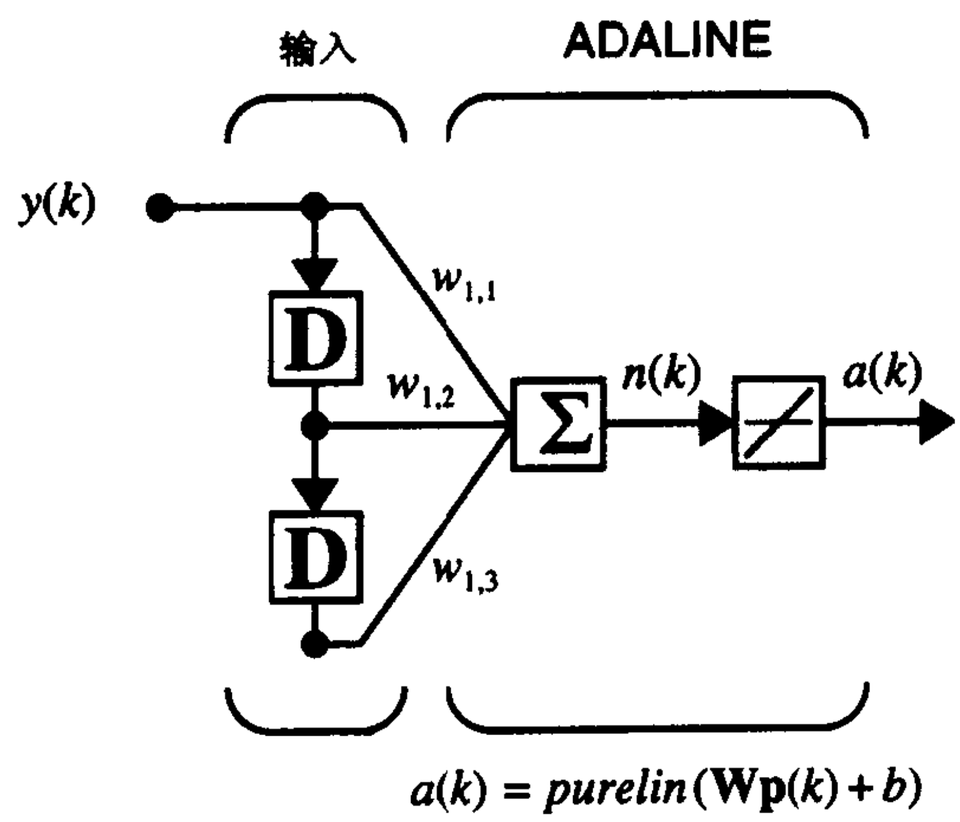



图 10-26 习题 E10.1 的自适应滤波器 ADALINE

**E10.2** 图 10-27 中给出了两类模式。

(i) 用 LMS 算法训练一个 ADALINE 网络，使之能区分类 I 和类 II 中的模式

（即要求网络能区水平线和垂直线）。

（ii）你能解释为什么 ADALINE 可能难于解决此问题吗？

图 10-27 习题 E10.2 的模式分类问题

10-42

**E10.3** 假定有下面的参模式和它们的目标输出：

$$\left\{\mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_1 = 1\right\}, \left\{\mathbf{p}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, t_2 = -1\right\}$$

在例题 P10.3 中假定输入到 ADALINE 的这些向量以等概率产生。现在假定向量 $\mathbf{p}_1$ 产生的概率为 0.75，向量 $\mathbf{p}_2$ 产生的概率 0.25。概率的改变是否会改变均方误差的曲面？若是，现在曲面的形状如何？最大稳态学习速度是多少？

**E10.4** 本习题中，例题 P10.3 中的参考模式 $\mathbf{p}_2$ 被修改为

$$\left\{\mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_1 = 1\right\}, \left\{\mathbf{p}_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, t_2 = -1\right\}$$

（i）假定两种模式以等概率产生。求均方误差并描绘出其轮廓线图。

（ii）求最大稳态学习速度。

（iii）写一个用 LMS 算法求解此问题的 MATLAB M-file。对一个稳态学习速度让算法执行 40 步。用零向量作为初始值。画出轮廓线图上的变化轨迹。

（iv）在将两个参数的初始值均设为 1 后，让算法执行 40 步。画出最终的判定边界。

（v）比较(iii)和(iv)的最终参数。解释比较的结果。

**E10.5** 再次使用例题 P10.3 中的参考模式和目标输出，假定模式以等概率产生。这里我们要训练一个有偏置值 ADALINE 网络。求三个参数：$w_{1,1}$，$w_{1,2}$和 $b$。

（i）求均方误差和最大稳态学习速度。

（ii）写一个用 LMS 算法求解此问题的 MATLAB M-file 文件。对一个稳态学习速度让算法执行 40 步。用零向量作为初始值。画出最终的判定边界。

（iii）将所有参数的初始值均设为 1，让算法执行 40 步。画出最终的判定边界。

（iv）比较(iii)和(iv)中得到的最终参数值和判定边界。解释比较的结果。 10-43

**E10.6** 考虑图 10-28 中的自适应预测器。

假定 $y(k)$是一个稳态过程，其自相关函数为：

$$C_y(n) = E[y(k)(y(k+n))]$$

（i）写出包含 $C_y(n)$项的均方误差的表达式。

（ii）当 $y(k) = \sin\left(\frac{k\pi}{5}\right)$时，写出均方误差的表达式。

（iii）求均方误差的赫森矩阵的特征值和特征向量。确定极小点并画出大致的轮廓线图。

（iv）求 LMS 算法的最大稳态学习速度。

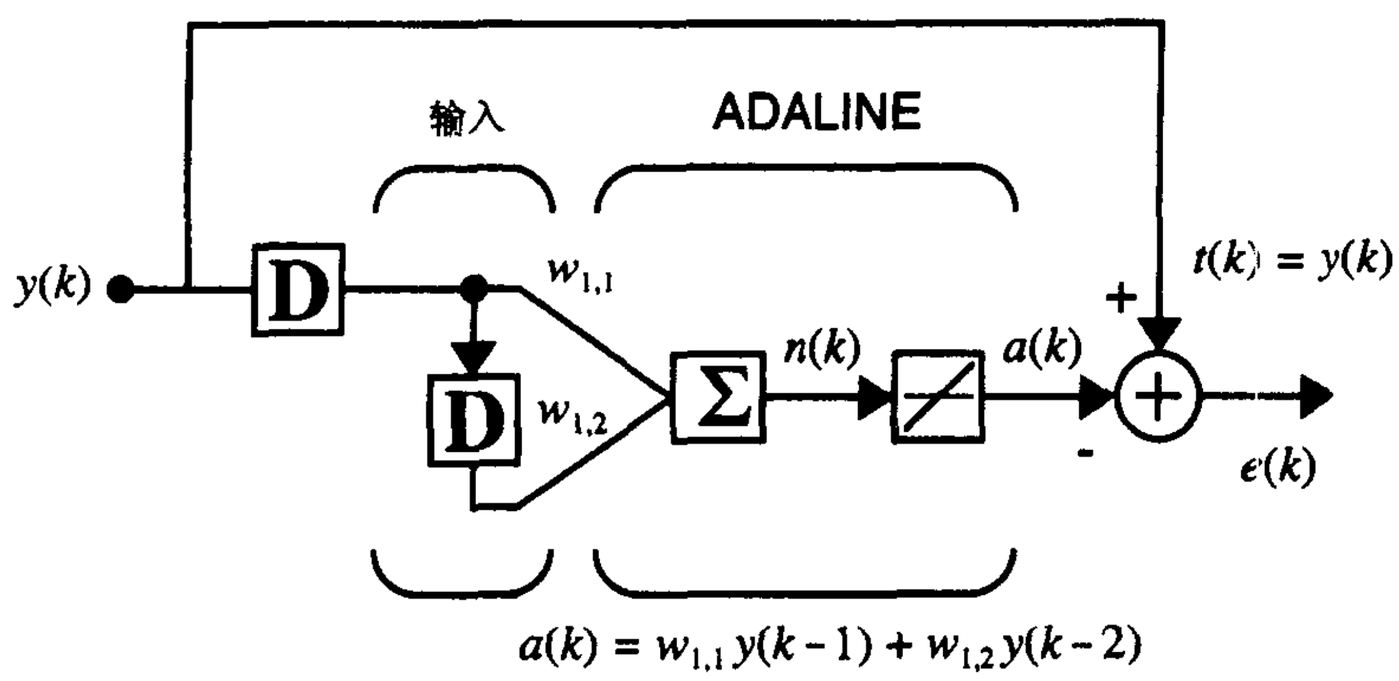



图 10-28 习题 E10.6 的自适应预测器

(v) 用一个稳学习速度，手工计算 LMS 算法的三步执行结果。用零向量作为初始值。

(vi) 写一个用 LMS 算法求解此问题的 MATLAB M-file 文件。对一个稳态学习速度，让算法执行 40 步。画出轮廓线图上权值的轨迹。用零向量作为初始值。验证算法收敛于最优点。

(vii) 以试验方式验证当学习速度大于(iv)中求得的学习速度时，算法不稳定。

**E10.7** 再次解例题 P10.9，不过用数字“1”，“2”和“4”代替字母“T”，“G”和“F”。对每个参考模式和噪声模式，测试经过训练后的网络。讨论网络的敏感性。(使用 *Neural Network Design Demonstration* *Linear Pattern Classification* (**nnd10lc**)。)

10-44

# 第11章 反向传播

## 11.1 目的

在第8章中我们开始性能学习的讨论，在第10章中给出了LMS算法，本章中继续对性能学习的讨论，并给出一个更一般的LMS算法，称为反向传播法，它可用来训练多层网络。根据LMS学习法则，反向传播法也是最速下降算法的近似，其中性能指数是均方误差。LMS算法和反向传播法的区别在于它们对导数的计算方式上。对单层的线性网络，误差是网络权值的显式线性函数，其相对于权值的导数较为容易求得。在具有非线性传输函数的多层网络中，网络权值和误差的关系更为复杂。为了计算导数，需要使用微积分的链法则。事实上，本章的一大部分是在讲述如何使用链法则上。 11-1

## 11.2 理论和实例

Frank Rosenblatt 的感知机学习规则和 Bernard Widrow 和 Marcian Hoff 的 LMS 算法是设计用来训练单层的类似感知器的网络的。如前面几章所述，这些单层网络的缺点是只能解线性可分的分类问题。Rosenblatt 和 Widrow 均意识到这些限制并且都提出了克服此类问题的方法：多层网络。但他们未将这类算法推广到用来训练功能更强的网络。

Paul Werboss 在他 1974 年的论文中第一次描述了训练多层神经网络的一个算法[Werbo74]，论文中的算法是在一般网络的情况中描述的，而将神经网络作为一个特例。论文没有在神经网络研究圈子内传播。直到20世纪80年代中期，反向传播算法才重新被发现并广泛地宣扬，它是被 David Rumelhart，Geoffrey Hinton 和 Ronald Williams [RuHi86]，David Parker[Park85]，以及 Yann Le Cun[LeCu85]分别独立地重新发现的。这个算法因被包括在《并行分布式处理》(*Parallel Distributed Processing*) [RuMc86]一书中而得到普及。这本书介绍了心理学家 David Rumelhart 和 James McClelland 领导的并行分布处理小组所做的研究工作。这本书的出版引发了神经网络的研究热潮。当前，用反向传播算法训练的多层感知机是应用最广的神经网络。

本章中，首先让我们来看看多层网络的能力，然后叙述反向传播算法。

### 11.2.1 多层感知机

首先我们介绍第2章中所用的多层网络的符号。为便于参考，我们在图 11-1 中重新画出一个三层感知机的图。注意三个感知机网络只是简单地被连接在一起。第一个网络的输出是第二个网络的输入，第二个网络的输出是第三个网络输入。每一层可以有不同数目的神经元，甚至传输函数也可以不同。在第2章中我们用上标来表示层号。因而，第一层的权值矩阵写作 $\mathbf{W}^1$，第二层的权值矩阵写作 $\mathbf{W}^2$。

为了表示多层网络的结构，有时我们使用下面的速记符号，其中在输入的数目后面跟着每一层的神经元数目：

11-2

$$R - S^1 - S^2 - S^3 \tag{11.1}$$

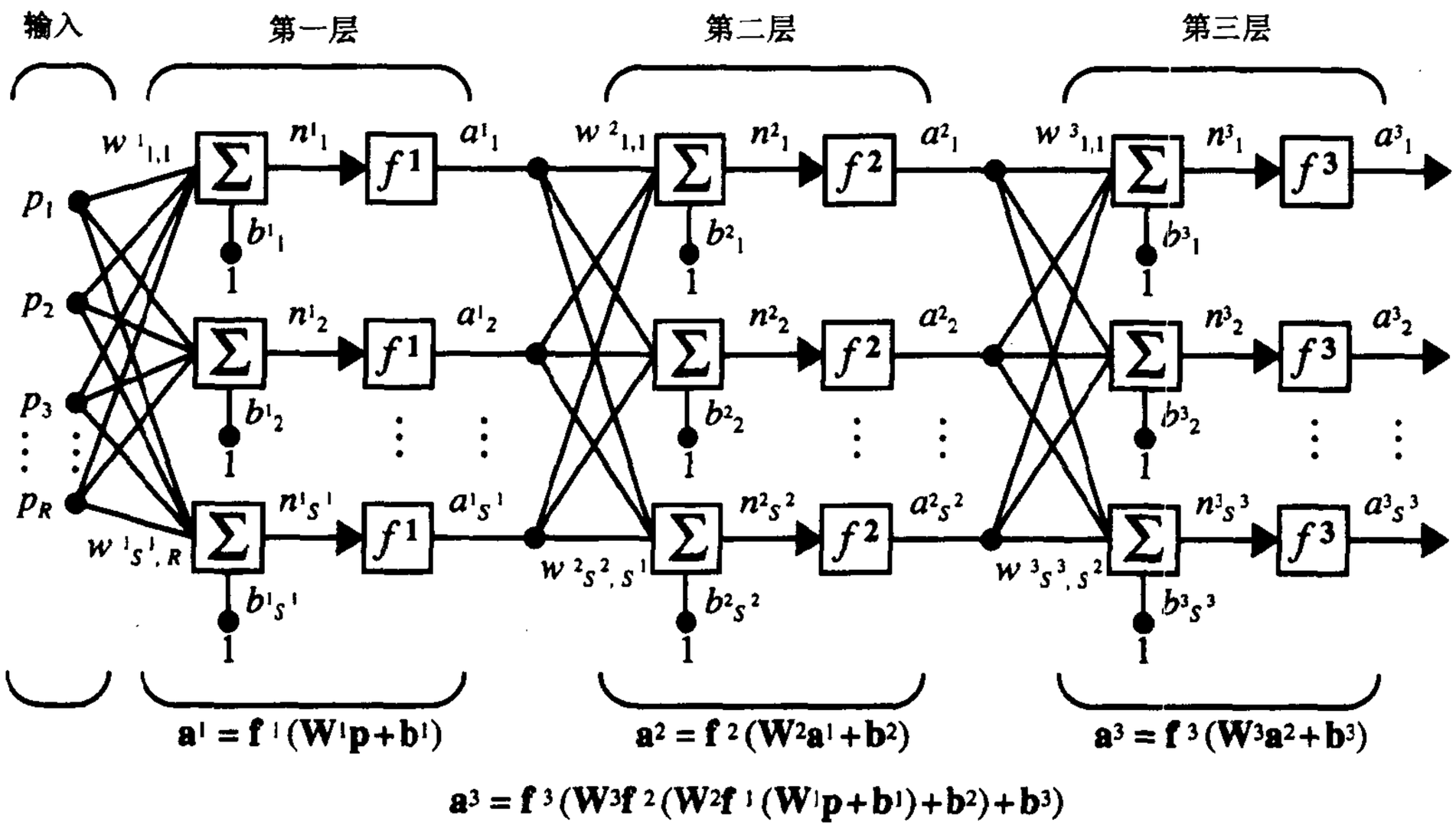



图 11-1 三层网络

现来看这些多层感知器网络的能力，首先看多层网络在模式分类中的使用，然后讨论在函数逼近中的应用。

1. 模式分类

要说明多层感知机用于模式分类的能力，考虑经典的异或(XOR)问题。异或的输入/目标输出对为

$$\left\{\mathbf{p}_1 = \begin{bmatrix}0\\0\end{bmatrix}, t_1 = 0\right\}\left\{\mathbf{p}_2 = \begin{bmatrix}0\\1\end{bmatrix}, t_2 = 1\right\}\left\{\mathbf{p}_3 = \begin{bmatrix}1\\0\end{bmatrix}, t_3 = 1\right\}\left\{\mathbf{p}_4 = \begin{bmatrix}1\\1\end{bmatrix}, t_4 = 0\right\}$$

1969 年 Minsky 和 Papert 曾用此问题来说明单层感知机的局限性，如图 11-2 所示，因为两个类别不是线性可分的，所以一个单层的感知机不能完成分类任务。

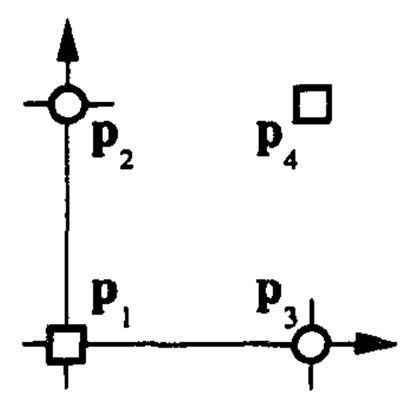



图 11-2

然而一个两层的网络能解决异或问题。事实上，有许多种多层网络可解决此问题。一种办法是在第一层中用两个神经元来产生两个判定边界。第一个边界将 $\mathbf{p}_1$ 和其他模式分开，第二个边界则将 $\mathbf{p}_4$ 分开。然后第二层网络用一个 AND 操作将两个边界结合在一起。对第一层的每个神经元，其判定边界如图 11-3 所示。

11-3

结果产生的两层 2－2－1 网络如图 11-4。这个网络整个的判定边界如图 11-5，阴影区域表示产生网络输出为 1 的那些输入。

多层网络在模式分类上的应用可见例题 P11.1 和 P11.2。

2. 函数逼近

直到现在为止，在本书中看到的神经网络的应用主要是在模式分类方面。神经网络在本质上也可被看作是函数逼近器。例如，在控制系统中，目标是要找到一个合适的反馈函数，

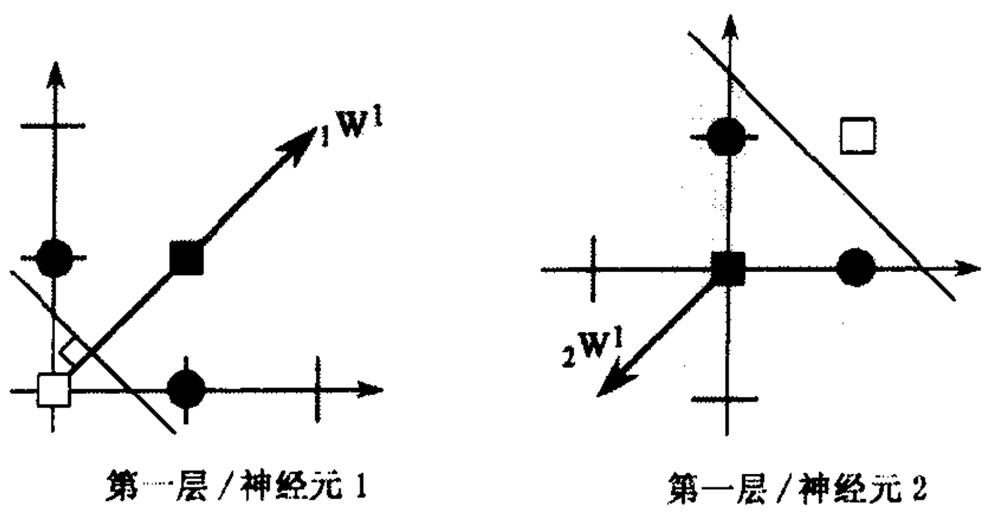



图 11-3 异或(XOR)网络的判定边界

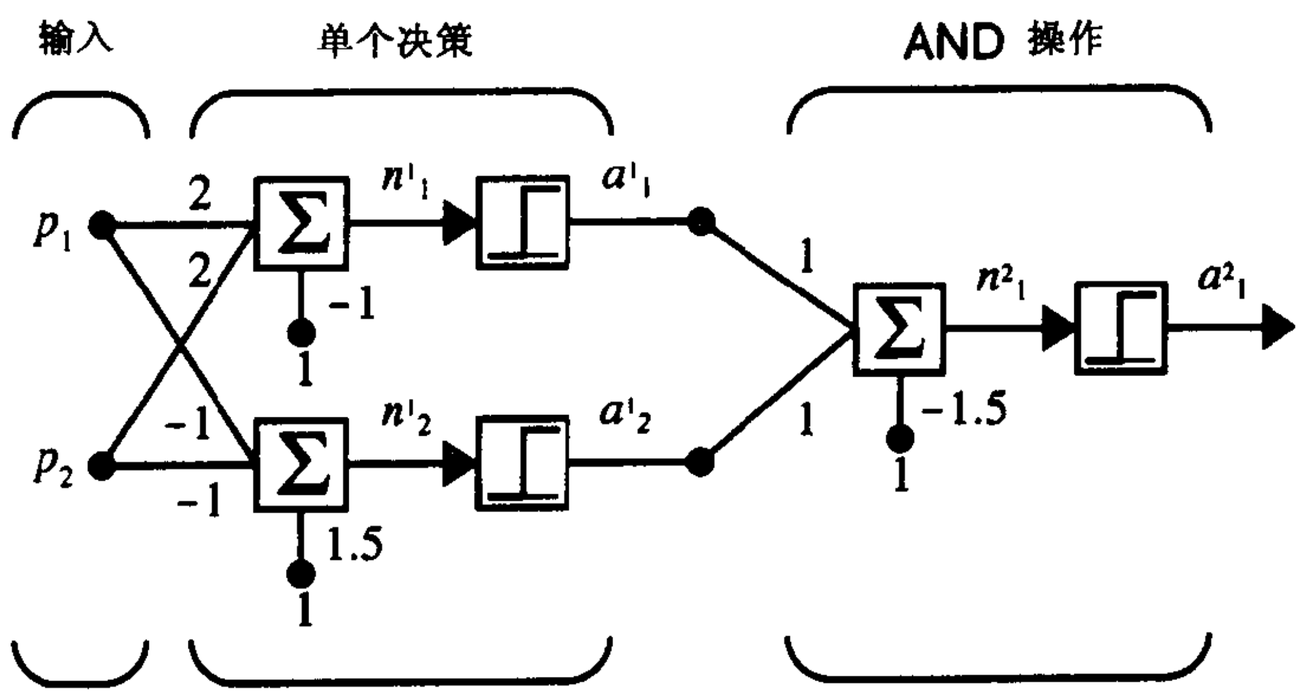



图 11-4 两层异或(XOR)网络

它能将测量到的输出映射为控制输入。在自适应滤波(第 10 章)中，目标是要找到一个函数，使得延迟的输入信号值被映射到相应的输出信号上。下面的例子将展示多层感知机在实现函数方面的灵活性。

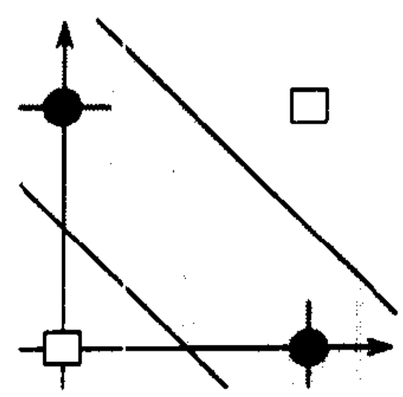


图 11-5

考虑图 11-6 中的两层的 1－2－1 网络。此例中，第一层的传输函数是 log－sigmoid 函数，第二层的是线性函数。换句话说，就是

$$f^1(n) = \frac{1}{1+e^{-n}} \quad 且 \quad f^2(n) = n \tag{11.2}$$

假定这个网络的权值和偏置值为：

$$w^1_{1,1} = 10,\ w^1_{2,1} = 10,\ b^1_1 = -10,\ b^1_2 = 10,\ w^2_{1,1} = 1,\ w^2_{1,2} = 1,\ b^2 = 0$$

网络在这些参数下的响应如图 11-7，图中网络输出 $a^2$ 为输入 $p$ 的函数，且 $p$ 的取值范围为 [－2，2]。

注意网络的响应包括两步，每一步对第一层中的一个对数-S 形神经元的响应。通过调整网络的参数，每一步的曲线形状和位置都可以发生改变，如在下面讨论中将会见到的那样。

每步的曲线中心对应网络第一层中的神经元的净输入为 0：

$$n^1_1 = w^1_{1,1}p + b^1_1 = 0 \quad \Rightarrow \quad p = -\frac{b^1_1}{w^1_{1,1}} = -\frac{-10}{10} = 1 \tag{11.3}$$

$$n_2^1 = w_{2,1}^1 p + b_2^1 = 0 \Rightarrow p = -\frac{b_2^1}{w_{2,1}^1} = -\frac{10}{10} = -1 \tag{11.4}$$

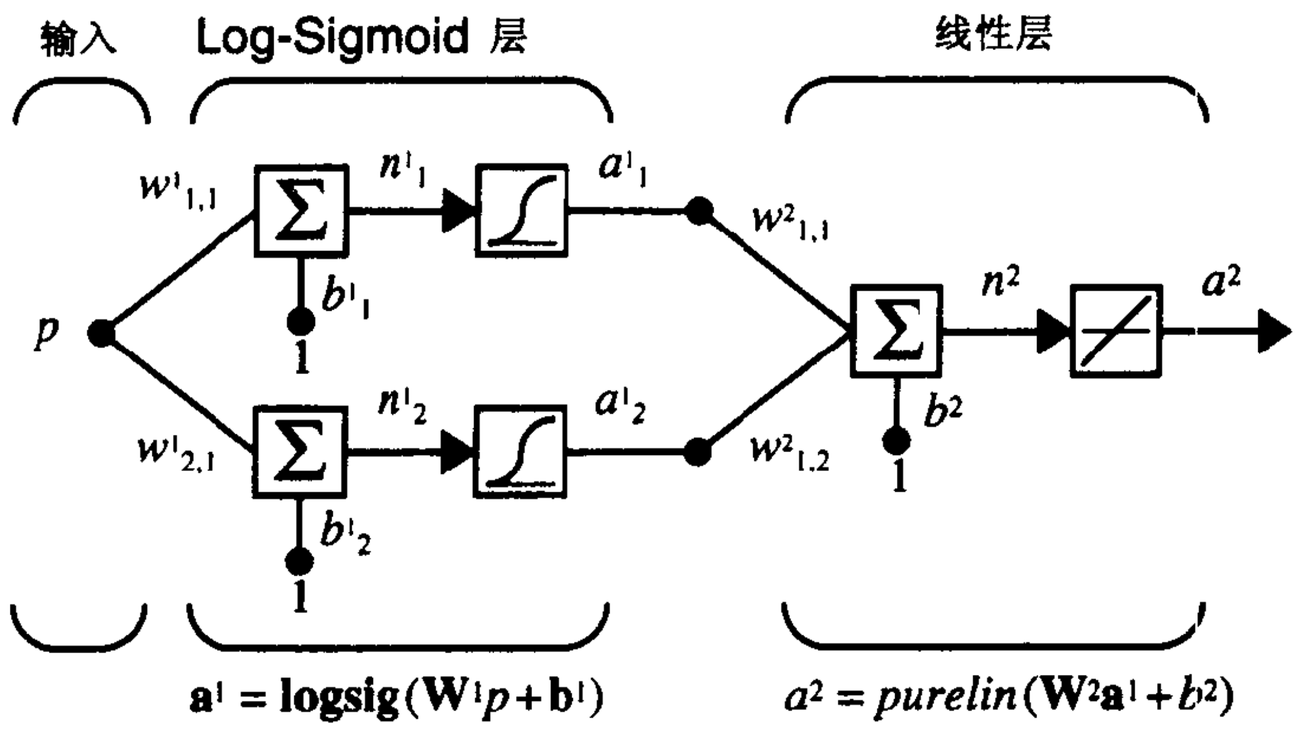



图 11-6 函数逼近网络的例子

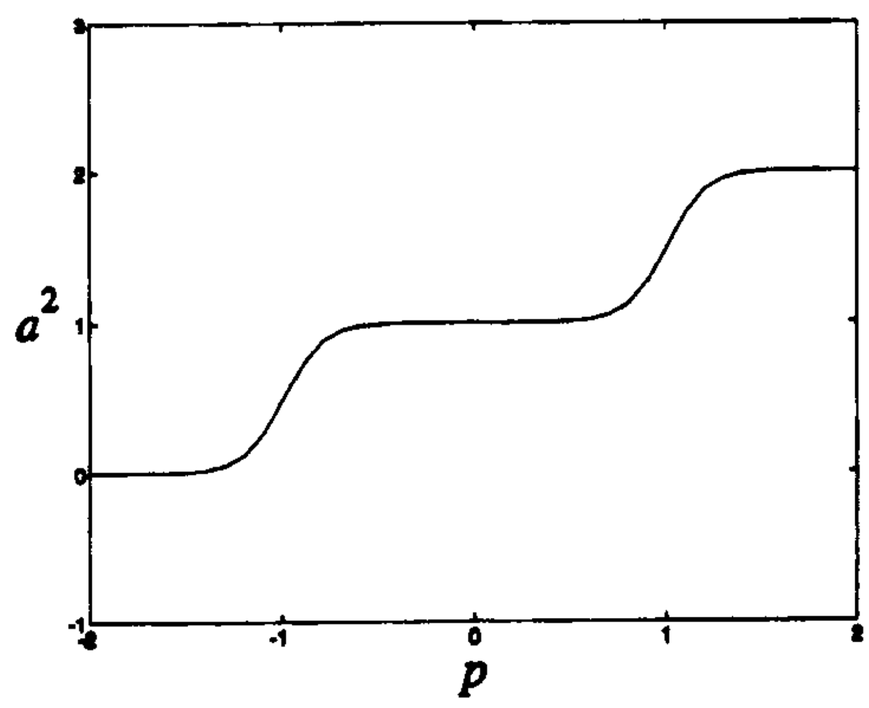



图 11-7 图 11-7 中网络的响应

通过调整网络的权值可以调整每一步曲线的陡度。

图 11-7 说明了参数改变对网络响应的影响。图中的曲线是参数未作调整前的网络响应。其他的曲线对应于当一个参数的取值在下面的范围时的网络响应：

$$-1 \leqslant w_{1,1}^2 \leqslant 1,\ -1 \leqslant w_{1,2}^2 \leqslant 1, 0 \leqslant b_2^1 \leqslant 20,\ -1 \leqslant b^2 \leqslant 1 \tag{11.5}$$

图 11-6(a)说明第一层(隐层)的网络偏置值如何被用来确定每一步曲线的位置。图 11-8(b)说明网络权值如何决定每步曲线的坡度。第二层(输出层)的网络偏置值使整个网络的响应曲线上移或下移，如图 11-8(d)所示。

从这个例子中，可以看到多层网络的灵活性。看起来，只要在隐层中有足够数量的神经元，我们可以用这样的网络来逼近几乎任何一个函数。事实上，研究已表明，两层网络在其隐层中使用 S 形传输函数，在输出层中使用线性传输函数，就几乎可以以任意精度逼近任何感兴趣的函数，只要隐层中有足够的单元可用(见[HoSt89])。

试验这个两层网络的响应曲线请用 *Neural Network Design Demonstration Network Function*(**nnd11nf**)。 11-6

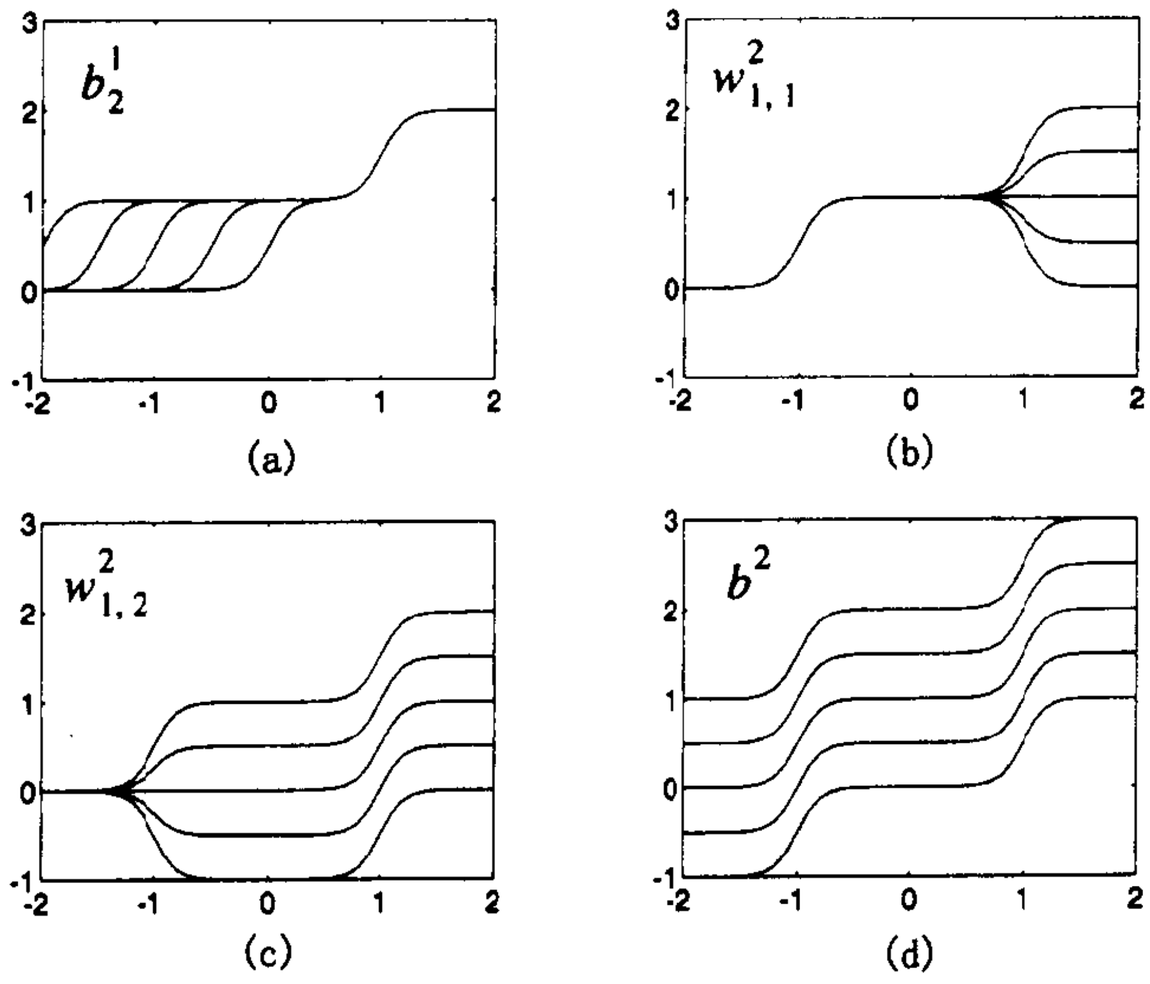



图 11-8 参数改变对网络响应的影响

我们已经有了对多层感知机网络在模式识别和函数逼近中的能力一些概念，下一步是要设计一个算法来训练这样的网络。

### 11.2.2 反向传播算法

使用第 2 章中引入的多层网络的缩写符号可以简化对反向传播算法(BP 算法)的讨论。图 11-9 中是使用缩写符号标记的三层神经网络。

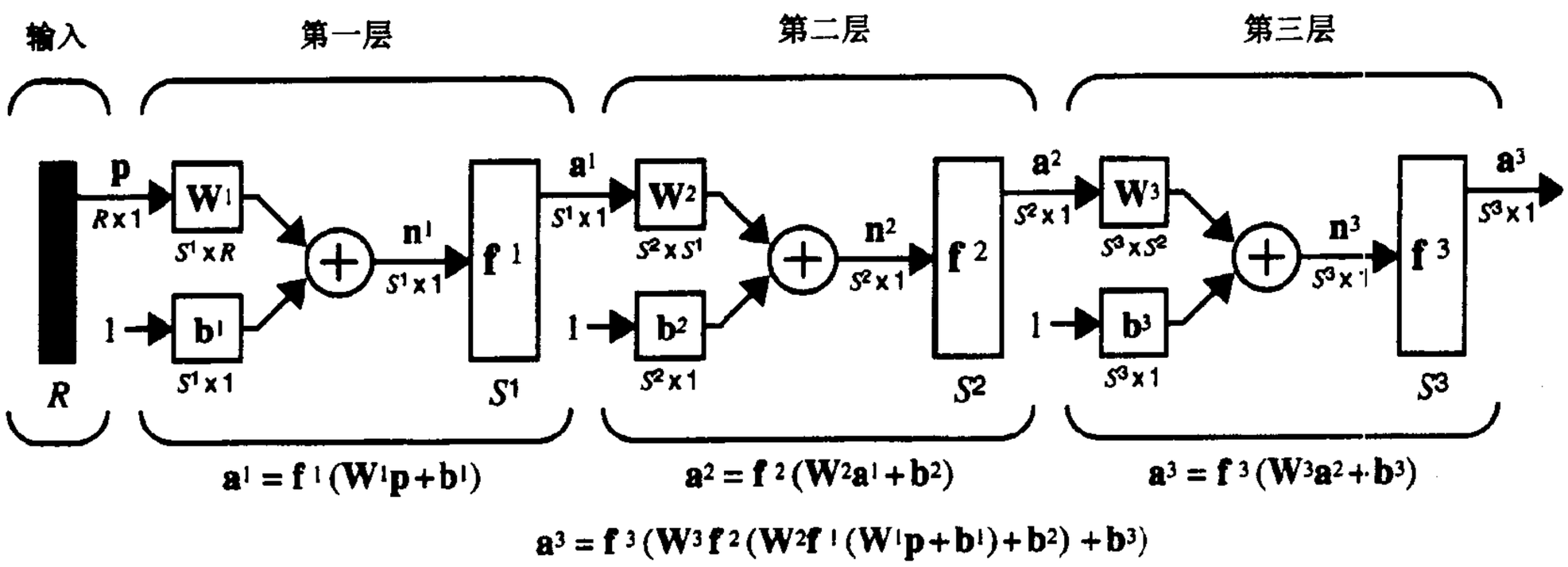



图 11-9 使用缩写符号的三层网络

如前所述，多层网络中某一层的输出成为下一层的输入。描述此操作的等式为：

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^{m} + \mathbf{b}^{m+1}),\ m = 0,1,\cdots,M-1 \tag{11.6}$$

这里，$M$ 是网络的层数。第一层的神经元从外部接收输入：

$$\mathbf{a}^0 = \mathbf{p} \tag{11.7}$$

它是等式(11.6)的起点。最后一层神经元的输出是网络的输出：

11-7

$$\mathbf{a} = \mathbf{a}^M \tag{11.8}$$

1. 性能指数

多层网络的 BP 算法是第 10 章中 LMS 算法的推广。两个算法均使用相同的性能指数：均方误差。算法的输入是一个网络正确行为的样本集合：

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \quad \{\mathbf{p}_2, \mathbf{t}_2\}, \quad \cdots, \quad \{\mathbf{p}_Q, \mathbf{t}_Q\} \tag{11.9}$$

这里 $\mathbf{p}_q$ 是网络的输入，$\mathbf{t}_q$ 是对应的目标输出。每输入一个样本，便将网络输出与目标输出相比较。算法将调整网络参数以使均方误差最小化：

$$F(\mathbf{x}) = E[e^2] = E[(t-a)^2] \tag{11.10}$$

这里，$\mathbf{x}$ 是网络权值和偏置值的向量(如第 10 章所述)。若网络有多个输出，则上式的一般形式为：

$$F(\mathbf{x}) = E[\mathbf{e}^T\mathbf{e}] = E[(\mathbf{t}-\mathbf{a})^T(\mathbf{t}-\mathbf{a})] \tag{11.11}$$

如同 LMS 算法，我们用 $\hat{F}(\mathbf{x})$ 来近似计算均方误差：

$$\hat{F}(\mathbf{x}) = (\mathbf{t}(k)-\mathbf{a}(k))^T(\mathbf{t}(k)-\mathbf{a}(k)) = \mathbf{e}^T(k)\mathbf{e}(k) \tag{11.12}$$

这里，均方误差的期望值被第 $k$ 次迭代时的均方误差所代替。

11-8

近似均方误差的最速下降算法为：

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha\frac{\partial\hat{F}}{\partial w_{i,j}^m} \tag{11.13}$$

$$b_i^m(k+1) = b_i^m(k) - \alpha\frac{\partial\hat{F}}{\partial b_i^m} \tag{11.14}$$

这里 $\alpha$ 是学习速度。

到此为止，整个过程与 LMS 算法是一样的。下面将叙述比较难的部分——偏导数的计算。

2. 链法则

对单层线性网络(ADALINE)，这些偏导数可以用式(10.33)和式(10.34)方便地求得。对多层网络，误差不是隐层中的权值的显式函数，因此这些偏导数并不容易求得。

因为误差是隐层中的权值的隐函数，所以下面将用微积分中的链法则来计算偏导数。假设有一个函数 $f$，它仅是变量 $n$ 的显式函数。现在求 $f$ 关于第三个变量 $w$ 的导数，链法则为：

$$\frac{\mathrm{d}f(n(w))}{\mathrm{d}w} = \frac{\mathrm{d}f(n)}{\mathrm{d}n} \times \frac{\mathrm{d}n(w)}{\mathrm{d}w} \tag{11.15}$$

例如，若

$$f(n) = e^n \text{且} n = 2w, \text{所以} f(n(w)) = e^{2w} \tag{11.16}$$

则

$$\frac{\mathrm{d}f(n(w))}{\mathrm{d}w} = \frac{\mathrm{d}f(n)}{\mathrm{d}n} \times \frac{\mathrm{d}n(w)}{\mathrm{d}w} = (e^n)(2) \tag{11.17}$$

下面用此法则来求式(11.13)和(11.14)中的偏导数：

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial w_{i,j}^m} \tag{11.18}$$

$$\frac{\partial \hat{F}}{\partial b_i^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial b_i^m} \tag{11.19}$$

11-9

每个等式中的第二项均可容易地算出，因为 $m$ 层的网络输入是那一层中的权值和偏置值的显式函数：

$$n_i^m = \sum_{j=1}^{s^{m-1}} w_{i,j}^m a_j^{m-1} + b_i^m \tag{11.20}$$

因此，

$$\frac{\partial n_i^m}{\partial w_{i,j}^m} = a_j^{m-1}, \frac{\partial n_i^m}{\partial b_i^m} = 1 \tag{11.21}$$

若定义

$$s_i^m = \frac{\partial \hat{F}}{\partial n_i^m} \tag{11.22}$$

($\hat{F}$ 对 $m$ 层的输入的第 $i$ 个元素变化的敏感性)，则式(11.18)和(11.19)可简化为

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = s_i^m a_j^{m-1} \tag{11.23}$$

$$\frac{\partial \hat{F}}{\partial b_i^m} = s_i^m \tag{11.24}$$

现在可以将近似最速下降法表示为

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha s_i^m a_j^{m-1} \tag{11.25}$$

$$b_i^m(k+1) = b_i^m(k) - \alpha s_i^m \tag{11.26}$$

用矩阵形式表示，则为

$$\mathbf{W}^m(k+1) = \mathbf{w}^m(k) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T \tag{11.27}$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m \tag{11.28}$$

11-10

这里

$$\mathbf{s}^m \equiv \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \begin{bmatrix} \frac{\partial \hat{F}}{\partial n_1^m} \\ \frac{\partial \hat{F}}{\partial n_2^m} \\ \vdots \\ \frac{\partial \hat{F}}{\partial n_{s^m}^m} \end{bmatrix} \tag{11.29}$$

(注意这个算法与式(10.33)和(10.34)的 LMS 算法之间的紧密关系。)

3. 敏感性的反向传播

现在还需要计算敏感性 $\mathbf{s}^m$，这要求再次使用链法则。正是这个过程给出了反向传播这个词，因为它描述了第 $m$ 层的敏感性通过第 $m+1$ 层的敏感性来计算的递推关系。

推出敏感性的递推关系需要使用下面的雅可比矩阵：

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^{m}} \equiv \begin{bmatrix} \frac{\partial n_1^{m+1}}{\partial n_1^m} & \frac{\partial n_1^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_1^{m+1}}{\partial n_{S^m}^m} \\ \frac{\partial n_2^{m+1}}{\partial n_1^m} & \frac{\partial n_2^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_2^{m+1}}{\partial n_{S^m}^m} \\ \vdots & \vdots & & \vdots \\ \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_1^m} & \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_{S^m}^m} \end{bmatrix} \tag{11.30}$$

11-11 下面求这个矩阵的一个表达式。考虑矩阵的 $i$，$j$ 元素：

$$\frac{\partial n_i^{m+1}}{\partial n_j^m} = \frac{\partial\left(\sum_{l=1}^{S^m} w_{i,l}^{m+1} a_l^m + b_i^{m+1}\right)}{\partial n_j^m} = w_{i,j}^{m+1} \frac{\partial a_j^m}{\partial n_j^m} \tag{11.31}$$

$$= w_{i,j}^{m+1} \frac{\partial f^m(n_j^m)}{\partial n_j^m} = w_{i,j}^{m+1} \dot{f}^m(n_j^m)$$

这里

$$\dot{f}^m(n_j^m) = \frac{\partial f^m(n_j^m)}{\partial n_j^m} \tag{11.32}$$

因而雅可比矩阵可写成

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^{m}} = \mathbf{W}^{m+1}\dot{\mathbf{F}}^m(\mathbf{n}^m) \tag{11.33}$$

这里

$$\dot{\mathbf{F}}^m(\mathbf{n}^m) = \begin{bmatrix} \dot{f}^m(n_1^m) & 0 & \cdots & 0 \\ 0 & \dot{f}^m(n_2^m) & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \dot{f}^m(n_{S^m}^m) \end{bmatrix} \tag{11.34}$$

现在可以使用矩阵形式的链法则写出敏感性的递推关系式：

$$\mathbf{s}^m = \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \left(\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m}\right)^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}} = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}} \tag{11.35}$$

$$= \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \mathbf{s}^{m+1}$$

现在我们可以看到反向传播算法得名的原因了。敏感性从最后一层通过网络被反向传播到第一层：

11-12

$$\mathbf{s}^M \rightarrow \mathbf{s}^{M-1} \rightarrow \cdots \rightarrow \mathbf{s}^2 \rightarrow \mathbf{s}^1 \tag{11.36}$$

这里值得强调的是，BP 算法使用的是在 LMS 算法中用到的相同的近似最速下降法。惟一复杂的是，为了计算梯度，需要首先反向传播敏感性。反向传播的优点是我们可以很有效地实现链法则。

完成 BP 算法前还有一点事情要做。我们需要递推关系式(11.35)的起始点 $\mathbf{s}^M$。这在最后一层得到

$$s_i^M = \frac{\partial \hat{F}}{\partial n_i^M} = \frac{\partial(\mathbf{t}-\mathbf{a})^T(\mathbf{t}-\mathbf{a})}{\partial n_i^M} = \frac{\partial \sum_{j=1}^{S^M}(t_j - a_j)^2}{\partial n_i^M} = -2(t_i - a_i)\frac{\partial a_i}{\partial n_i^M} \tag{11.37}$$

由于

$$\frac{\partial a_i}{\partial n_i^M} = \frac{\partial a_i^M}{\partial n_i^M} = \frac{\partial f^M(n_i^M)}{\partial n_i^M} = \dot{f}^M(n_i^M) \tag{11.38}$$

可以写出

$$s_i^M = -2(t_i - a_i)\dot{f}^M(n_i^M) \tag{11.39}$$

这可以用矩阵形式表示成

$$\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a}) \tag{11.40}$$

4. 小结

下面小结 BP 算法。第一步是通过网络将输入向前传播：

$$\mathbf{a}^0 = \mathbf{p} \tag{11.41}$$

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{w}^{m+1}\mathbf{a}^m + \mathbf{b}^{m+1}),\ m = 0,1,\cdots,M-1 \tag{11.42}$$

$$\mathbf{a} = \mathbf{a}^M \tag{11.43}$$

下一步是通过网络将敏感性反向传播：

$$\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a}) \tag{11.44}$$ 11-13

$$\mathbf{s}^m = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T\mathbf{s}^{m+1},\ m = M-1,\cdots,2,1 \tag{11.45}$$

最后，使用近似的最速下降法更新权值和偏置值：

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha\mathbf{s}^m(\mathbf{a}^{m-1})^T \tag{11.46}$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha\mathbf{s}^m \tag{11.47}$$

### 11.2.3 例子

下面我们选择一个网络并将 BP 算法用在其上来解决一个特定问题。首先，采用本章开始时讨论的 1－2－1 网络。为方便起见，将此网络重画于图 11-10 中。

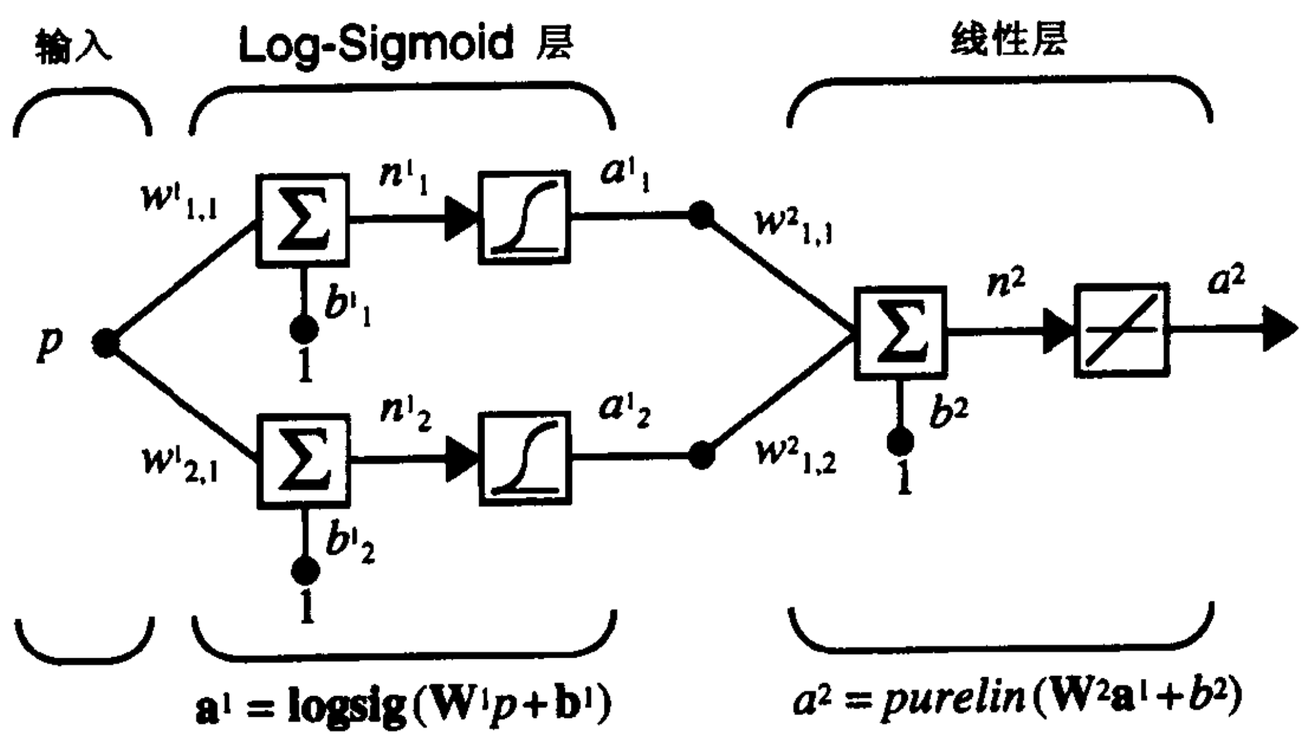



图 11-10 用网络逼近函数的例子

下一步定义此网络要解决的问题。假定我们用此网络来逼近函数

$$g(p) = 1 + \sin\left(\frac{\pi}{4}p\right),\ -2 \leqslant p \leqslant 2 \tag{11.48}$$

训练集可以通过计算函数在几个 $p$ 值上的函数值来得到。

在开始 BP 算法前，需要选择网络权值和偏置值的初始值。通常选择较小的随机值。下一章将讨论为什么要这样做。现在，选择的值 11-14

$$\mathbf{W}^1(0)=\begin{bmatrix}-0.27\\-0.41\end{bmatrix},\mathbf{b}^1(0)=\begin{bmatrix}-0.48\\-0.13\end{bmatrix},\mathbf{W}^2(0)=[0.09\ \ -0.17],\mathbf{b}^2(0)=[0.48]$$

网络对这些初始值的响应如图 11-11 所示，图中还包括要逼近的正弦函数的曲线。

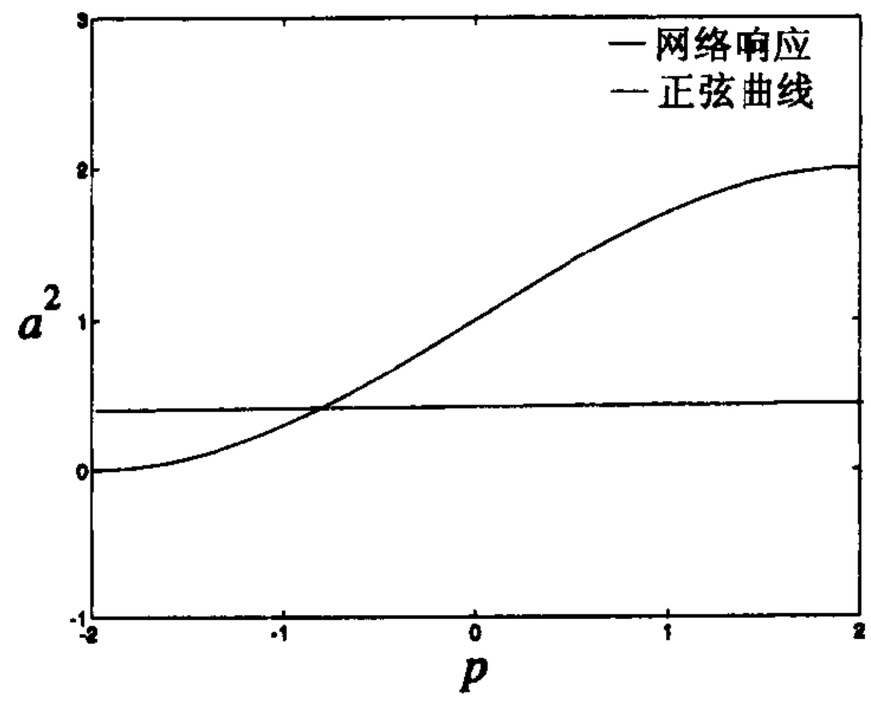



图 11-11 网络对初始值的响应

现在可以开始执行算法了。对初始输入，我们选择 $p=1$：

$$a^0=p=1$$

第一层的输出为

$$\mathbf{a}^1=\mathbf{f}^1(\mathbf{W}^1\mathbf{a}^0+\mathbf{b}^1)=\mathbf{logsig}\left(\begin{bmatrix}-0.27\\-0.41\end{bmatrix}[1]+\begin{bmatrix}-0.48\\-0.13\end{bmatrix}\right)=\mathbf{logsig}\left(\begin{bmatrix}-0.75\\-0.54\end{bmatrix}\right)$$

$$=\begin{bmatrix}\dfrac{1}{1+e^{0.75}}\\\dfrac{1}{1+e^{0.54}}\end{bmatrix}=\begin{bmatrix}0.321\\0.368\end{bmatrix}$$

第二层的输出为

$$a^2=f^2(\mathbf{W}^2\mathbf{a}^1+\mathbf{b}^2)=purelin\left([0.09\quad -0.17]\begin{bmatrix}0.321\\0.368\end{bmatrix}+[0.48]\right)=[0.446]$$

11-15 误差将为

$$e=t-a=\left\{1+\sin\left(\frac{\pi}{4}p\right)\right\}-a^2=\left\{1+\sin\left(\frac{\pi}{4}1\right)\right\}-0.446=1.261$$

算法的下一阶段是反向传播敏感性值。在开始反向传播前，需要先求传输函数的导数 $\dot{f}^1(n)$和 $\dot{f}^2(n)$。对第一层：

$$\dot{f}^1(n)=\frac{\mathrm{d}}{\mathrm{d}n}\left(\frac{1}{1+e^{-n}}\right)=\frac{e^{-n}}{(1+e^{-n})^2}=\left(1-\frac{1}{1+e^{-n}}\right)\left(\frac{1}{1+e^{-n}}\right)=(1-a^1)(a^1)$$

对第二层：

$$\dot{f}^2(n)=\frac{\mathrm{d}}{\mathrm{d}n}(n)=1$$

下面可以执行反向传播了。起始点在第二层。由式(11.44)：

$$\mathbf{s}^2=-2\dot{\mathbf{F}}^2(\mathbf{n}^2)(\mathbf{t}-\mathbf{a})=-2[\dot{f}^2(n^2)](1.261)=-2[1](1.261)=-2.522$$

第一层的敏感性由计算第二层的敏感性反向传播得到，由(11.45)式：

$$\mathbf{s}^1 = \dot{\mathbf{F}}^1(\mathbf{n}^1)(\mathbf{w}^2)^T\mathbf{s}^2 = \begin{bmatrix} (1-a_1^1)(a_1^1) & 0 \\ 0 & (1-a_2^1)(a_2^1) \end{bmatrix}\begin{bmatrix} 0.09 \\ -0.17 \end{bmatrix}[-2.522]$$

$$= \begin{bmatrix} (1-0.321)(0.321) & 0 \\ 0 & (1-0.368)(0.368) \end{bmatrix}\begin{bmatrix} 0.09 \\ -0.17 \end{bmatrix}[-2.522]$$

$$= \begin{bmatrix} 0.218 & 0 \\ 0 & 0.233 \end{bmatrix}\begin{bmatrix} -0.227 \\ 0.429 \end{bmatrix} = \begin{bmatrix} -0.0495 \\ 0.0997 \end{bmatrix}$$

算法的最后阶段是更新权值。为简单起见，学习速度设为 $\alpha = 0.1$。(学习速度的选择将在第 12 章中作更详细的讨论。)由式(11.46)和(11.47)：

$$\mathbf{W}^2(1) = \mathbf{W}^2(0) - \alpha\mathbf{s}^2(\mathbf{a}^1)^T = [0.09 \quad -0.17] - 0.1[-2.522][0.321 \quad 0.368]$$
$$= [0.171 \quad -0.0772]$$

11-16

$$\mathbf{b}^2(1) = \mathbf{b}^2(0) - \alpha\mathbf{s}^2 = [0.48] - 0.1[-2.522] = [0.732]$$

$$\mathbf{W}^1(1) = \mathbf{W}^1(0) - \alpha\mathbf{s}^1(\mathbf{a}^0)^T = \begin{bmatrix} -0.27 \\ -0.41 \end{bmatrix} - 0.1\begin{bmatrix} -0.0495 \\ 0.0997 \end{bmatrix}[1] = \begin{bmatrix} -0.265 \\ -0.420 \end{bmatrix}$$

$$\mathbf{b}^1(1) = \mathbf{b}^1(0) - \alpha\mathbf{s}^1 = \begin{bmatrix} -0.48 \\ -0.13 \end{bmatrix} - 0.1\begin{bmatrix} -0.0495 \\ 0.0997 \end{bmatrix} = \begin{bmatrix} -0.475 \\ -0.140 \end{bmatrix}$$

这就完成了 BP 算法的第一次迭代。下一步可以选择另一个输入 $p$，执行算法的第二次迭代过程。迭代过程一直进行下去，直到网络响应和目标函数之差达到某一可接受的水平。在第 12 章中将对收敛准则作更详细的讨论。

试验在此两层网络上使用 BP 算法请用 *Neural Network Design Demonstration Backpropagation Calculation* (**nnd11bc**)。

### 11.2.4 反向传播

本节中将叙述与反向传播法的实际实现相关的一些问题，包括网络结构的选择、网络收敛性和一般化的问题。(第 12 章中将再次讨论实现问题，讨论改进算法的过程。)

1. 网络结构的选择

如本章前面所述，多层网络可用来逼近几乎任一个函数，只要在隐层中有足够的神经元。然而，通常并不能说，多少层或多少神经元就足以得到足够的性能。本节中我们通过一些例子来考察这个问题。

第一个例子：假定要逼近如下的函数：

$$g(p) = 1 + \sin\left(\frac{i\pi}{4}p\right), \quad -2 \leqslant p \leqslant 2 \tag{11.49}$$

其中 $i$ 取值 1，2，4 和 8。随 $i$ 的增加，函数变得更为复杂，在 $-2 \leqslant p \leqslant 2$ 的区间内将有更多的正弦波周期。当 $i$ 增加时，很难用隐层中神经元数目固定的神经网络来逼近 $g(p)$。 11-17

对这个例子，我们使用一个 1－3－1 网络，第一层的传输函数为对数－S 形，第二层的传输函数是线性函数。根据 11.2.1 节中函数逼近的例子，这种两层网络的响应是三个对数－S 形函数之和(或多个对数－S 形函数之和，只要隐层中有同样多的神经元)。显然，对这个网络能实现的函数有多么复杂有一个限制。图 11-12 是网络经训练来逼近 $g(p)$(对 $i = 1$，

2，4，8)后的响应曲线。最终的网络响应曲线用图中画出的曲线来表示。

可以看到，对 $i=4$，这个 1－3－1 网络达到了它的最大能力。当 $i>4$ 时，网络不能产生 $g(p)$精确的逼近曲线。从图 11-12 右下方的图中可以看到 1－3－1 网络试图逼近 $i=8$ 时的函数 $g(p)$。网络的响应和 $g(p)$之间的均方误差达到了最小化，但网络响应曲线只能与函数的一小部分相匹配。

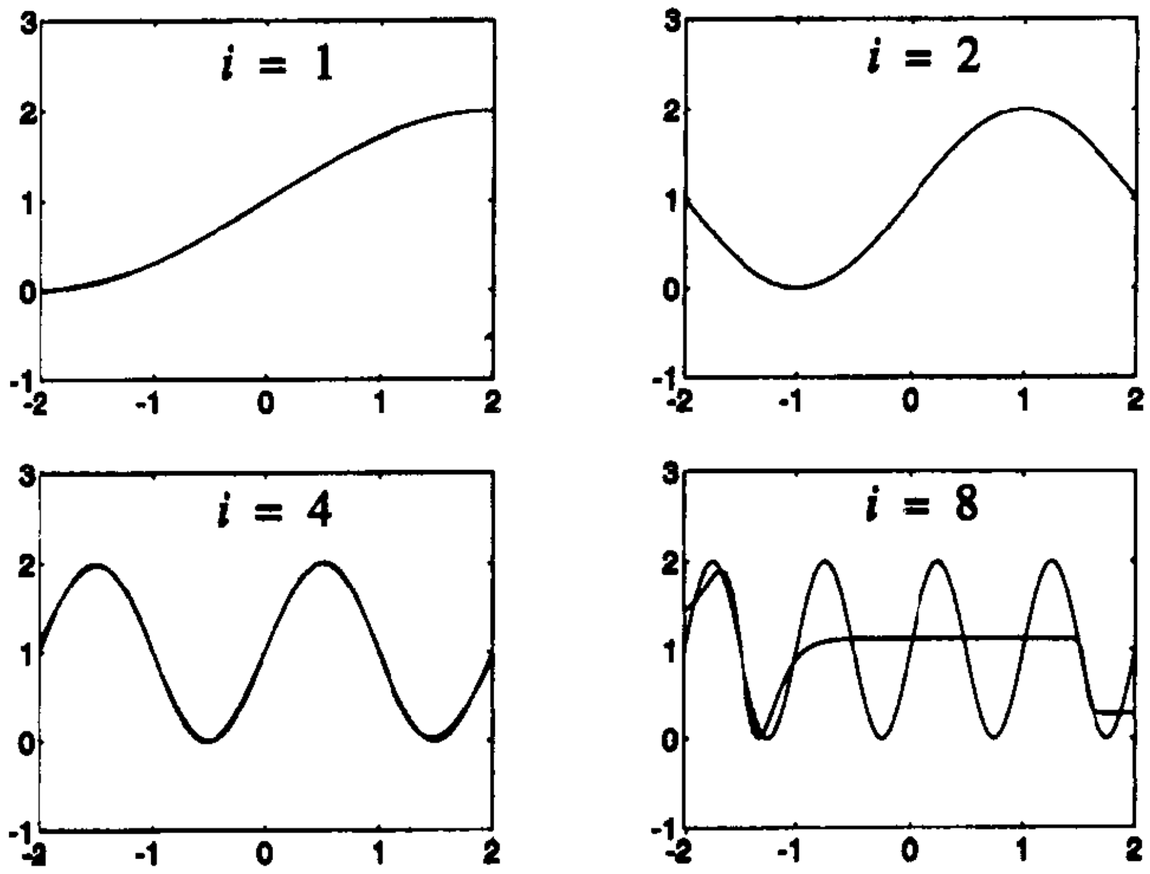



图 11-12 用 1－3－1 网络作函数逼近

下一个例子中将从一个稍有些不同的角度来解决此问题。这次我们选择函数 $g(p)$，然后使用越来越大的网络直到能精确地逼近函数为止。$g(p)$采用

$$g(p) = 1 + \sin\left(\frac{6\pi}{4}p\right), \ -2 \leqslant p \leqslant 2 \tag{11.50}$$

11-18 我们用两层网络来逼近此函数，第一层的传输函数是对数－S 形函数，第二层的是线性函数(1－$S^1$－1 网络)。如本章前面所述，网络的响应是 $S^1$S 形函数的迭加。

图 11-13 为第一层(隐层)的神经元数目增加时的网络响应曲线。除非网络隐层中至少有 5 个神经元，否则网络不能精确地表示 $g(p)$。

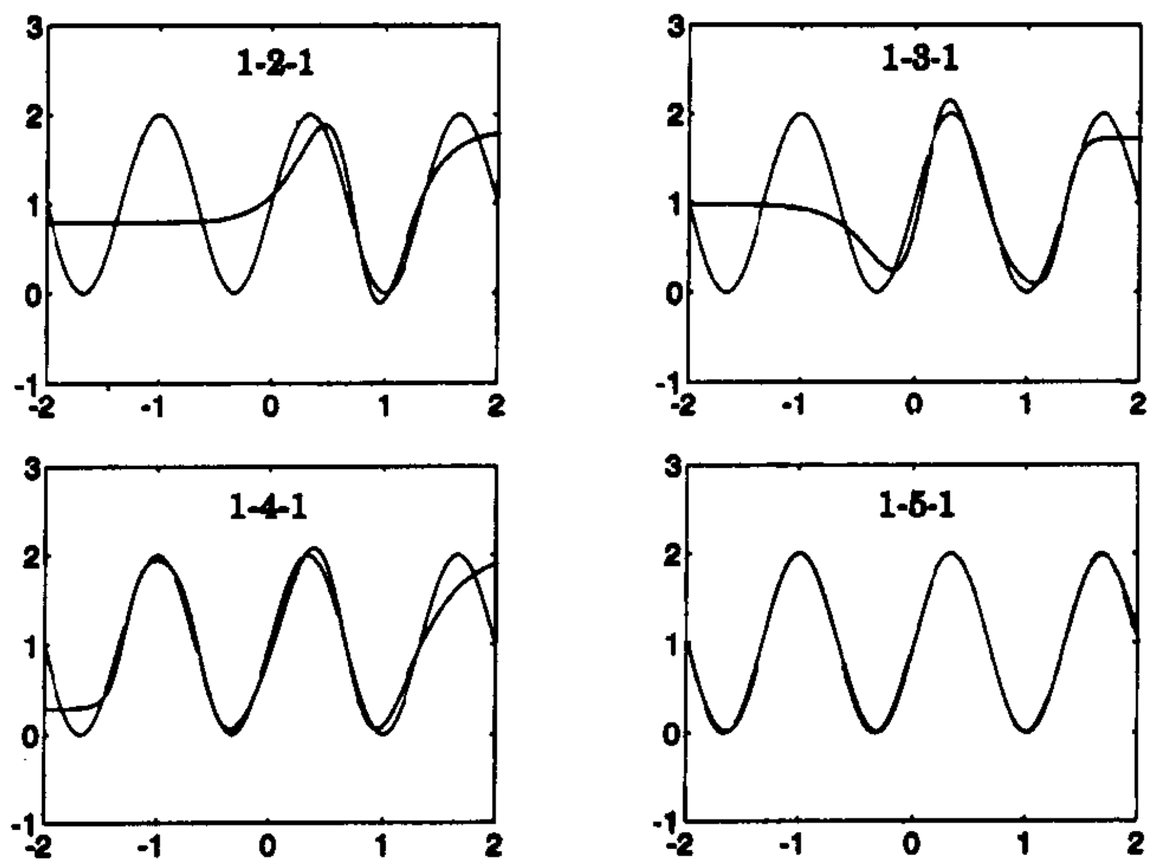
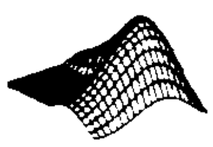



图 11-13 增加隐层中的神经元数目的影响

总结起来说，1 - $S^1$ - 1 网络在隐层中为 S 形神经元而在输出层中为线性神经元时，可以产生 $S^1$S 形函数相叠加的网络响应曲线。若要逼近有大量拐点的函数，隐层中就要有大量的神经元。



使用 *Neural Network Design Demonstration Function Approximation* (**nnd11fa**)可以达到对两层网络能力更深的认识。

2. 收敛性

在前一节给出的一些例子中，尽管 BP 算法可以获得使均方误差最小化的网络参数，网络的响应却不能精确地逼近所期望的函数。这是由于网络的能力受隐层中神经元数目的限制。本节将给出一个例子，其中网络能逼近函数，但学习算法不能产生精确逼近解的网络参数。下一章将更详细地讨论这个问题，并解释为什么会这样。现在先来叙述这个问题。 11-19

网络要逼近的函数为

$$g(p) = 1 + \sin(\pi p),\ -2 \leqslant p \leqslant 2 \tag{11.51}$$

我们用一个 1 - 3 - 1 网络来逼近此函数，其中第一层的传输函数是对数-S 形函数，第二层的是线性函数。

图 11-14 说明学习算法收敛到使均方误差最小的一个解的情况。细线表示中间迭代结果，粗线表示最终解，此时算法收敛。(每条曲线旁边的数字表示迭代的顺序，0 表示初始条件，5 表示最终解。这些曲线没有列出对应的迭代次数，数字仅表示一个顺序。)

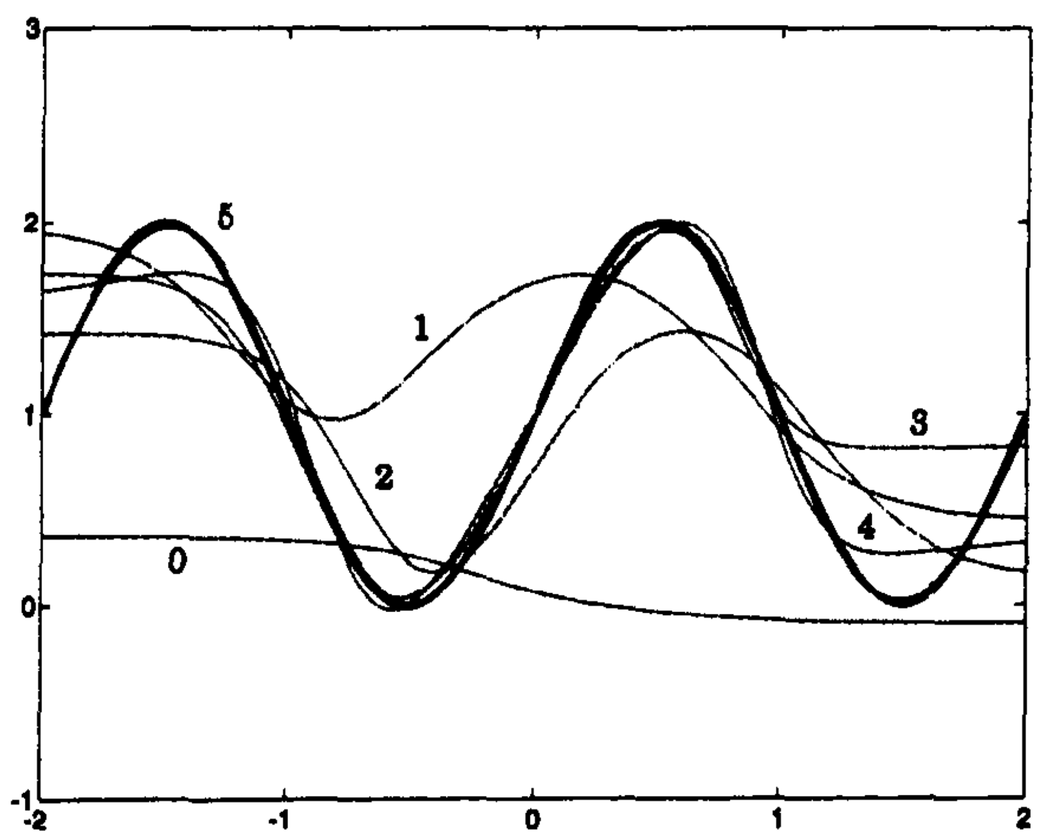



图 11-14 收敛到全局最小值

图 11-15 说明学习算法收敛到一个解但均方误差并没有被最小化的一种情况。粗线(标记为 5)代表最终的迭代中的网络响应。在最终的迭代计算中，均方误差的梯度为 0，因而得到一个局部极小值，但正如图 11-14 中表示的，存在一个更好的解。图 11-15 中的结果与图 11-14 中的结果之间的差别仅仅是初始条件。从一个初始条件开始，算法收敛到全局极小值点，而从另一个初始条件开始，算法收敛到一个局部极小值点。 11-20

注意 LMS 算法不会产生这样的结果。ADALINE 网络中均方误差性能指标是只有一个极小值点的二次函数(在大多数条件下)。因而只要学习速率足够小，LMS 算法保证收敛到全局极小值。通常，多层网络的均方误差非常复杂且有许多局部极小值(在下一章中将看到

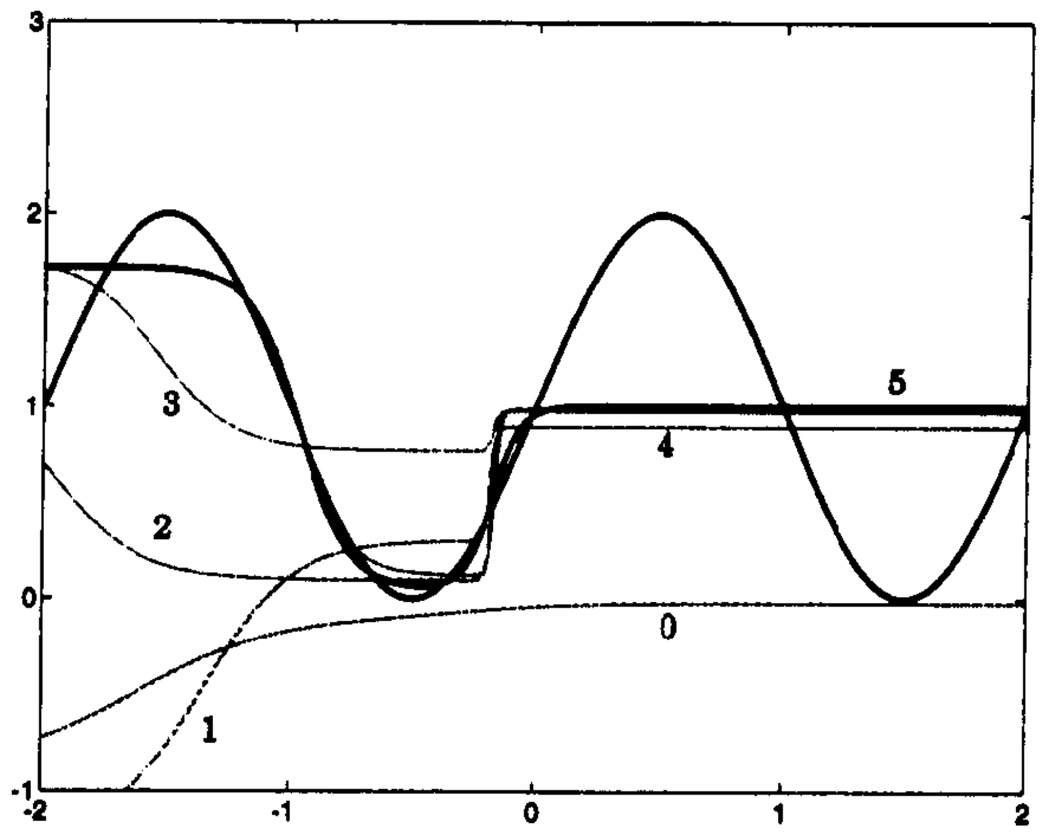



图 11-15 收敛到局部极小值

这一点)。当 BP 算法收敛时，我们并不能确定是否求到了最优解。最好的办法是多试几个不同的初始条件以保证得到最优的解。

3. 推广

在大多数情况下，用有限多个正确网络行为的例子来训练多层网络：

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \cdots, \{\mathbf{p}_Q, \mathbf{t}_Q\} \tag{11.52}$$

这个训练集常常代表一类大得多的可能的输入/输出对。因而网络将它学习到的例子成功地推广到总体这一点是重要的。

例如，假定训练集是通过采样下面的函数得到的：

$$g(p) = 1 + \sin\left(\frac{\pi}{4}p\right) \tag{11.53}$$

采样点为 $p = -2, -1.6, -1.2, \cdots, 1.6, 2$(总共有 11 个输入/输出对)。在图 11-16 中，
11-21 可以看到经这些数据训练后的 1－2－1 网络的响应。细线代表 $g(p)$，粗线代表网络的响应，符号“+”表示训练集。

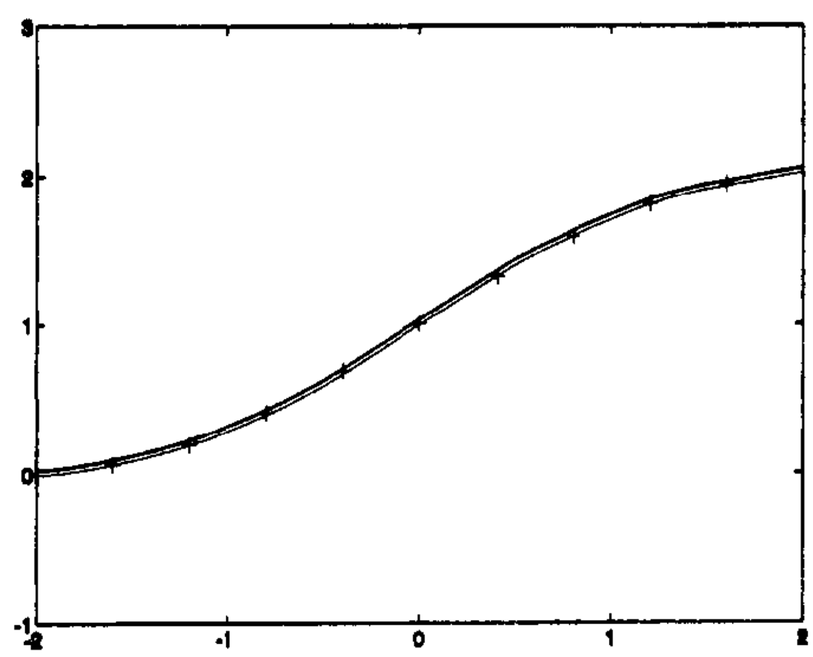



图 11-16 用 1－2－1 网络逼近函数 $g(p)$

可以看到，网络响应曲线可以很精确地表示 $g(p)$。若要求网络在 $p$ 点的响应值，而 $p$ 点不包含在训练集中(如 $p = -0.2$)，网络仍将产生接近于 $g(p)$的输出。网络的推广结果很好。

再看图 11-17，它表示使用同样的数据集训练一个 1－9－1 网络后得到的网络响应结果。注意在所有训练点上，网络的响应很精确地逼近 $g(p)$。然而，若我们计算不包含在训练集中的点 $p$(如 $p=-0.2$)对应的网络响应，网络所产生的结果可能与 $g(p)$的结果差得很远。因而这个网络没有被很好地推广。

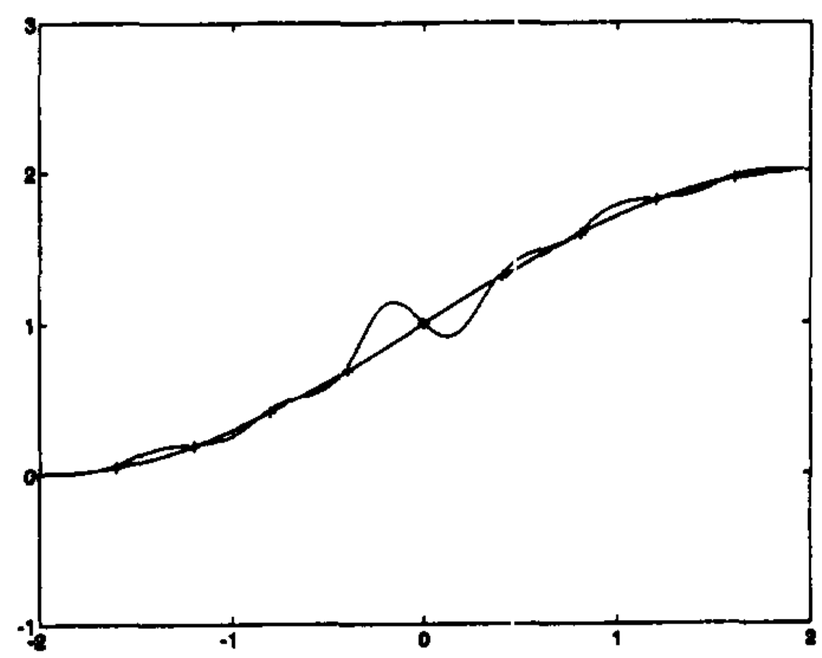



图 11-17 用 1－9－1 网络逼近 $g(p)$

11-22

对此问题，1－9－1 网络又过于灵活了；它总共有 28 个可调节的参数(18 个权值和 10 个偏置值)，然而在训练集中只有 11 个数据点。1－2－1 网络只有 7 个参数，因而它能实现的函数类型非常受限。

一个网络要能被推广，它应当具有比训练集中的数据点少的参数。在神经网络中，正如在所有建模问题中，要用足以表示训练集的最简单的网络。只要有一个更小的网络能工作，就不要使用更大的网络(常被称作 Ockham 的“剃刀”)。

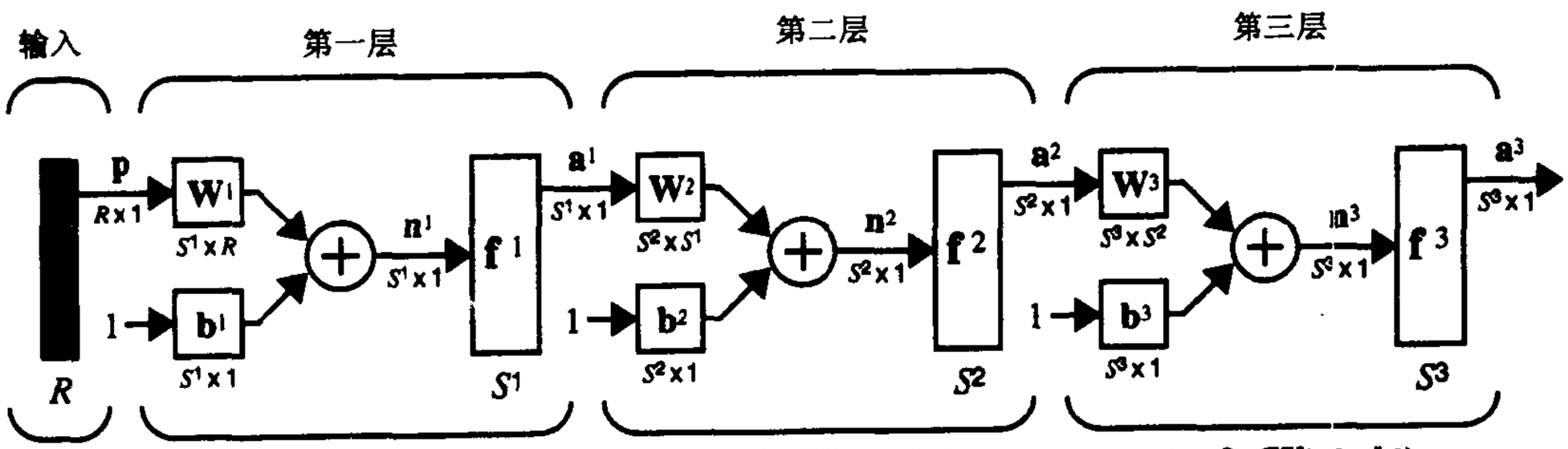
若不使用最简单的网络，那么另一种办法是在网络得到恰当调整后停止训练。这个过程的介绍以及其他提高网络推广性能的技术可参见第 19 章。

试验神经网络的推广请用 *Neural Network Design Demonstration* **Generalization** (**nnd11gn**)。

11-23

## 11.3 小结

### 多层网络




$$\mathbf{a}^1=\mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1) \qquad \mathbf{a}^2=\mathbf{f}^2(\mathbf{W}^2\mathbf{a}^1+\mathbf{b}^2) \qquad \mathbf{a}^3=\mathbf{f}^3(\mathbf{W}^3\mathbf{a}^2+\mathbf{b}^3)$$

$$\mathbf{a}^3=\mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$

### 反向传播算法

**性能指标**

$$F(\mathbf{x})=E[\mathbf{e}^T\mathbf{e}]=E[(\mathbf{t}-\mathbf{a})^T(\mathbf{t}-\mathbf{a})]$$

**逼近性能指标**

$$\hat{F}(\mathbf{x}) = \mathbf{e}^T(k)\mathbf{e}(k) = (\mathbf{t}(k) - \mathbf{a}(k))^T(\mathbf{t}(k) - \mathbf{a}(k))$$

**敏感性**

$$\mathbf{s}^m \equiv \frac{\partial \hat{F}}{\partial n^m} = \begin{bmatrix} \frac{\partial \hat{F}}{\partial n_1^m} \\ \frac{\partial \hat{F}}{\partial n_2^m} \\ \vdots \\ \frac{\partial \hat{F}}{\partial n_{S^m}^m} \end{bmatrix}$$

11-24

**前向传播**

$$\mathbf{a}^0 = \mathbf{p}$$

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^m + \mathbf{b}^{m+1}), m = 0,1,\cdots,M-1$$

$$\mathbf{a} = \mathbf{a}^M$$

**反向传播**

$$\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a})$$

$$\mathbf{s}^m = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T\mathbf{s}^{m+1}, m = M-1,\cdots,2,1$$

这里

$$\dot{\mathbf{F}}^m(\mathbf{n}^m) = \begin{bmatrix} \dot{f}^m(n_1^m) & 0 & \cdots & 0 \\ 0 & \dot{f}^m(n_2^m) & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & & \dot{f}^m(n_{sm}^m) \end{bmatrix}$$

$$\dot{f}^m(n_j^m) = \frac{\partial f^m(n_j^m)}{\partial n_j^m}$$

**权值更新(近似最速下降法)**

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha\mathbf{s}^m(\mathbf{a}^{m-1})^T$$

11-25

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha\mathbf{s}^m$$

## 11.4 例题

**P11.1** 考虑图 11-18 中的两类模式，类Ⅰ表示垂直线，类Ⅱ表示水平线。

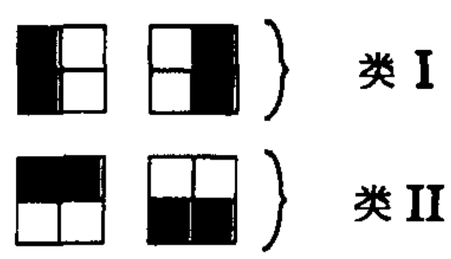



图 11-18 例题 P11.1 的模式类别

(i) 这些类别是否是线性可分的?

（ii）设计一个多层网络来区分这些类别。

**解**

（i）首先通过依次扫描模式中的各列将模式表示成向量。每个白的方块用“-1”表示，黑的方块用“1”表示。垂直线(类Ⅰ的模式)则表示为

$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \quad 和 \quad \mathbf{p}_2 = \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$$

水平线(类Ⅱ的模式)表示为

$$\mathbf{p}_3 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} \quad 和 \quad \mathbf{p}_4 = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}$$

为了使这些类别线性可分，必须能在这两个类别之间放置一个超平面。即是，必须有一个权值矩阵 $\mathbf{W}$ 和偏置值 $b$ 满足

$$\mathbf{W}\mathbf{p}_1 + b > 0,\ \mathbf{W}\mathbf{p}_2 + b > 0,\ \mathbf{W}\mathbf{p}_3 + b < 0,\ \mathbf{W}\mathbf{p}_4 + b < 0$$

这些条件可转化为 11-26

$$\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} = \begin{bmatrix} w_{1,1} + w_{1,2} - w_{1,3} - w_{1,4} \end{bmatrix} > 0$$

$$\begin{bmatrix} -w_{1,1} - w_{1,2} + w_{1,3} + w_{1,4} \end{bmatrix} > 0$$

$$\begin{bmatrix} w_{1,1} - w_{1,2} + w_{1,3} - w_{1,4} \end{bmatrix} < 0$$

$$\begin{bmatrix} -w_{1,1} + w_{1,2} - w_{1,3} + w_{1,4} \end{bmatrix} < 0$$

前两个条件可化简为

$$w_{1,1} + w_{1,2} > w_{1,3} + w_{1,4} \text{ 和 } w_{1,3} + w_{1,4} > w_{1,1} + w_{1,2}$$

这是矛盾的。后两个条件可化简为

$$w_{1,1} + w_{1,3} > w_{1,2} + w_{1,4} \text{ 和 } w_{1,2} + w_{1,4} > w_{1,1} + w_{1,3}$$

这也是矛盾的。因此，没有超平面可以将这两个类别分开。

（ii）有许多多层网络可解决此问题。设计网络时首先注意到，对类Ⅰ的向量，或者是前面两个元素，或者是后面两个元素为“1”。类Ⅱ的向量具有“1”和“-1”交替出现的模式。因而所设计的网络如图 11-19。 11-27

第一层中的第一个神经元测试输入向量的前两个元素。若它们均为“1”，则输出“1”，否则输出“-1”。第一层中的第二个神经元测试输入向量的后两个元素。第一层中的神经元均执行 AND 操作。第二层网络测试第一层的输出是否为“1”。它执行 OR 操作。这样，当输入向量前两个元素或后两个元素都为“1”时，网络将输出“1”。

**P11.2** 图 11-20 中为一个分类问题，类Ⅰ向量代表空心圆，类Ⅱ向量代表实心圆。这些类别不是线性可分的。设计一个能将它们正确分类的多层网络。

我们将用一个能用于任意分类问题的过程来解决这个问题。它需要一个三层网络，每一

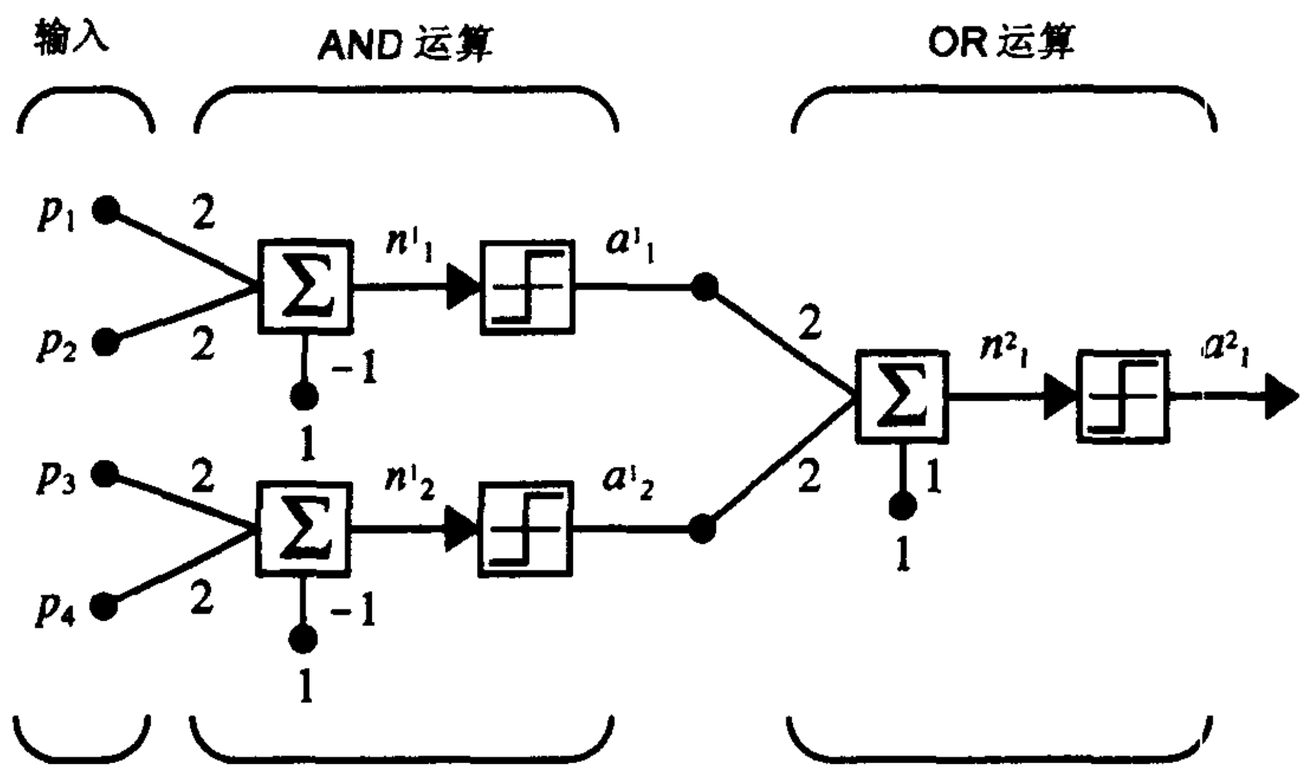



图 11-19 区分水平线和垂直线的网络

层的神经元数目是固定的(hard-limiting)。第一层产生将类 I 向量和类 II 向量分开的线性判定边界集合。这个问题中要用 11 个这样的边界，如图 11-21 所示。

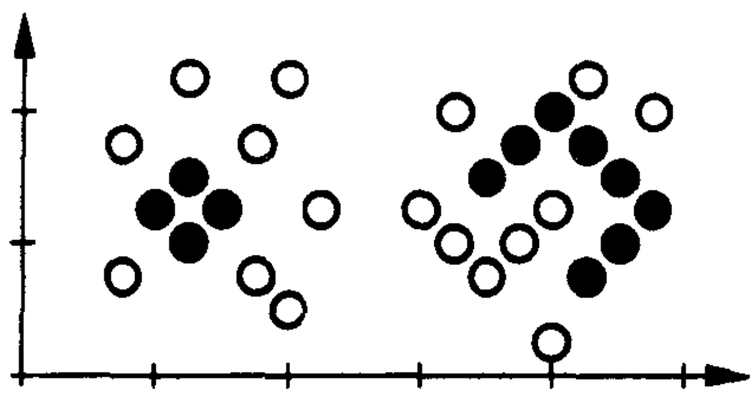


图 11-20 分类问题

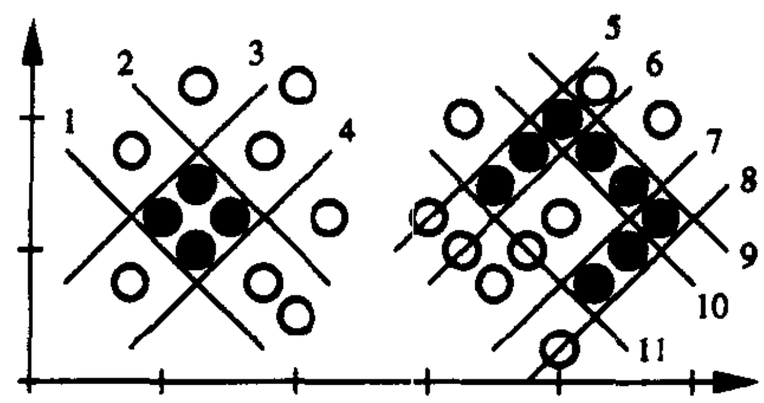



图 11-21 第一层判定边界

第一层权值矩阵中的每一行对应于一个判定边界。第一层的权值矩阵和偏置值为

11-28
$$(\mathbf{W}^1)^T = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 \end{bmatrix}$$

$$(\mathbf{b}^1)^T = [-2\ 3\ 0.5\ 0.5\ -1.75\ 2.25\ -3.25\ 3.75\ 6.25\ -5.75\ -4.75]$$

(回想第 3，4 章和第 10 章中对一个给定判定边界计算合适的权值矩阵和偏置值的过程。)下面可以用第二层的 AND 神经元将第一层 11 个神经元的输出划分为组。AND 神经元跟我们在例题 P11.1 中第一层网络所使用的一样。第二层的权值矩阵和偏置值为

$$\mathbf{W}^2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}, \quad \mathbf{b}^T = \begin{bmatrix} -3 \\ -3 \\ -3 \\ -3 \end{bmatrix}$$

第二层的四个判定边界如图 11-22 所示。例如，神经元 2 的判定边界由第一层的边界 5，6，9 和 11 组合而成。这可以在 $\mathbf{W}^2$ 的第 2 行中看到。

第三层中，我们将第二层的四个判定区域通过 OR 操作结合成一个判定区域。如例题

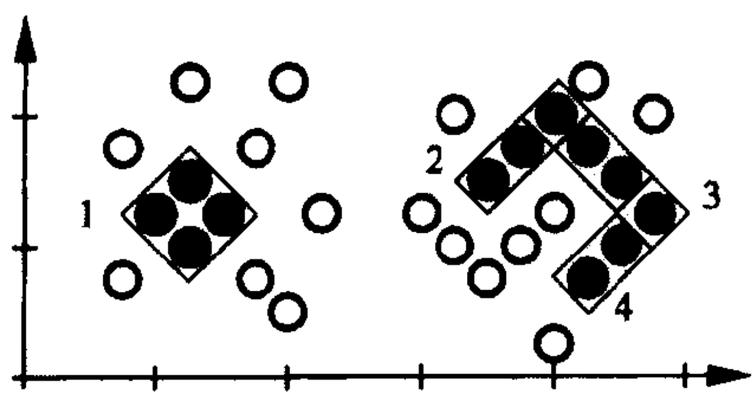



图 11-22 第二层判定区域

P11.1 中最后一层网络那样。第三层的权值矩阵和偏置值为

$$\mathbf{W}^3 = [1 \quad 1 \quad 1 \quad 1],\ \mathbf{b}^3 = [3]$$

整个网络如图 11-23。

只要在隐层中有足够的神经元，设计上面网络的过程可用来解决具有任意的判定边界分类问题。办法是用第一层网络产生一定数量的线性边界，然后在第二层用 AND 神经元，第三层中用 OR 神经元，将它们结合起来。第二层的决策区域是凸的，但第三层产生的最终的判定边界可以是任意形状的。 11-29

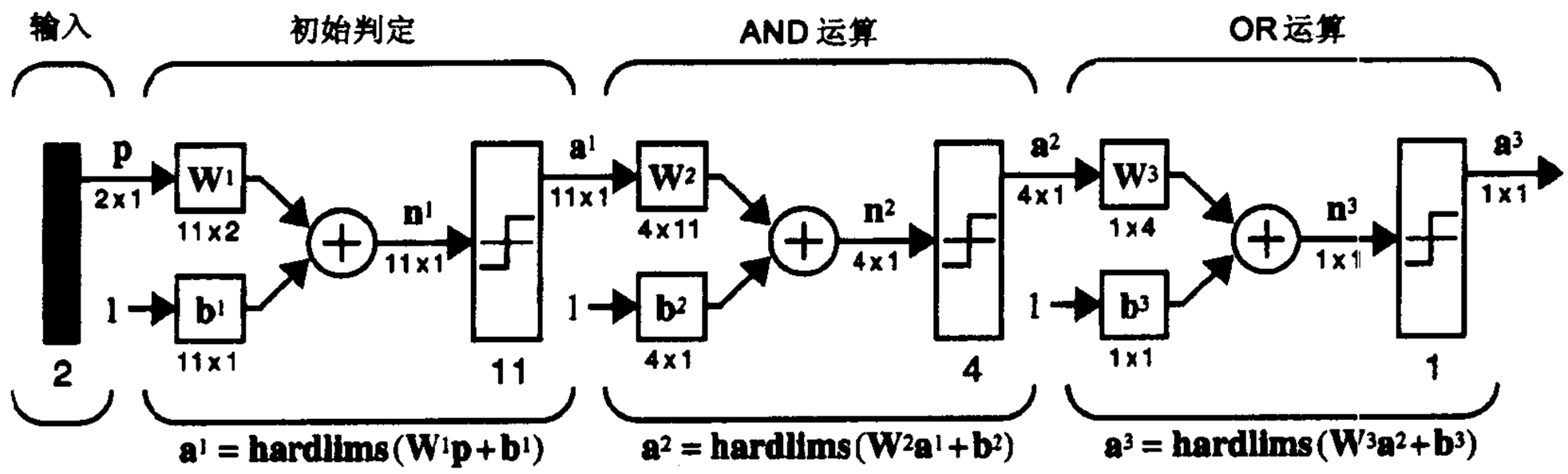



图 11-23 例题 P11.2 的网络

网络的最终判定区域由图 11-24 给出。在阴影区域的任向向量将产生网络输出 1，它对应于类Ⅱ。任何其他向量产生网络输出 - 1，它对应于类Ⅰ。

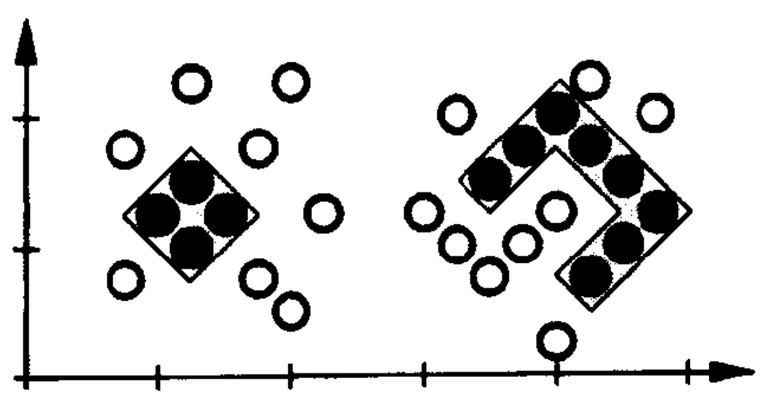


图 11-24 最终的判定边界

**P11.3** 说明具的线性传输函数的多层网络等价于单层线性网络。

**解**

对多层线性网络，前向传播等式为

$$\mathbf{a}^1 = \mathbf{W}^1\mathbf{p} + \mathbf{b}^1$$

$$\mathbf{a}^2 = \mathbf{W}^2\mathbf{a}^1 + \mathbf{b}^2 = \mathbf{W}^2\mathbf{W}^1\mathbf{p} + [\mathbf{W}^2\mathbf{b}^1 + \mathbf{b}^2]$$

$$\mathbf{a}^3 = \mathbf{W}^3\mathbf{a}^2 + \mathbf{b}^3 = \mathbf{W}^3\mathbf{W}^2\mathbf{W}^1\mathbf{p} + [\mathbf{W}^3\mathbf{W}^2\mathbf{b}^1 + \mathbf{W}^3\mathbf{b}^2 + \mathbf{b}^3]$$

继续下去可以看到，对 $M$ 层的线性网络，等价的单层线性网络将具有如下的权值矩阵和偏置值向量：

11-30

$$\mathbf{W} = \mathbf{W}^M\mathbf{W}^{M-1}\cdots\mathbf{W}^2\mathbf{W}^1$$

$$\mathbf{b} = [\mathbf{W}^M\mathbf{W}^{M-1}\cdots\mathbf{W}^2]\mathbf{b}^1 + [\mathbf{W}^M\mathbf{W}^{M-1}\cdots\mathbf{W}^3]\mathbf{b}^2 + \cdots + \mathbf{b}^M$$

**P11.4** 此问题的目的是说明链法则的使用。考虑如下的动态系统：

$$y(k+1) = f(y(k))$$

要求选择初始条件 $y(0)$，使得在某一终止时刻 $k = K$，系统的输出 $y(K)$将尽可能地接近某一目标输出 $t$。我们将用最速下降法使性能指标最小化。性能指标为

$$F(y(0)) = (t - y(K))^2$$

为此需求得梯度

$$\frac{\partial}{\partial y(0)}F(y(0))$$

寻求一个用链法则计算它的过程。

**解**

梯度为

$$\frac{\partial}{\partial y(0)}F(y(0)) = \frac{\partial(t - y(K))^2}{\partial y(0)} = 2(t - y(K))\left[-\frac{\partial}{\partial y(0)}y(K)\right]$$

关键项为

$$\left[\frac{\partial}{\partial y(0)}y(K)\right]$$

而它不能被直接求得，因为 $y(K)$并不是 $y(0)$的显式函数。先定义一个中间项

$$r(k) \equiv \frac{\partial}{\partial y(0)}y(k)$$

这样就可以使用链法则：

$$r(k+1) = \frac{\partial}{\partial y(0)}y(k+1) = \frac{\partial y(k+1)}{\partial y(k)} \times \frac{\partial y(k)}{\partial y(0)} = \frac{\partial y(k+1)}{\partial y(k)} \times r(k)$$

从系统的动态方程可知

$$\frac{\partial y(k+1)}{\partial y(k)} = \frac{\partial f(y(k))}{\partial y(k)} = \dot{f}(y(k))$$

11-31 因此，计算 $r(k)$的递归等式为

$$r(k+1) = \dot{f}(y(k))r(k)$$

在 $k=0$ 的初始值为

$$r(0) = \frac{\partial y(0)}{\partial y(0)} = 1$$

于是，计算梯度的整个过程为

$$r(0) = 1$$

$$r(k+1) = \dot{f}(y(k))r(k),\ k = 0,1,\cdots,\ K-1$$

$$\frac{\partial}{\partial y(0)}F(y(0)) = 2(t - y(K))[-r(K)]$$

**P11.5** 考虑图 11-25 中的两层网络，初始的权值和偏置值设为

$$w^1 = 1, b^1 = 1, w^2 = -2, b^2 = 1$$

一个输入/目标对为

$$((p = 1),(t = 1))$$

(i) 求均方误差$(e)^2$，它为所有权值和偏置值的显式函数。

(ii) 用(i)题的结果来求初始权值和偏置值下的$\partial(e)^2/\partial w^1$。

(iii) 用反向传播算法重复(ii)题，并比较结果。

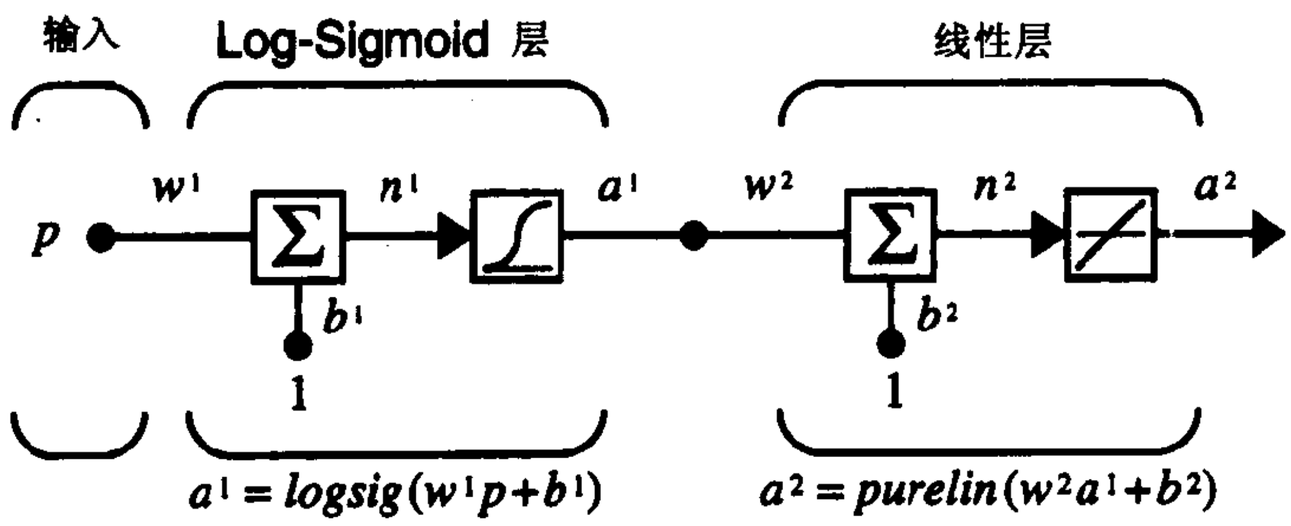



图 11-25 例题 P11.5 的两层网络

**解** 11-32

(i) 均方误差为

$$(e)^2 = (t - a^2)^2 = \left(t - \left\{w^2 \frac{1}{(1 + \exp(-(w^1 p + b^1)))} + b^2\right\}\right)^2$$

(ii) 导数为

$$\frac{\partial(e)^2}{\partial w^1} = 2e\frac{\partial e}{\partial w^1} = 2e\left\{w^2 \frac{1}{(1 + \exp(-(w^1 p + b^1)))^2}\exp(-(w^1 p + b^1))(-p)\right\}$$

为了计算在初始权值和偏置值下的这个导数值，求

$$a^1 = \frac{1}{(1 + \exp(-(w^1 p + b^1)))} = \frac{1}{(1 + \exp(-(1(1) + 1)))} = 0.8808$$

$$a^2 = w^2 a^1 + b^2 = (-2)0.8808 + 1 = -0.7616$$

$$e = (t - a^2) = (1 - (-0.7616)) = 1.7616$$

$$\frac{\partial(e)^2}{\partial w^1} = 2e\left\{w^2 \frac{1}{(1 + \exp(-(w^1 p + b^1)))^2}\exp(-(w^1 p + b^1))(-p)\right\}$$

$$= 2(1.7616)\left\{(-2)\frac{1}{(1 + \exp(-(1(1) + 1)))^2}\exp(-(1(1) + 1))(-1)\right\}$$

$$= 3.5232\left(0.2707\frac{1}{(1.289)^2}\right) = 0.7398$$

(iii) 使用式(11.44)和(11.45)进行敏感性反向传播算法：

$$\mathbf{s}^2 = -2\dot{\mathbf{F}}^2(\mathbf{n}^2)(\mathbf{t} - \mathbf{a}) = -2(1)(1 - (-0.7616)) = -3.5232$$

$$\mathbf{s}^1 = \dot{\mathbf{F}}^1(\mathbf{n}^1)(\mathbf{W}^2)^T\mathbf{s}^2 = [a^1(1 - a^1)](-2)\mathbf{s}^2$$

$$= [0.8808(1 - 0.8808)](-2)(-3.5232) = 0.7398$$

由式(11.23)可计算$\partial(e)^2/\partial w^1$：

$$\frac{\partial(e)^2}{\partial w^1} = s^1 a^0 = s^1 p = (0.7398)(1) = 0.7398$$

11-33 这与我们在(ii)题中得到的结果一致。

**P11.6** 在本章前面我们已经表明，若神经元传输函数为对数-S 形函数

$$a = f(n) = \frac{1}{1+e^{-n}}$$

则通过下式可方便地求得导数

$$\dot{f}(n) = a(1-a)$$

寻找一种方便的办法求双曲正切 S 形函数的导数：

$$a = f(n) = tansig(n) = \frac{e^n - e^{-n}}{e^n + e^{-n}}$$

**解**

直接计算导数得

$$\dot{f}(n) = \frac{df(n)}{dn} = \frac{d}{dm}\left(\frac{e^n - e^{-n}}{e^n + e^{-n}}\right) = -\left(\frac{e^n - e^{-n}}{e^n + e^{-n}}\right)^2 (e^n - e^{-n}) + \frac{e^n + e^{-n}}{e^n + e^{-n}}$$

$$= 1 - \frac{(e^n - e^{-n})^2}{(e^n + e^{-n})^2} = 1 - (a)^2$$

**P11.7** 对图 11-26 中的网络，初始权值和偏置值为

$$w^1(0) = -1,\ b^1(0) = 1,\ w^2(0) = -2,\ b^2(0) = 1$$

一个输入/目标对是

$$((p = -1),(t = 1))$$

11-34 设 $\alpha = 1$，执行一次反向传播算法迭代。

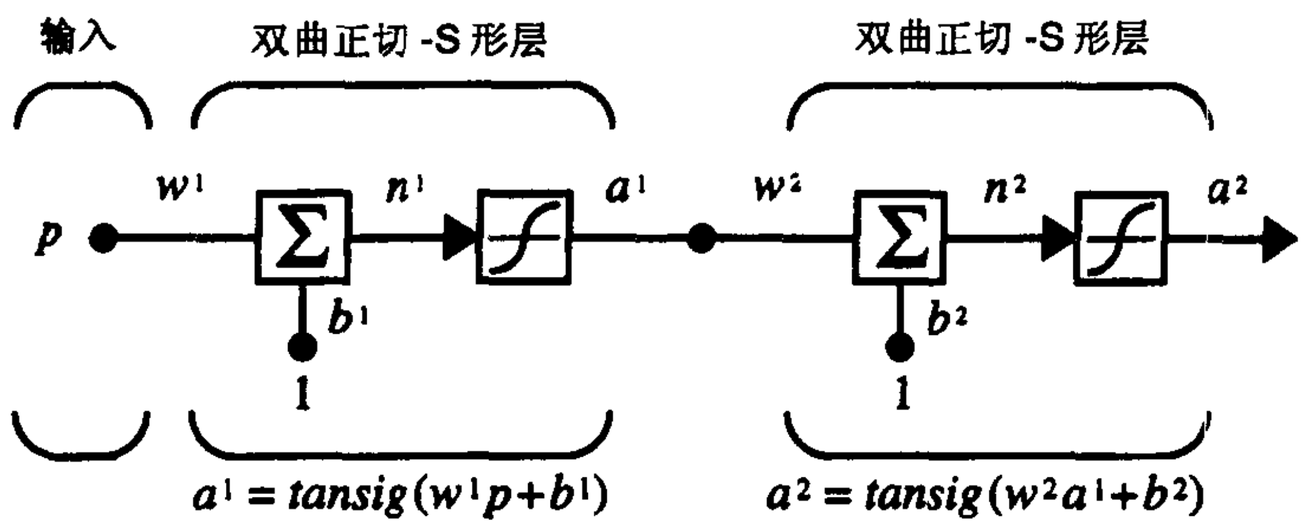



图 11-26 两层双曲正切-S 形网络

**解**

第一步是通过网络传播输入。

$$n^1 = w^1 p + b^1 = (-1)(-1) + 1 = 2$$

$$a^1 = tansig(n^1) = \frac{\exp(n^1) - \exp(-n^1)}{\exp(n^1) + \exp(-n^1)} = \frac{\exp(2) - \exp(-2)}{\exp(2) + \exp(-2)} = 0.964$$

$$n^2 = w^2 a^1 + b^2 = (-2)(0.964) + 1 = -0.928$$

$$a^2 = tansig(n^2) = \frac{\exp(n^2) - \exp(-n^2)}{\exp(n^2) + \exp(-n^2)} = \frac{\exp(-0.928) - \exp(0.928)}{\exp(-0.928) + \exp(0.928)} = -0.7297$$

$$e = (t - a^2) = (1 - (-0.7297)) = 1.7297$$

现在用式(11.44)和(11.45)反传敏感性：

$$\mathbf{s}^2 = -2\dot{\mathbf{F}}^2(\mathbf{n}^2)(\mathbf{t} - \mathbf{a}) = -2[1 - (a^2)^2](e) = -2[1 - (-0.7297)^2]1.7297$$
$$= -1.6175$$
$$\mathbf{s}^1 = \dot{\mathbf{F}}^1(\mathbf{n}^1)(\mathbf{W}^2)^T\mathbf{s}^2 = [1 - (a^1)^2]w^2\mathbf{s}^2 = [1 - (0.964)^2](-2)(-1.6175)$$
$$= 0.2285$$

11-35

最后，用式(11.46)和(11.47)更新权值和偏置值：

$$w^2(1) = w^2(0) - \alpha s^2(a^1)^T = (-2) - 1(-1.6175)(0.964) = -0.4407$$
$$w^1(1) = w^1(0) - \alpha s^1(a^0)^T = (-1) - 1(0.2285)(-1) = -0.7715$$
$$b^2(1) = b^2(0) - \alpha s^2 = 1 - 1(-1.6175) = 2.6175$$
$$b^1(1) = b^1(0) - \alpha s^1 = 1 - 1(0.2285) = 0.7715$$

**P11.8** 图 11-27 是将标准两层前向传播网络稍作修改得到的网络。从输入有一条到第二层的直接连接。推导此网络的反向传播算法。

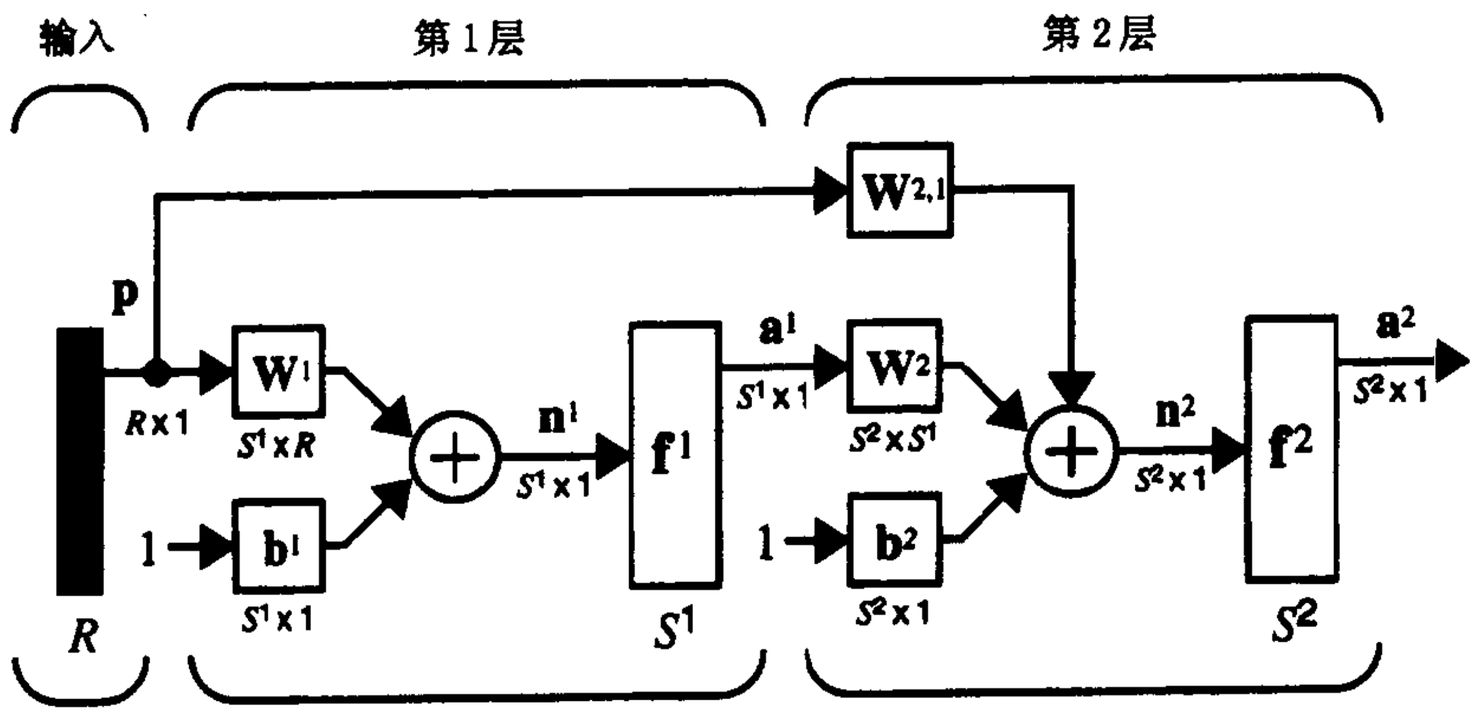



图 11-27 具有旁路连接的网络

**解**

首先，前向方程为

$$\mathbf{n}^1 = \mathbf{W}^1\mathbf{p} + \mathbf{b}^1$$
$$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{n}^1) = \mathbf{f}^1(\mathbf{W}^1\mathbf{p} + \mathbf{b}^1)$$
$$\mathbf{n}^2 = \mathbf{W}^2\mathbf{a}^1 + \mathbf{W}^{2,1}\mathbf{p} + \mathbf{b}^2$$
$$\mathbf{a}^2 = \mathbf{f}^2(\mathbf{n}^2) = \mathbf{f}^2(\mathbf{W}^2\mathbf{a}^1 + \mathbf{W}^{2,1}\mathbf{p} + \mathbf{b}^2)$$

与标准两层网络相比，敏感性的反向传播方程不会改变。敏感性是均方误差对网络输入的导数；由于我们仅仅在网络输入中增加了一项，这些导数不会改变。

11-36

下一步需要求权值更新方程的梯度元素。对标准的权值和偏置值有

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial w_{i,j}^m} = s_i^m a_j^{m-1}$$
$$\frac{\partial \hat{F}}{\partial b_i^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial b_i^m} = s_i^m$$

因此，$\mathbf{W}^1$，$\mathbf{b}^1$，$\mathbf{W}^2$ 和 $\mathbf{b}^2$ 的更新方程不变。需要的是一个额外的 $\mathbf{W}^{2,1}$的方程：

$$\frac{\partial \hat{F}}{\partial w_{i,j}^{2,1}} = \frac{\partial \hat{F}}{\partial n_i^2} \times \frac{\partial n_i^2}{\partial w_{i,j}^{2,1}} = s_i^2 \times \frac{\partial n_i^2}{\partial w_{i,j}^{2,1}}$$

为求等式右边的导数，注意

$$n_i^2 = \sum_{j=1}^{S^1} w_{i,j}^2 a_j^1 + \sum_{j=1}^{R} w_{i,j}^{2,1} p_j + b_i^2$$

因此

$$\frac{\partial n_i^2}{\partial w_{i,j}^{2,1}} = p_j \quad 且 \quad \frac{\partial \hat{F}}{\partial w_{i,j}^{2,1}} = s_i^2 p_j$$

因而更新方程可以写成矩阵形式：

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T, \ m = 1,2$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m, \ m = 1,2$$

$$\mathbf{W}^{2,1}(k+1) = \mathbf{W}^{2,1}(k) - \alpha \mathbf{s}^2 (\mathbf{a}^0)^T = \mathbf{W}^{2,1}(k) - \alpha \mathbf{s}^2 (\mathbf{p})^T$$

此问题的要点是，反传的概念可被用于比标准的多层前馈网络更一般的网络。

**P11.9** 基于反向传播的概念，求一个能更新图 11-28 中所示的递归网络的权值 $w_1$ 和 $w_2$ 的算法。

11-37

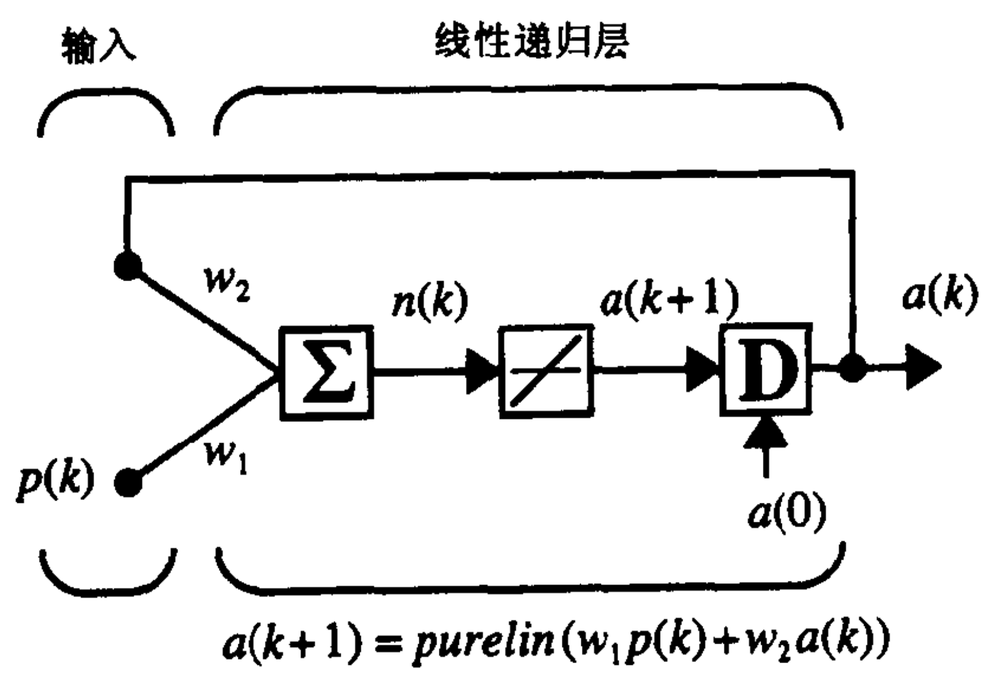



图 11-28 线性递归网络

**解**

第一步是定义性能指数。如同多层网络，我们使用均方误差

$$\hat{F}(\mathbf{x}) = (t(k) - a(k))^2 = (e(k))^2$$

使用最速下降法进行权值更新：

$$\Delta w_i = -\alpha \frac{\partial}{\partial w_i} \hat{F}(\mathbf{x})$$

这些导数可计算如下：

$$\frac{\partial}{\partial w_i} \hat{F}(\mathbf{x}) = \frac{\partial}{\partial w_i} (t(k) - a(k))^2 = 2(t(k) - a(k)) \left\{ -\frac{\partial a(k)}{\partial w_i} \right\}$$

因此，需要计算的关键项是

$$\frac{\partial a(k)}{\partial w_i}$$

要计算这些项，首先需要写出网络方程

$$a(k+1) = purelin(w_1 p(k) + w_2 a(k)) = w_1 p(k) + w_2 a(k)$$

两边对网络权值求导数得：

$$\frac{\partial a(k+1)}{\partial w_1} = p(k) + w_2 \frac{\partial a(k)}{\partial w_1}$$

$$\frac{\partial a(k+1)}{\partial w_2} = a(k) + w_2 \frac{\partial a(k)}{\partial w_2}$$

11-38

(注意我们必须考虑到 $a(k)$本身是 $w_1$ 和 $w_2$ 的函数的事实。)最速下降法中更新权值时使用这两个递归方程来计算导数。方程用

$$\frac{\partial a(0)}{\partial w_1} = 0, \frac{\partial a(0)}{\partial w_2} = 0$$

初始化，这是由于初始条件不是权值的函数。

要说明此过程，先假定 $a(0)=0$。第一次网络更新为

$$a(1) = w_1 p(0) + w_2 a(0) = w_1 p(0)$$

第一个导数为：

$$\frac{\partial a(1)}{\partial w_1} = p(0) + w_2 \frac{\partial a(0)}{\partial w_1} = p(0), \quad \frac{\partial a(1)}{\partial w_2} = a(0) + w_2 \frac{\partial a(0)}{\partial w_2} = 0$$

第一次权值的更新为

$$\Delta w_i = -\alpha \frac{\partial}{\partial w_i} \hat{F}(\mathbf{x}) = -\alpha\left[2(t(1) - a(1))\left\{-\frac{\partial a(1)}{\partial w_i}\right\}\right]$$

$$\Delta w_1 = -2\alpha(t(1) - a(1))\{-p(0)\}$$

$$\Delta w_2 = -2\alpha(t(1) - a(1))\{0\} = 0$$

这个算法属于动态反向传播类型，其中梯度是用不同的方程计算的。

**P11.10** 对单层线性网络(ADALINE)，说明反向传播算法退化为 LMS 算法。

**解**

对单层线性网络，敏感性的计算为：

$$\mathbf{s}^1 = -2\dot{\mathbf{F}}^1(\mathbf{n}^1)(\mathbf{t} - \mathbf{a}) = -2\mathbf{I}(\mathbf{t} - \mathbf{a}) = -2\mathbf{e}$$

权值的更新(式(11.46)和(11.47))为

$$\mathbf{W}^1(k+1) = \mathbf{W}^1(k) - \alpha\mathbf{s}^1(\mathbf{a}^0)^T = \mathbf{W}^1(k) - \alpha(-2\mathbf{e})\mathbf{p}^T = \mathbf{W}^1(k) + 2\alpha\mathbf{e}\mathbf{p}^T$$

$$\mathbf{b}^1(k+1) = \mathbf{b}^1(k) - \alpha\mathbf{s}^1 = \mathbf{b}^1(k) - \alpha(-2\mathbf{e}) = \mathbf{b}^1(k) + 2\alpha\mathbf{e}$$

这与第 10 章中的 LMS 算法相同。

11-39

## 11.5 结束语

本章中讲述了多层感知器网络和反向传播学习规则。多层网络扩展了单层感知器网络，功能更强大。单层网络只能区分线性可分的模式，但多层网络能用于任意的分类问题。此外，多层网络可用作通用的函数逼近器。研究表明，一个两层网络只要在其隐层中有足够的神经元，且隐层神经元的传输函数是 S 形类型的，便可以逼近任何实际的函数。

BP 算法是 LMS 算法的扩展，可用来训练多层网络。LMS 算法和 BP 算法都是使均方误差最小化的最速下降法。它们的惟一区别在于梯度的计算方法。为了计算均方误差对于隐层

中的权值和偏置值的导数，BP 算法使用了链法则。导数首先在网络的最后一层被计算，然后反向传播通过网络，并用链法则计算隐层中的导数，算法也因此而被称为反向传播法。

反向传播的一个主要问题是它需要较长的训练时间。使用基本反向传播算法求解实际问题是不可行的，因为它需要用几周的时间来训练网络，甚至要用大型机。由于反向传播算法首先得以流行，已经有了许多研究加速算法收敛的工作。在第 12 章中，我们将讨论反向
11-40 传播算法收敛慢的原因，并将给出提高算法性能的几种技术。

## 参考文献

[HoSt89] K. M. Hornik, M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359 - 366, 1989.

这篇文章证明了具有任意 squashing 函数的多层前馈网络能逼近任意从一个有限维空间到另一个有限维空间的 Borel 可积函数。

[LeCu85] Y. Le Cun, "Une procedure d'apprentissage pour reseau a seuil assymetrique," *Cognitiva*, vol. 85, pp. 599 - 604, 1985.

Yann Le Cun 几乎与 Parker 和 Rumelhart、Hinton 和 Williams 同时发现了 BP 算法。这篇文章描述了他的算法。

[Park85] D. B. Parker, "Learning-logic: Casting the cortex of the human brain in silicon," Technical Report TR-47, Center for Computational Research in Economics and Management Science, MIT, Cambridge, MA, 1985.

David Parker 大约同一时间独立推导出了 Le Cun 和 Rumelhart、Hinton 和 Williams 发现的 BP 算法，这份报告描述了他的算法。

[RuHi86] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533 - 536, 1986.

这篇文章包含了被广为宣扬的 BP 算法的描述。

[RuMc86] D. E. Rumelhart and J. L. McClelland, eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, Cambridge, MA: MITPress, 1986.

这本书是在 20 世纪 80 年代对神经网络领域研究兴趣的复兴有重要影响的两个事件之一。除其他主题之外，该书给出了训练多层神经网络的 BP 算法。

[Werbo74] P. J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," Ph. D. Thesis, Harvard University, Cambridge, MA, 1974.

11-41 这篇博士论文看起来是第一个对 BP 算法进行描述的文章(尽管没有使用反向传播的名字)。这里，算法是在一般网络的上下文中描述的，而将神经网络作为一个特例。直到 20 世纪 80 年代中期 Rumelhart、Hinton 和 Williams [RuHi86]，David Parker
11-42 [Park85]和 Yann Le Cun [LeCu85]重新发现了 BP 算法，此算法才广为人知。

## 习题

**E11.1** 设计一个能完成图 11-29 中的分类问题的多层网络。只要输入向量在阴影区域(或边界上)，网络应输出 1，否则输出 - 1。

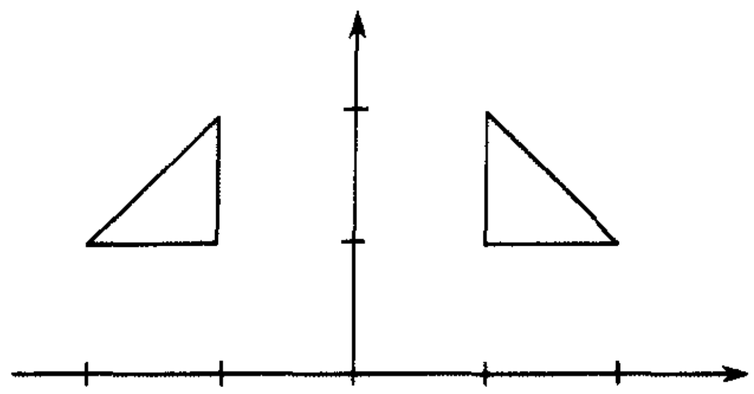


图 11-29 模式分类区域

**E11.2** 求一个与图 11-30 中的网络有相同输入/输出特性的单层网络。

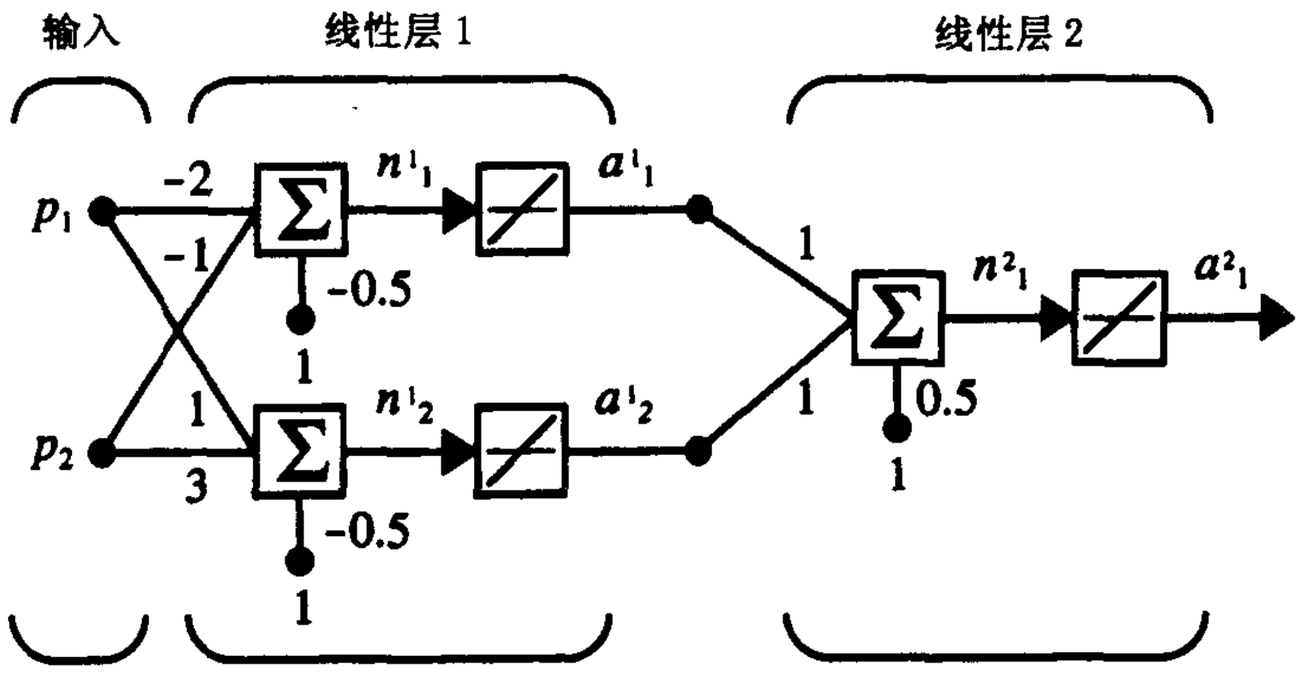



图 11-30 两层线性网络

**E11.3** 选择图 11-4 中的 1－2－1 网络的权值和偏置值，使得网络响应曲线通过图 11-31 中圆圈所指示的点。

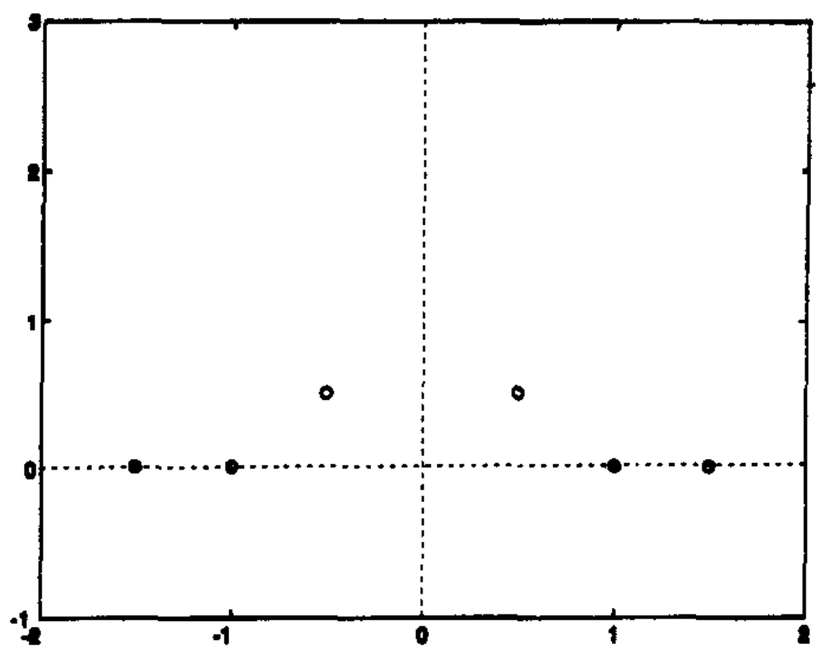


图 11-31 函数逼近习题

使用 *Neural Network Design Demonstration* Two-layer Network Function (**nnd11nf**)来检查你的结果。 11-43

**E11.4** 用链法则来求下面函数的导数$\partial f/\partial w$：

(i) $f(n)=\sin(n)$, $n(w)=w^2$

(ii) $f(n)=\tanh(n)$, $n(w)=5w$

(iii) $f(n)=\exp(n)$, $n(w)=\cos(w)$

(iv) $f(n)=\text{logsig}(n)$，$n(w)=\exp(w)$

**E11.5** 使用下面描述的"反向"法重新计算例题 P11.4。在例题 P11.4 中，动态系统为

$$y(k+1)=f(y(k))$$

我们需要选择初始条件 $y(0)$，使得在某一终止时刻 $k=K$，系统输出 $y(K)$将尽可能地接近目标输出 $t$。我们用最速下降法使性能指标

$$F(y(0))=(t-y(K))^2=e^2(K)$$

最小化，因而需求梯度

$$\frac{\partial}{\partial y(0)}F(y(0))$$

在用链法则计算这个梯度的过程中，涉及到下面项的递归方程：

11-44

$$r(k)=\frac{\partial}{\partial y(0)}y(k)$$

它随时间前进而展开。梯度也可以以时间的反向顺序展开下面项来得到：

$$q(k)\equiv\frac{\partial}{\partial y(k)}e^2(K)$$

**E11.6** 再次考虑 11.2.3 节的反向传播例子。

(i) 求均方误差$(e)^2$，它是所有权值和偏置值的显式函数。

(ii) 用(i)题的结果计算在初始权值和偏置值下的$\partial(e)^2/\partial w_{1,1}^1$。

(iii) 比较(ii)题中的结果和文中由反传算法得到的结果。

**E11.7** 对图 11-32 中的网络，初始权值和偏置值设为

$$w^1(0)=1,b^1(0)=-2,w^2(0)=1,b^2(0)=1$$

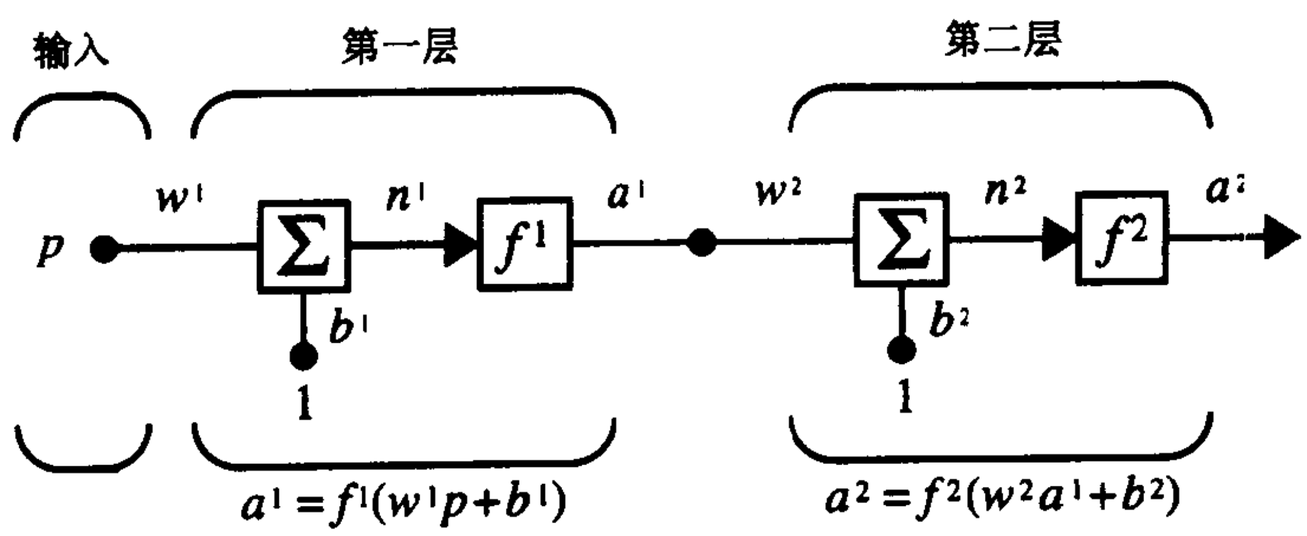
网络传输函数为

$$f^1(n)=(n)^2,f^2(n)=\frac{1}{n}$$

一个输入/目标对为

$$((p=1),(t=1))$$

对 $\alpha=1$ 的反传算法，执行一次迭代。




11-45

图 11-32 习题 E11.7 的两层网络

**E11.8** 对图 11-33 中的网络，神经元传输函数为

$$f^1(n)=(n)^2$$

一个输入/输出对为

$$\left(\mathbf{p}=\begin{bmatrix}1\\1\end{bmatrix}\right),\left(\mathbf{t}=\begin{bmatrix}8\\2\end{bmatrix}\right)$$

对 $\alpha=1$ 的反向传播法，执行一次迭代。

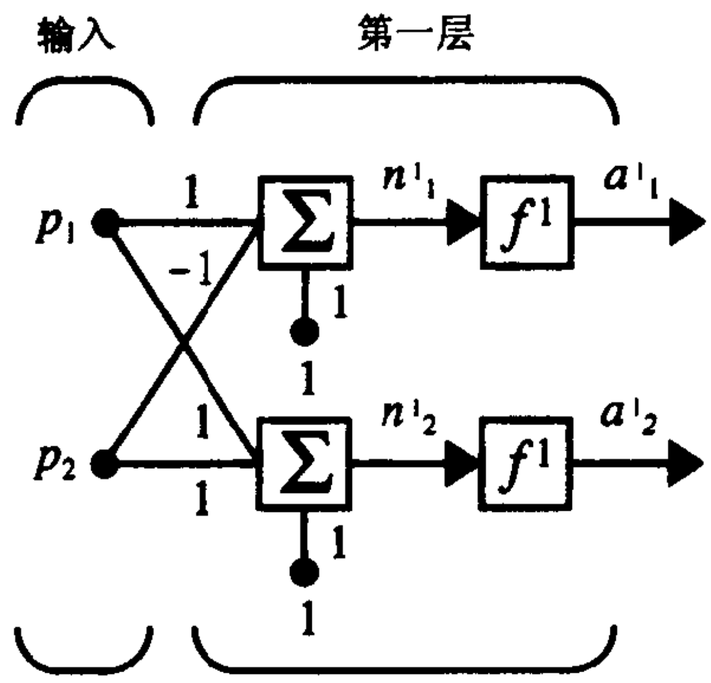



图 11-33 习题 E11.8 的单层网络

**E11.9** 图 11-34 中的网络没有使用我们所用的标准神经元格式。网络输出是网络输入的乘积：

$$a = w_1 p_1 + w_{1,2} p_1 p_2 + w_2 p_2 + b$$

用近似最速下降法，求 $w_1$，$w_{1,2}$，$w_2$ 和 $b$ 的象 BP 算法中所用的那样的学习规则。 11-46

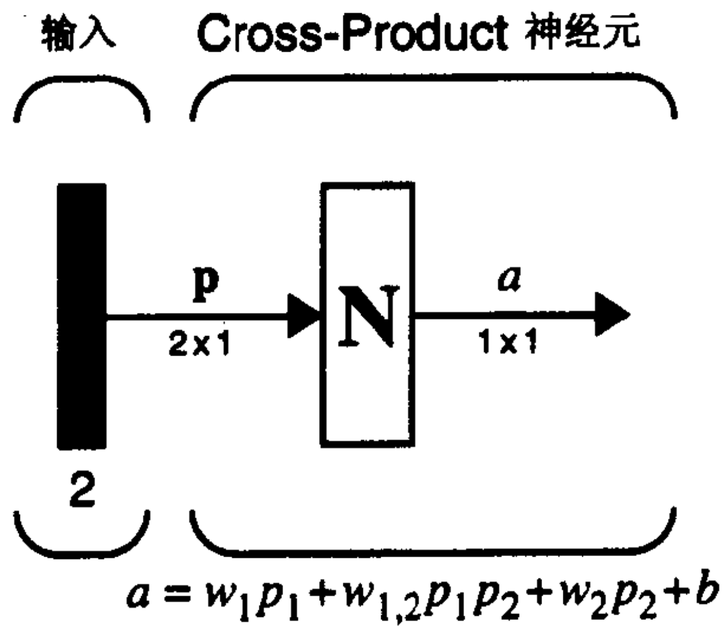



图 11-34 交叉积(Cross-Product)网络

**E11.10** 图 11-35 中为一个两层网络，它有一个从输入端直接到第二层的附加连接。推导此网络的反向传播算法。

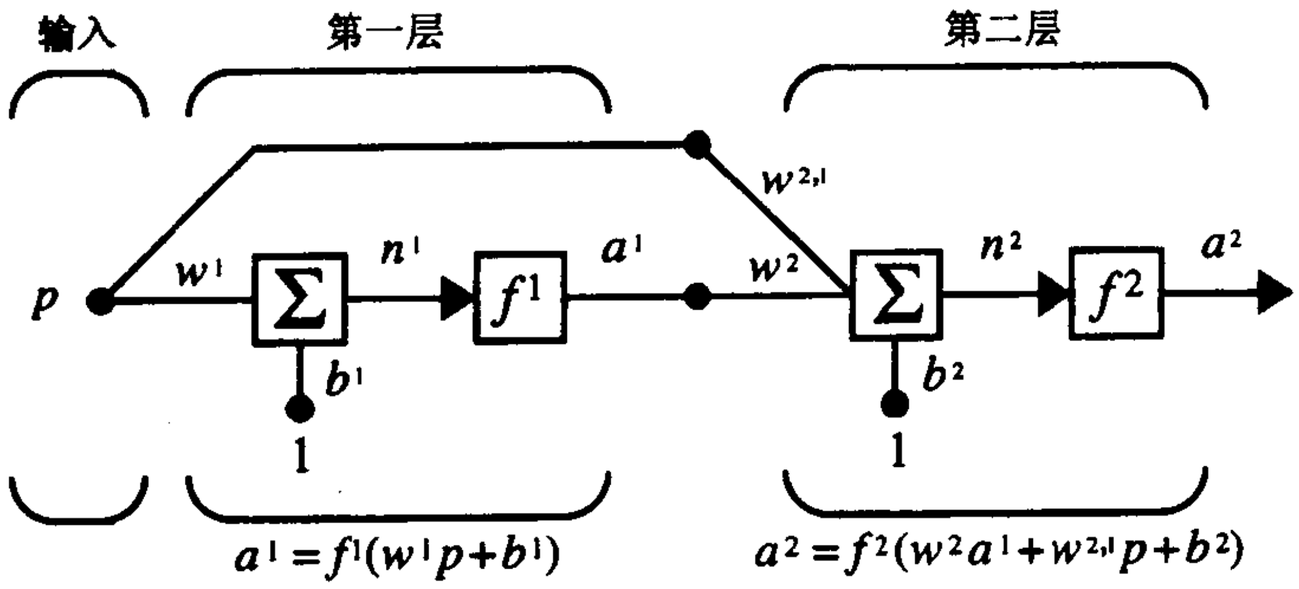



图 11-35 有旁路(bypass)连接的两层网络

**E11.11** 对图 11-6 中的 1－2－1 网络，写一个实现 BP 算法的 MATLAB 程序。初始权值和偏置值设为均匀分布于－0.5 和 0.5 之间的随机数(使用 MATLAB 函数 **rand**)，并训练网络使之逼近函数

$$g(p) = 1 + \sin\left(\frac{\pi}{8}p\right), \ -2 \leq p \leq 2$$

11-47 使用几个不同的初始条件，试验几个不同的学习速率 α。讨论算法的收敛性。

# 第 12 章　反向传播算法的变形

## 12.1　目的

第 11 章中介绍的反向传播算法是神经网络研究中的重大进展。然而，基本的算法对大多实际应用来说都太慢了。本章将介绍一些反向传播算法的变形，能显著提高速度并使算法实用化。

本章将用一个函数近似的例子来集中地说明为什么反向传播算法很慢，接着提供一些算法的改进。这里需要注意反向传播算法是一个近似最速下降的算法。第 9 章中，我们看到最速下降是一个最简单但通常是最慢的最小化方法。共轭梯度算法和牛顿法一般有更快的收敛速度。本章中，将解释如何用这些快速的方法去加速反向传播的收敛速度。 12-1

## 12.2　理论和实例

当基本的反向传播算法应用于实际问题时，训练将花去数天甚至数星期的机时。这引起了对提高算法收敛速度研究的极大热情。

快速算法的研究粗略地分成两类。第一类包括那些使用启发式信息的技术，这源于对标准反向传播算法特定性能的研究。这些启发式技术包括可变的学习速度，使用动量和改变比例变量(例如[VoMa88]，[Jacob88]，[Toll90]和[RiIr90])。本章将讨论动量的使用和可变的学习速度。

另一类研究集中在标准数值优化技术(例如[Shan90]，[Barn92]，[Batt92]和[Char92])。正如第 10 和 11 章讨论的那样，训练前向神经网络减小均方误差只是一个数值优化的问题。由于数值优化做为一个重要的研究课题已经有三四十年了(参见第 9 章)，因而从大量已有的数值优化技术中选择快速训练算法是比较合理的。除非绝对需要，否则没有必要再发明新的训练算法。本章将介绍两个成功的应用于多层感知机训练的算法：共轭梯度算法和 Levenberg-Marquardt 算法(牛顿法的变形)。

**SDBP**　要强调的是本章中描述的所有算法都使用了反向传播过程，所有的导数都是从网络的最后一层处理到网络的第一层。因此，它们都可以被称为“反向传播”算法。算法的区别在于用结果导数来修改权值。在某些情况下，可叹的是我们平常所说的反向传播算法实际上是最速下降算法。为了明确我们的讨论，在本章的其余部分，称基本的反向传播算法为最速下降反传算法(Steepest Descent Backpropagation，SDBP)。

下一节将用一个简单的例子来说明 SDBP 在收敛方面的问题。接着在随后几节中，将提供不同的过程去提高算法的收敛性。 12-2

### 12.2.1　BP 算法的缺点

回忆第 10 章中 LMS 算法在学习速度不大时能保证收敛到具有最小均方误差的解。这是由于对单层线性网络来说均方误差是一个二次函数。二次函数只有一个驻点。另外，二次函

数的赫森矩阵是常数，因此在给定方向的曲率是不变的，并且函数的轮廓线是椭圆。

SDBP 是 LMS 算法的推广。与 LMS 类似，它也是最小均方误差的近似最快下降算法。实际上，在使用单层线性网络时，SDBP 等价于 LMS 算法(请见例题 P11.10)。但在应用于多层网络时，SDBP 的特性完全不同。这是由于单层线性网络和多层非线性网络在均方误差性能曲面上的不同。单层线性网络的均方误差只有一个极小点，并具有常数曲率。但是多层网络的性能曲面可能有多个局部极小点而且在参数空间的不同区域曲率也是变化的。这在下面的例子中将可从清楚地看到。

1. 性能曲面的例子

可以用一个简单的函数逼进的例子来说明多层网络的均方误差性能曲面。这里使用图 12-1 中所示的 1-2-1 网络，其中每层都用对数-S 形传输函数。

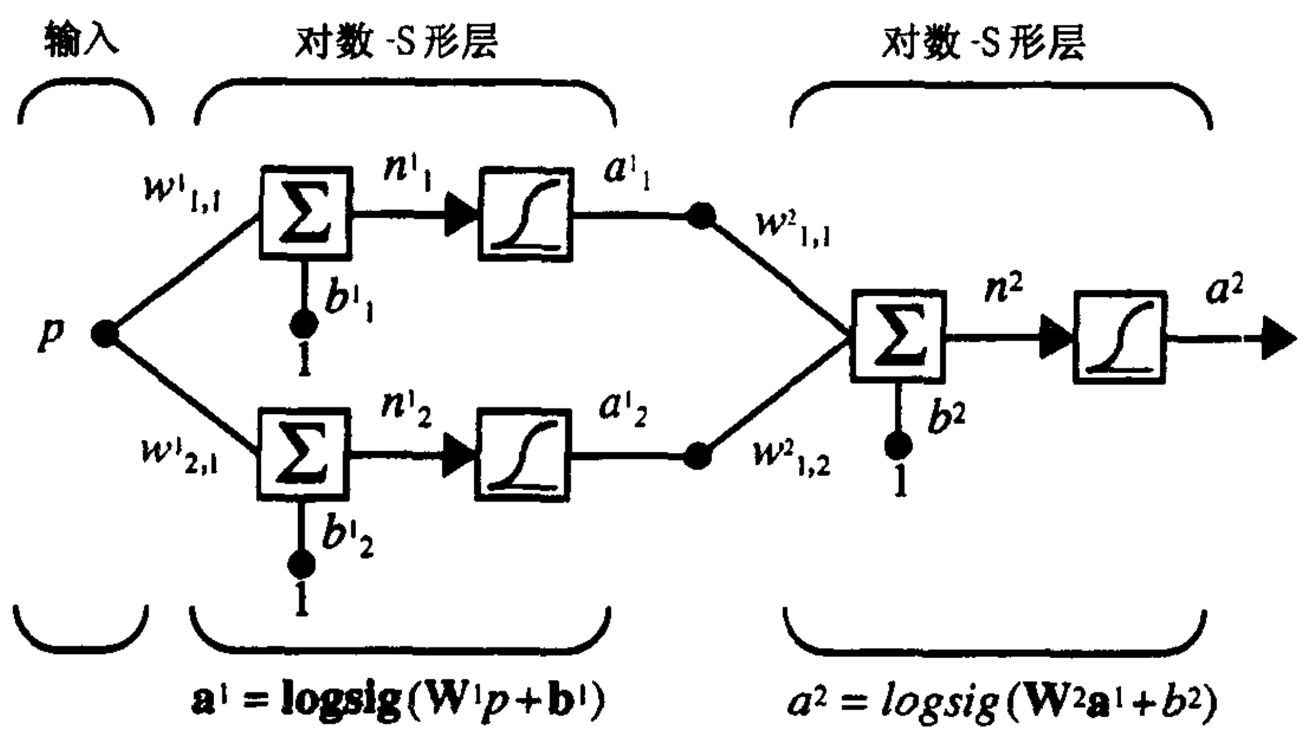



图 12-1 1-2-1 函数逼近网络

12-3 为了简化的分析，对网络给出一个已知优化解的问题。这里要逼近的函数是对图中同一个 1-2-1 网络的响应，具有如下权值和偏置值：

$$w^1_{1,1} = 10, \quad w^1_{2,1} = 10, \quad b^1_1 = -5, \quad b^1_2 = 5 \tag{12.1}$$

$$w^2_{1,1} = 1, \quad w^2_{1,2} = 1, \quad b^2 = -1 \tag{12.2}$$

网络对这些参数的响应如图 12-2 所示，它表示当输入 $p$ 在[-2，2]区间变化时，输出 $a^2$ 的图形。

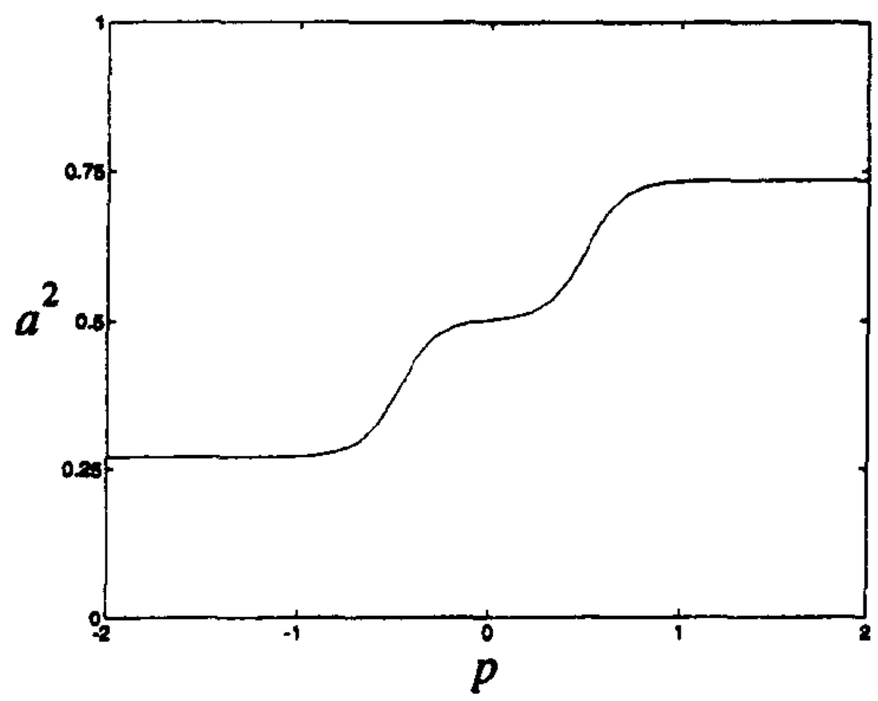



图 12-2 指定函数

我们想训练图 12-1 所示的网络以逼近图 12-2 中所示的函数。当网络参数被设置为(12.1)和(12.2)式中的值时，逼近是精确的。自然这是一个非常不自然的问题，但它很简单并能说明一些重要的概念。

现在来考虑问题的性能指数。假设函数在下述值被采样：

$$p = -2, -1.9, -1.8, \cdots, 1.9, 2 \tag{12.3}$$

并且每一个都以相同的概率发生。性能指数是这 41 个点的平方误差之和。(不必担心求均方误差，它可以通过除以 41 得到。)

为了作出性能指数图，每次只变化两个参数。图 12-3 显示仅当改变 $w_{1,1}^1$和 $w_{1,1}^2$的值而其他参数都设置成式(12.1)和(12.2)中给出的优化值时的平方误差。注意最小的误差是 0，它发生在 $w_{1,1}^1 = 10$ 和 $w_{1,1}^2 = 1$ 时，如图中小圆圈所示。 12-4

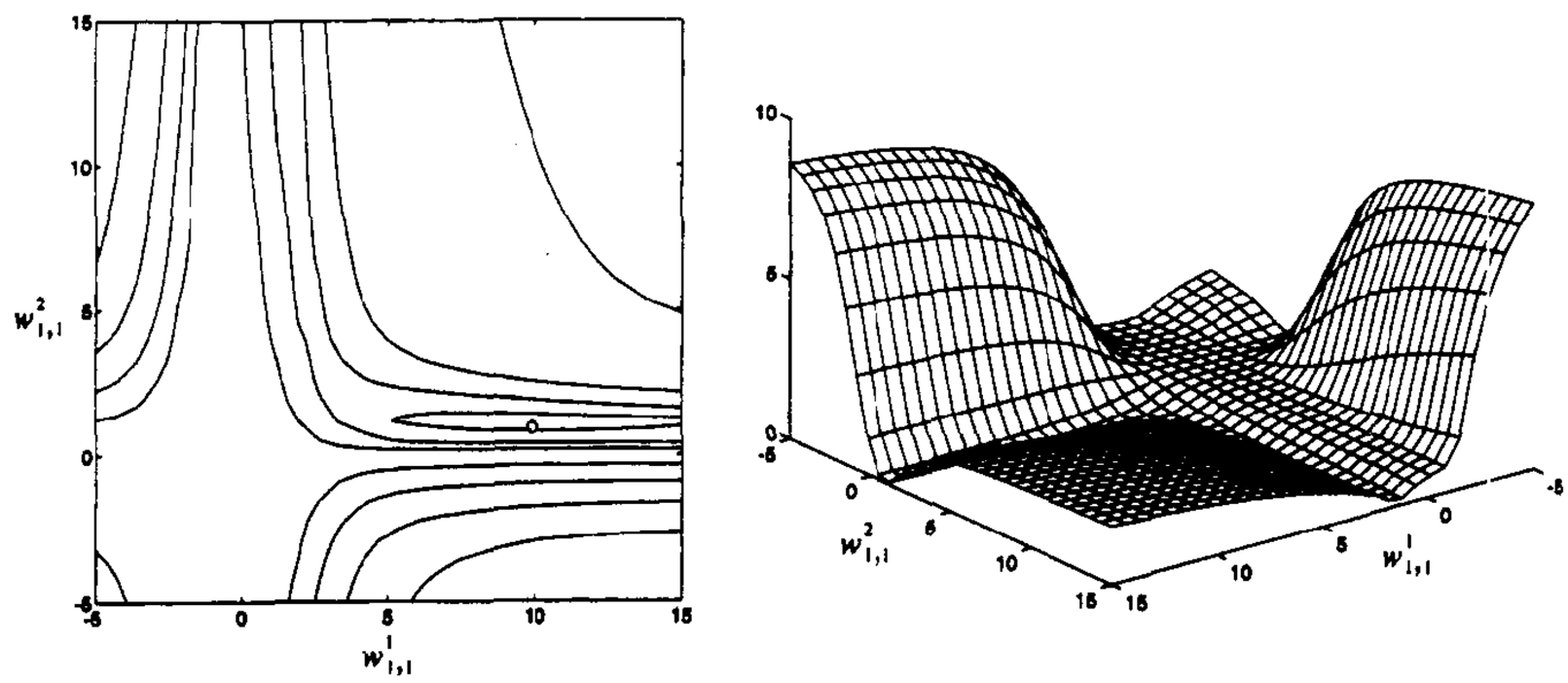



图 12-3　$w_{1,1}^1$和 $w_{1,1}^2$的平方误差曲面

注意该误差曲面中的若干特征。首先，它明显不是一个二次函数，曲率在参数空间中的变化很大。因此，难以为最速下降算法选择一个合适的学习速度。在一些区域曲面非常平坦，这需要一个大的学习速度，同时在其他区域曲率很高，这需要一个小的学习速度。(参考第 9 章和第 10 章关于最速下降算法的学习速度选择的讨论。)

需要注意的是，在给定了网络的 S 形传输函数时，平坦的性能曲面区域并不是意料之外的，S 形函数对大的输入是非常平坦的。

该误差曲面图的另一个特征是存在多个局部极小点。沿着平行于 $w_{1,1}^1$轴的谷，在 $w_{1,1}^1 = 10$和 $w_{1,1}^2 = 1$ 有全局极小。然而在沿着平行于 $w_{1,1}^2$轴的谷也有一个局部极小点(该局部极小点的位置是 $w_{1,1}^1 = 0.88$ $w_{1,1}^2 = 38.6$)。下一节中将研究该曲面上反向传播算法的性能。

图 12-4 指出了当其他参数设置为优化值时，$w_{1,1}^1$和 $b_1^1$ 变化时的平方误差。注意：最小误差是 0，出现在 $w_{1,1}^1 = 10$ 和 $b_1^1 = -5$ 时，由图中小圆圈表示。

可以发现曲面具有非常扭曲的形状：在一些区域很弯，在另一些区域很平坦。用标准的最速下降算法处理这个曲面时必定会碰到困难。例如，如果以 $w_{1,1}^1 = 0$，$b_1^1 = -10$ 作为初始值，梯度接近于 0，即使是没有靠近局部极小点，最速下降算法此时也将停滞。 12-5

图 12-5 指出了当其他参数设置到它们的优化值而 $b_1^1$ 和 $b_2^1$ 变化时的平方误差。在 $b_1^1 = -5$，$b_2^1 = 5$ 时达到最小误差(如图中小圆圈所示)。

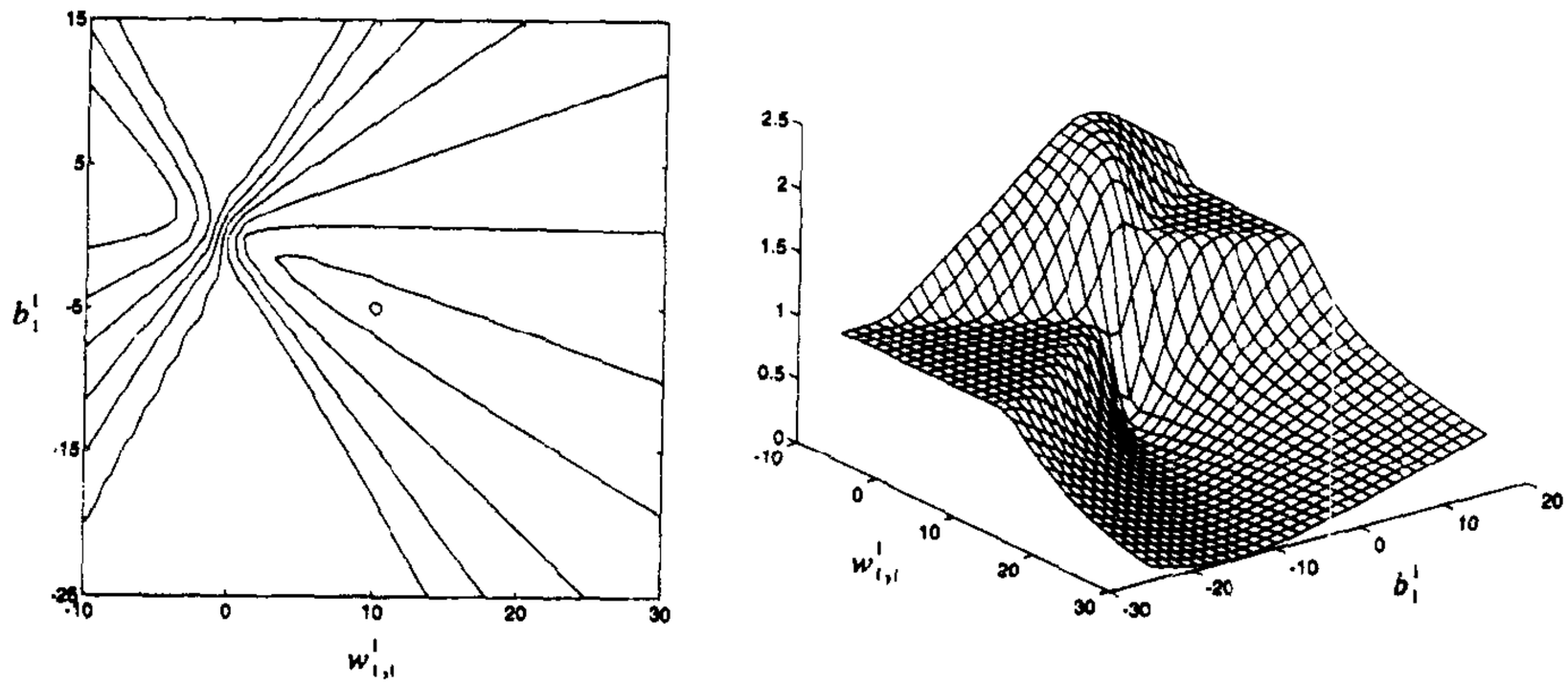



图 12-4 $w_{1,1}^1$和 $b_1^1$ 的平方误差曲面

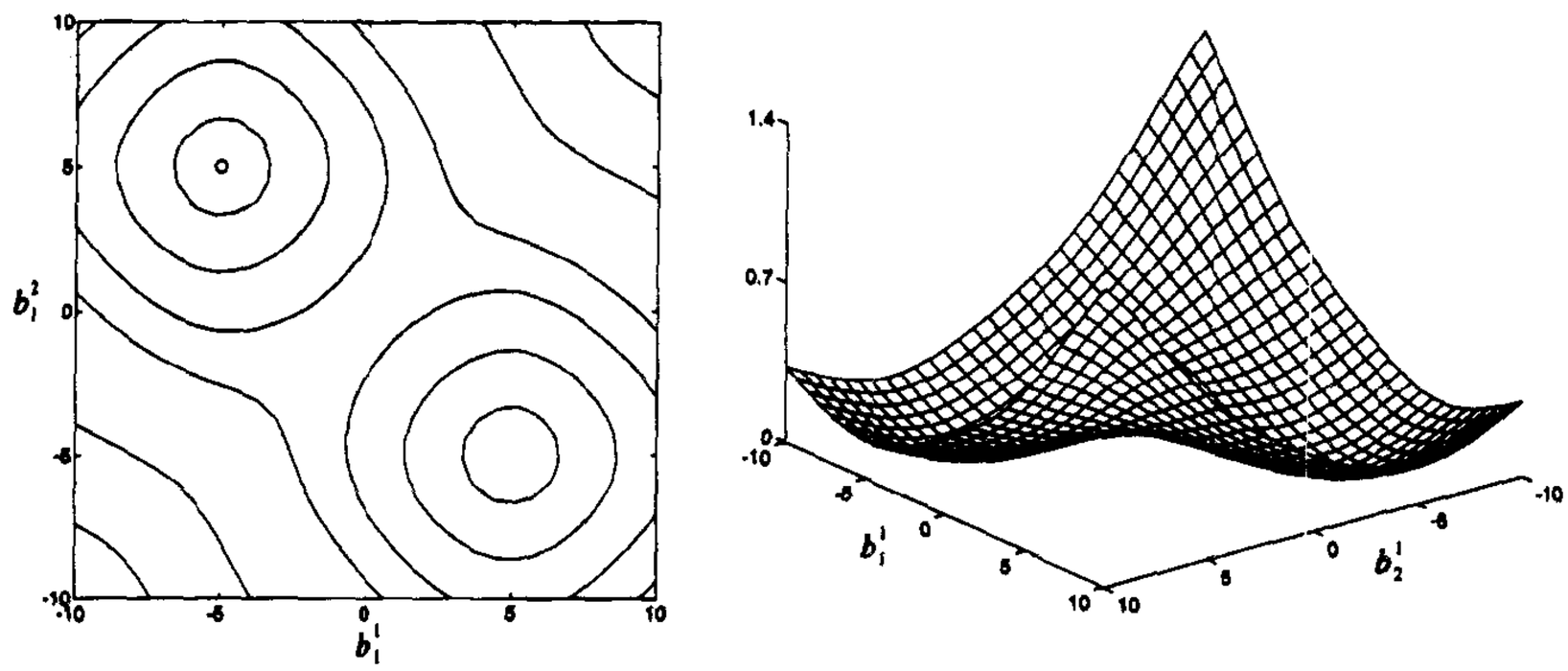



图 12-5 $b_1^1$ 和 $b_2^1$ 的平方误差曲面

该曲面显示了多层网络的一个重要特性：它们具有对称性。这里我们看到有两个局部极小点，它们都有相同的平方误差值。第二个解对应于相同网络的上下翻转(即将第一层神经元顶层的神经元与底层的神经元对换)。这是由于没有把初始权值和偏置值设成 0 的神经网络特征。对称性使 0 成为了性能曲面的一个鞍点。

对多层网络性能曲面的简单研究给出了一些如何设置 SDBP 算法初始参数的暗示。首先，不能把初始参数设置为 0。这是由于对性能曲面来说，参数空间的原点趋向鞍点。其次，不能把初始参数设置过大。这是由于在远离优化点的位置，性能曲面将变得十分平坦。

典型情况下，可以选择一些小的随机值作为初始权值和偏置值。这样我们可以在不离开性能曲面平坦区域的同时避开可能的鞍点。(另外一种选择初始值的方式在[NgWi90]中描述。)正如下节将看到的，可以选择多个不同的初始值以确保算法收敛到全局极小点。

12-6

**2. 收敛性举例**

**批处理** 前面已经研究过性能曲面，现在来看 SDBP 算法的性能。本节将使用一个称为批处理的标准算法的变种。在此方法中，当整个训练集都出现后网络参数才会更新。每个训练例子的梯度被平均在一起，以获得更精确的梯度估计。(如果训练集是完备的，即包括了所有可能的输入/输出对，则梯度的估计是精确的。)

图 12-6 中，我们可以看到 SDBP(批处理模式)算法在仅调整参数 $w_{1,1}^1$和 $w_{1,1}^2$时的两条轨迹。其中初始条件被标记为“a”的轨迹中，算法最终收敛到优化的解，但收敛的速度很慢。原因是曲面在轨迹路径上存在曲率变化。在经过初始化时的中等斜率后，轨迹通过一个非常平坦的曲面区域，直到它落入一个斜度很平缓的谷中。如果提高学习速度，算法将通过初始的平坦曲面而快速收敛。但正如稍后所见的，它在落入谷后将变得不稳定。

轨迹“b”显示了算法如何收敛到局部极小点。轨迹陷入一个谷中并且偏离了优化解。如果允许继续的话，轨迹将收敛到 $w_{1,1}^1 = 0.88$，$w_{1,1}^2 = 38.6$。多个局部极小点的存在是多层网络性能曲面的典型特征。因此，最好是选择多个初始点，以保证得到全局极小点。(如图 12-5 所示，一些局部极小点有相同的均方误差。所以，不能期望对每组初始值都能收敛到相同的参数值，只能期待获得同样的最小误差)。

12-7

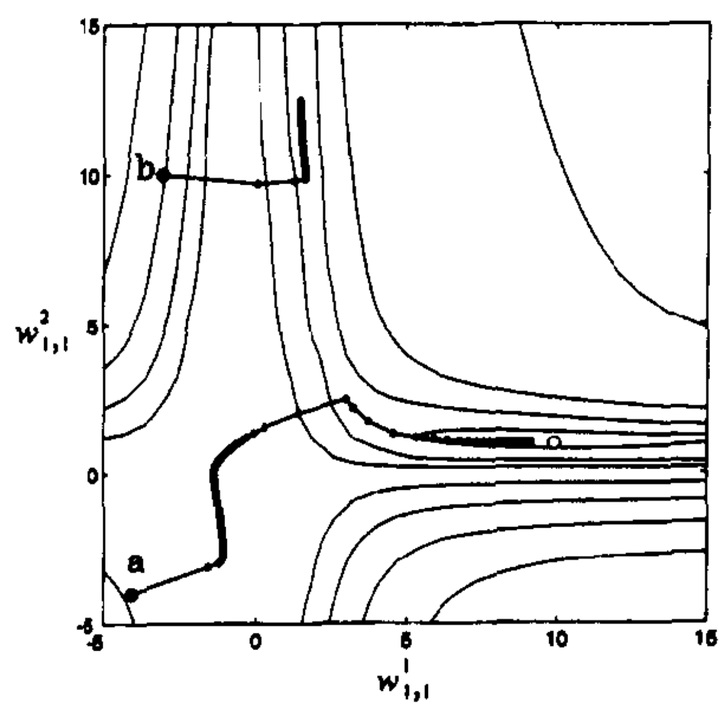



图 12-6　两个 SDBP 算法(批处理方式)的轨迹

算法的进展过程如图 12-7 所示。图中表示了迭代次及和均方误差的关系。左边的曲线对应于轨迹“a”右边的曲线对应于轨迹“b”。这些曲线是典型的 SDBP，具有长时间的慢进展和短时间的快进展。

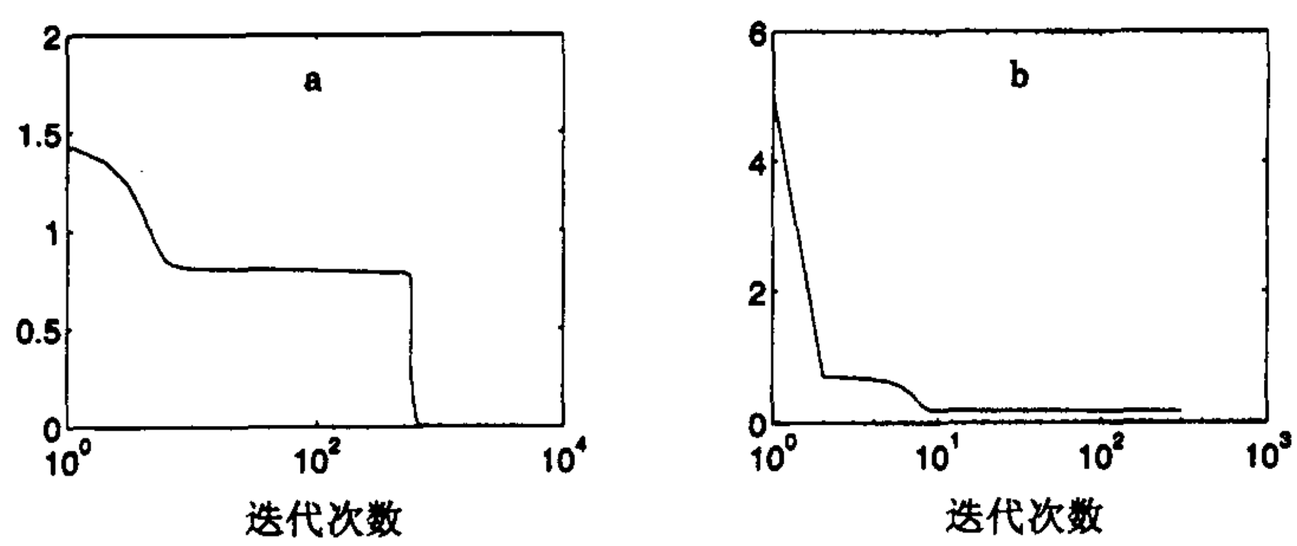



图 12-7　均方误差收敛模式

可以看出图 12-7 中的平坦区域对应于算法通过图 12-6 中性能曲面平坦区域的次数。在这些区域中应该增加学习速度从提高收敛速度。然而，如果在算法达到性能曲面中较陡峭的部分中时增加算法的学习速度将使它变得不稳定。

学习速度的影响在图 12-8 中表示。该轨迹对应于图 12-6 中的轨迹“a”，只是学习速度较

高。算法一开始收敛得很快，但是当轨迹到达包含极小点的窄谷时，算法开始发散。这意味
12-8 着改变学习速度是非常有效的。我们可以在平坦曲面时增加学习速度，在斜率增加时减少学习速度。问题是："算法怎么知道何时在平坦的曲面上呢?"我们将在稍后讨论这个问题。

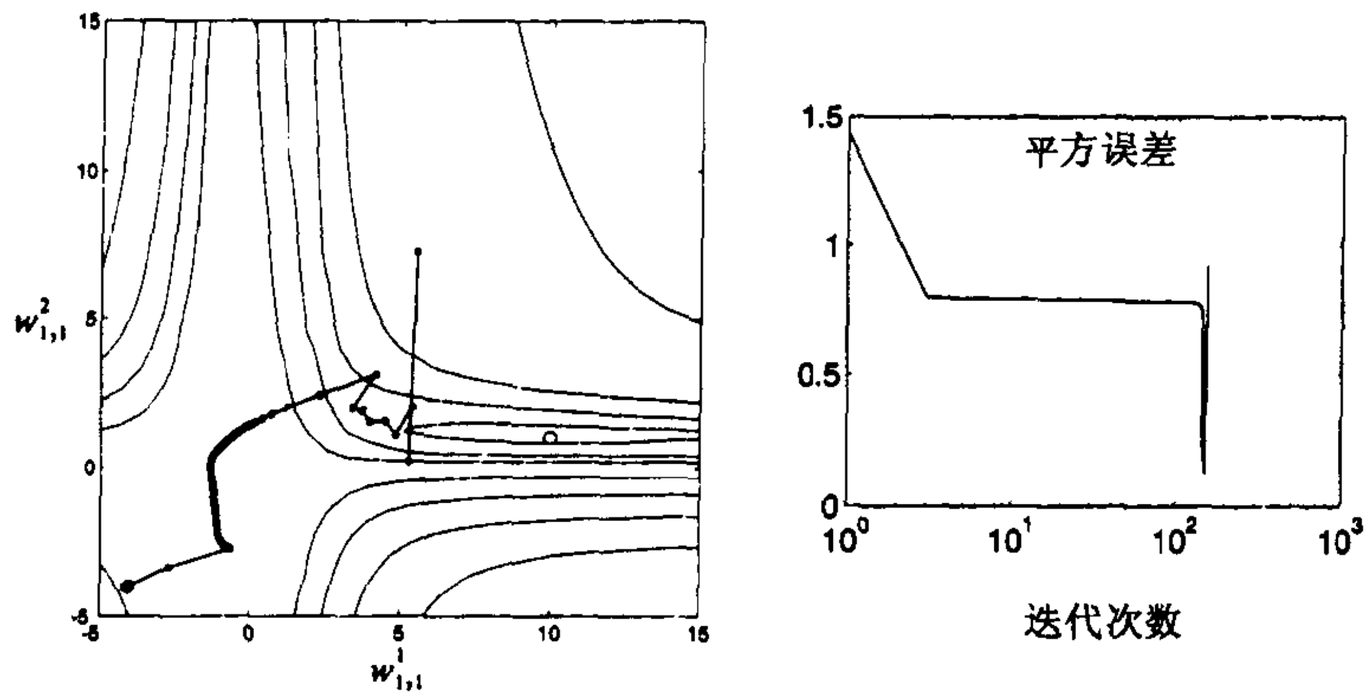



图 12-8 学习速度过大时的轨迹

提高收敛性的另一种方法是平滑轨迹。注意在图 12-8 中，当算法开始发散时，它在窄谷来回振荡。如果我们用平均改变参数的方法过滤轨迹，这样可以平滑掉振荡并产生一个稳定的轨迹。将在下节中讨论该过程。

*试验这个反向传播的例子请用 Neural Network Design Demonstration* ***Steepest Descent Backpropagation*** (**nnd12sd**)。

### 12.2.2 BP 算法的启发式改进

现在我们已经考察了反向传播(最速下降算法)的一些缺点，让我们考虑一些改进算法的方法。本节中，将讨论两种启发式方法。下节将提供两种基于标准数值优化算法的方法。

*1. 动量方法*

第一种方法是使用动量。这种改进是基于在上节中的观察：如果能平滑轨迹中的振荡将能提高收敛性能。可以用一个低通滤器来实现它。

在将动量应用于神经网络应用之前，首先考虑一个平滑效果的简单例子。下面是一阶滤
12-9 波器：

$$y(k) = \gamma y(k-1) + (1-\gamma)w(k) \tag{12.4}$$

其中 $w(k)$是滤波器输入，$y(k)$是滤波器输出，$\gamma$ 是动量系数，满足

$$0 \leqslant \gamma < 1 \tag{12.5}$$

滤波器的效果如图 12-9 所示。对这个例子滤波器输入取成正弦波：

$$w(k) = 1 + \sin\left(\frac{2\pi k}{16}\right) \tag{12.6}$$

左图中动量系数 $\gamma$ 被设为 0.9 而右图中 $\gamma$ 被设为 0.98。这里可以看到滤波器输出的振荡低于滤波器输入的振荡(正如我们对低通滤波器的希望)。另外，当 $\gamma$ 增加时，滤波器输出的振荡减少。也要注意平均滤波器输出与平均滤波器输入相同，虽然当 $\gamma$ 增加的时候，滤波器输出的反应变慢。

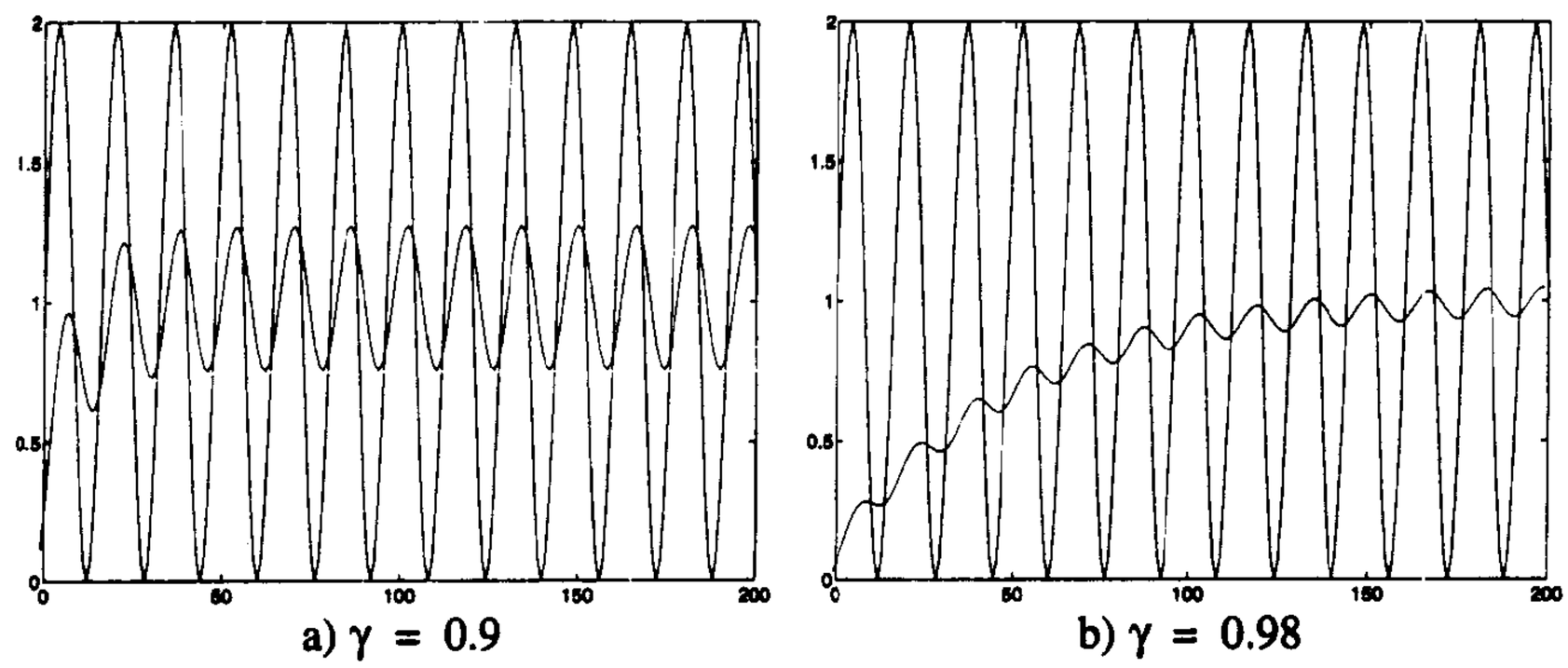



图 12-9　动量的平滑效应

总之，滤波器有助于减少振荡的数目，同时仍然保持平均值。现在，来看怎样把这种方法用于神经网络。首先，回忆 SDBP(式(11.46)和(11.47))的参数更新为

$$\Delta\mathbf{W}^m(k) = -\alpha\mathbf{s}^m(\mathbf{a}^{m-1})^T \tag{12.7}$$

$$\Delta\mathbf{b}^m(k) = -\alpha\mathbf{s}^m \tag{12.8}$$

12-10

**动量　MOBP**　当动量滤波器加到参数的改变后，得到了下述反向传播的动量改进(MOBP)公式：

$$\Delta\mathbf{W}^m(k) = \gamma\Delta\mathbf{W}^m(k-1) - (1-\gamma)\alpha\mathbf{s}^m(\mathbf{a}^{m-1})^T \tag{12.9}$$

$$\Delta\mathbf{b}^m(k) = \gamma\Delta\mathbf{b}^m(k-1) - (1-\gamma)\alpha\mathbf{s}^m \tag{12.10}$$

如果将这些改进后的公式用于上节的例子中，可以得到图 12-10 中所示的结果。(本例中使用了 MOBP 的批处理形式，即在整个训练集出现以后参数才被更新。每个训练样本的梯度计算之和被平均在一起以达到更精确的梯度估计。)这个轨迹对应于图 12-8 中同样的初始条件和学习速度，不同的是动量系数 $\gamma = 0.8$。可以看到算法现在是稳定的。由于使用了动量项，可以在维持算法稳定前提下使用更高的学习速度。动量的另一个特征是当轨迹进入某个一致的方向后，它可以加速收敛。

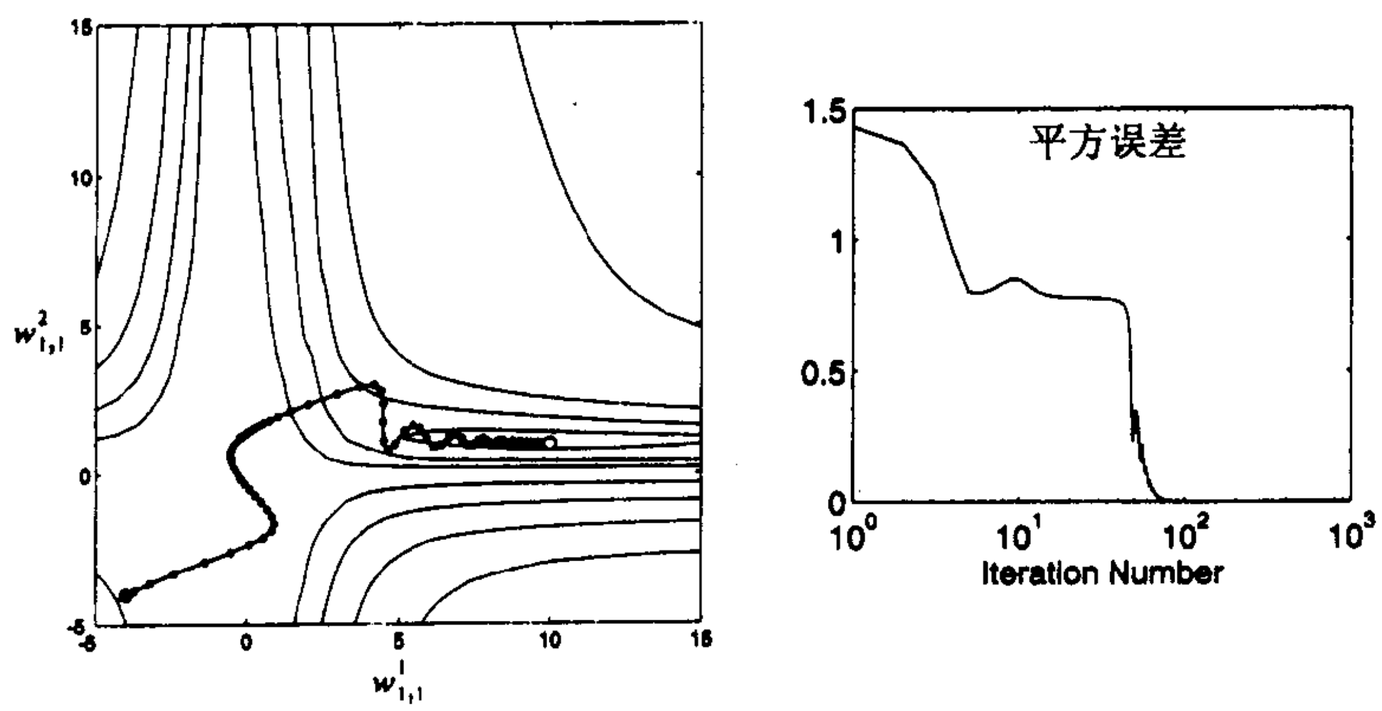



图 12-10　带动量时的轨迹

如果你仔细观察图 12-10，可以发现该方法为什么用动量这个词。它总是试图保持轨迹于同一方向。当 $\gamma$ 越大的时候，轨迹的“动量”越强。

12-11 试验动量请用 *Neural Network Design Demonstration Momentum Backpropagation* (**nnd12mo**)。

2. 可变的学习速度

在本章的前一部分曾建议为了提高收敛速度，可以在较平坦的曲面提高学习速度，而在斜率增大时减小学习速度。本节中将尝试这个方法。

前面曾指出单层线性网络的均方误差性能曲面总是一个二次函数，且赫森矩阵是常数的，最速下降算法的最大稳定学习速度是 2 除以赫森矩阵的最大特征值(见(9.25)式)。

正如我们所见的，多层网络的误差曲面不是二次函数。曲面的形状随参数空间区域的不同而不同。也许可以在学习过程中通过调整学习速度来提高收敛速度。技巧是决定何时改变学习速度和怎样改变学习速度。

**可变学习速度的 VLBP** 有许多不同的方法来改变学习速度。这里介绍一种非常直观的批处理过程[VoMa88]，它的学习速度是根据算法的性能改变的。可变学习速度反向传播算法(variable learing rate backpropagation，VLBP)的规则如下：

1) 如果均方误差(在整个训练集上)权值在更新后增加了，且超过了某个设置的百分数 $\zeta$ (典型值为 1% 至 5%)，则权值更新被取消，学习速度被乘以一个因子 $\rho(0<\rho<1)$，并且动量系数 $\gamma$(如果有的话)被设置为 0。
2) 如果平方误差在权值更新后减少，则权值更新被接受，而且学习速度将被乘以一个因子 $\eta>1$。如果 $\gamma$ 被设置为 0，则恢复到以前的值。
3) 如果平方误差的增长小于 $\zeta$，则权值更新被接受，但学习速度保持不变。如果 $\gamma$ 过去被设置为 0，则恢复到以前的值。

(关于 VLBP 的数值例子请见例题 P12.3。)

为了说明 VLBP，让我们将它应用于前节的函数逼近问题中。图 12-11 显示了算法的轨迹，其中的初始条件、初始学习速度和动量系数与图 12-10 所用的相同。新参数的设置为：

12-12

$$\eta = 1.05, \quad \rho = 0.7, \quad \zeta = 4\% \tag{12.11}$$

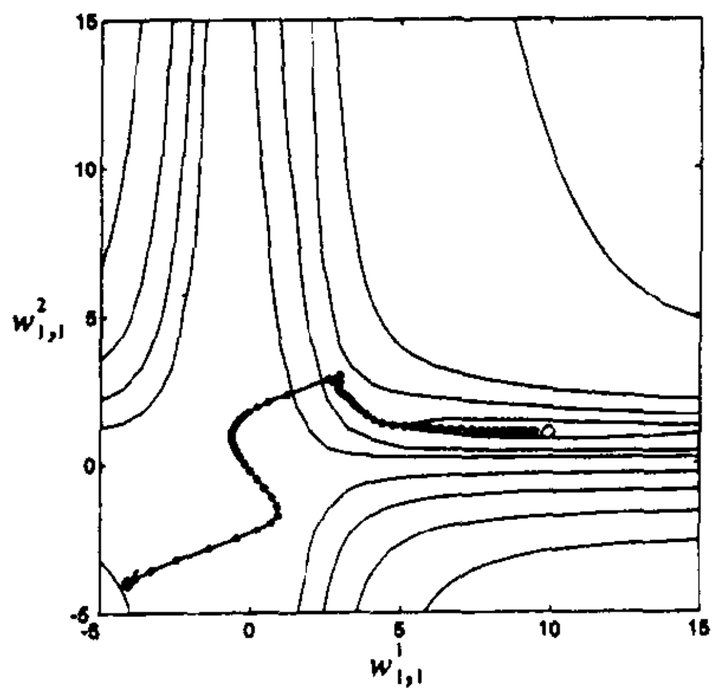



图 12-11 可变学习速度的轨迹

注意学习速度和步长在轨迹通过按常数递减误差的直线时如何保持增加。这个效应在图 12-12 中也可以看出。图 12-12 指出了平方误差和学习速度与迭代次数之间的关系。

当轨迹进入一个窄谷时，学习速度迅速递减。否则轨迹将产生振荡，并使误差迅速增加。在每步增加超过 4% 的误差当中，学习速度减少而且动量项被消除，这样使轨迹可以迅速转向沿窄谷到极小点的方向。接着学习速度继续增加以加速收敛。当轨迹超过极小点时学习速度再次下降，此时算法已经基本收敛。这个过程是典型的 VLBP 轨迹。

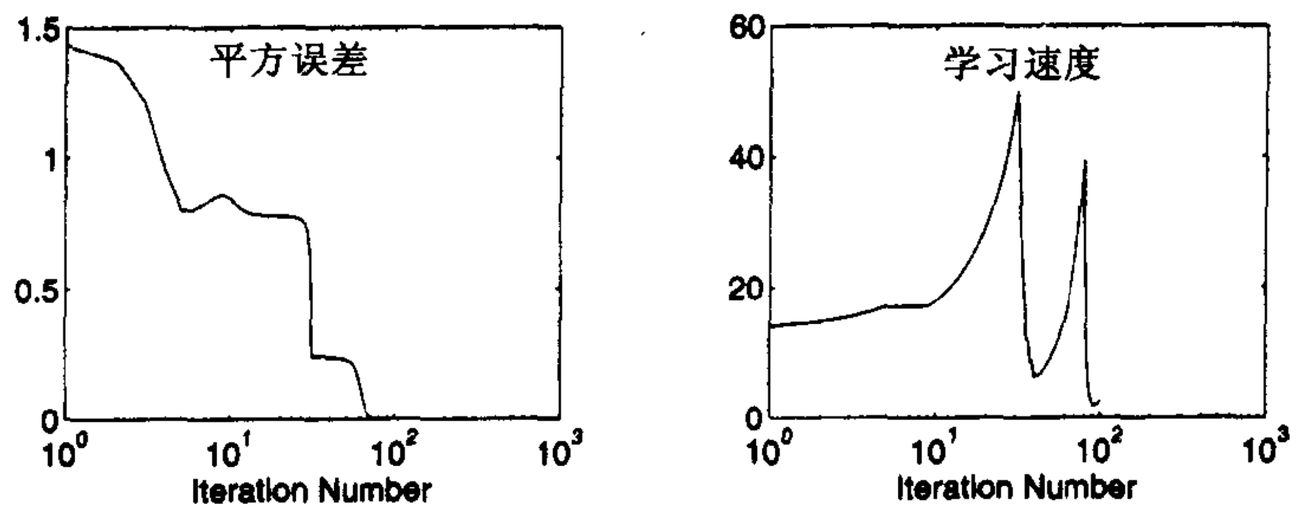
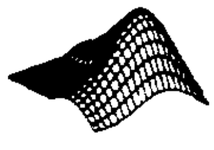



图 12-12　VLBP 的收敛特征

有许多关于可变学习速度算法的变型。Jacobs[Jaco88]提出了 delta-bar-delta 学习规则，其中每一个网络参数(权值和偏置值)都有自己的学习速度。如果某个参数在几次迭代中都沿同一方向变化，算法则增加网络参数的学习速度。如果参数的改变方向发生变化，则学习速度递减。Tollenaere[Toll90]的 SuperSAB 算法与 delta-bar-delta 规则类似，但它在改变学习速度的规则方面更加复杂。 12-13

另一种对 SDBP 的启发式变型是 Fahlman[Fahl 88]的 Quickprop 算法。它假设误差曲面是抛物面且在极小点附近是向上凹的，另外每个参数的影响被认为是相互独立的(参见 19 章给出的其他 SDBP 改进方法)。

对 SDBP 进行启发式改进对某些问题会提高收敛速度。但这些方法有两个主要缺点：首先这些改进需要设置一些参数(例如 $\zeta$，$\rho$ 和 $\gamma$)，而 SDBP 只需要一个学习速度参数。某些更复杂的启发式改进需要设置五六个参数。算法的性能对这些参数的改变往往十分敏感。参数的选择还是问题相关的。这些对 SDBP 的改进的第二个缺点是它们对某些 SDBP 最终能找到解的问题却不能收敛。应用越复杂的算法这些问题越容易发生。



*实验 VLBP 请用 Neural Network Design Demonstration Variable Learning Rate Backpropagation* (**nnd12vl**)。

### 12.2.3　数值优化技术

我们已经研究了一些用启发式方法改进 SDBP 的算法，现在来考虑一些基于标准数值优化技术的方法。这里将考察两种技术：共轭梯度法和 Levenberg-Marquardt 方法。二次函数的共轭梯度法在第 9 章中已经介绍过了。我们要为该算法增加两个过程以适应更一般的函数。

本章中讨论的第二种数值优化方法是 Levenberg-Marquardt 算法，它是牛顿法的一个改进并且非常适合于神经网络训练。

1. 共轭梯度法

**CGBP** 第 9 章中介绍了 3 种数值优化技术：最速下降法、共轭梯度法和牛顿法。最速下降法是最简单的算法，但收敛较慢。牛顿法要快得多，但是需要计算赫森矩阵和它的逆。共轭梯度法是某种折衷：它不需要计算二次导数，但仍然具有二次收敛的特性(它在有限次迭代后能收敛于到二次函数的极小点)。本节将介绍怎样将共轭梯度法应用于训练多层网络。我们称这种方法为共轭梯度反向传播算法(conjugate gradient backpropagation，CGBP)。

12-14

让我们首先重温共轭梯度法。为便于引用起见，我们将重复第 9 章的算法步骤。

1）选择初始搜索方向 $\mathbf{p}_0$ 为梯度的反向量，如式(9.59)：

$$\mathbf{p}_0 = -\mathbf{g}_0 \tag{12.12}$$

其中

$$\mathbf{g}_k \equiv \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k} \tag{12.13}$$

2）根据式(9.57)取一步，选择学习速度 $\alpha_k$，沿搜索方向最小化函数：

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \tag{12.14}$$

3）根据式(9.60)选择下一个搜索方向，利用式(9.61)，(9.62)和(9.63)计算 $\beta_k$：

$$\mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1} \tag{12.15}$$

其中

$$\beta_k = \frac{\Delta\mathbf{g}_{k-1}^T\mathbf{g}_k}{\Delta\mathbf{g}_{k-1}^T\mathbf{p}_{k-1}} \text{ 或 } \beta_k = \frac{\mathbf{g}_k^T\mathbf{g}_k}{\mathbf{g}_{k-1}^T\mathbf{g}_{k-1}} \text{ 或 } \beta_k = \frac{\Delta\mathbf{g}_{k-1}^T\mathbf{g}_k}{\mathbf{g}_{k-1}^T\mathbf{g}_{k-1}} \tag{12.16}$$

4）如果算法不收敛，继续第 2 步。

这样的共扼梯度算法不能直接应用于神经网络训练，因为性能指数不是二次的。这在两个方面影响算法。首先，不能用式(9.31)沿直线最小化函数(这是第 2 步所需的)。其次通常不能在有限步内得到精确的最小值，因此算法在迭代过若干次之后需要重新设置。

首先来看线性搜索。需要一个一般的过程去确定函数在某个特定方向的极值。这包括两步：区间定位和区间缩小。区间定位步的目的是找某个包含局部极小点的初始区间。区间缩小步接着将缩小初始区间直到满足一定精度的极小点被定位。

12-15

**区间定位** 我们使用一种函数比较方法[Scal85]去处理区间定位，这一步如图 12-3 所示。一开始计算某个初始点的性能指数，由图中 $a_1$ 表示。该点表示网络权值和偏置值的当前值。按句话说，我们是在计算

$$F(\mathbf{x}_0) \tag{12.17}$$

下一步是计算第二点的函数值，由图中 $b_1$ 点表示，它距初始点距离是 $\varepsilon$ 且沿初始搜索方向 $\mathbf{p}_0$。换句话说，我们是在计算

$$F(\mathbf{x}_0 + \varepsilon\mathbf{p}_0) \tag{12.18}$$

继续计算新点 $b_i$ 的性能指数(点之间的距离依次增加一倍)。这一过程直到连续两次计算的函数值增加时结束。这在图 12-13 中用 $b_3$，$b_4$ 表示。此时可以知道极小点是在 $a_5$ 和 $b_5$ 之间。不能将区间缩得更小，这是因为极小值可能在[$a_4$，$b_4$]之间或在[$a_3$，$b_3$]之间。这两种可能的情况如图 12-14(a)所示。

**区间缩小** 现在已经定位了包含极小点的区间，线性搜索的下一步是区间缩小。它将包括计算区间[$a_5$，$b_5$]内点的函数值，[$a_5$，$b_5$]是由区间定位步得到的。从图 12-14 中可以看

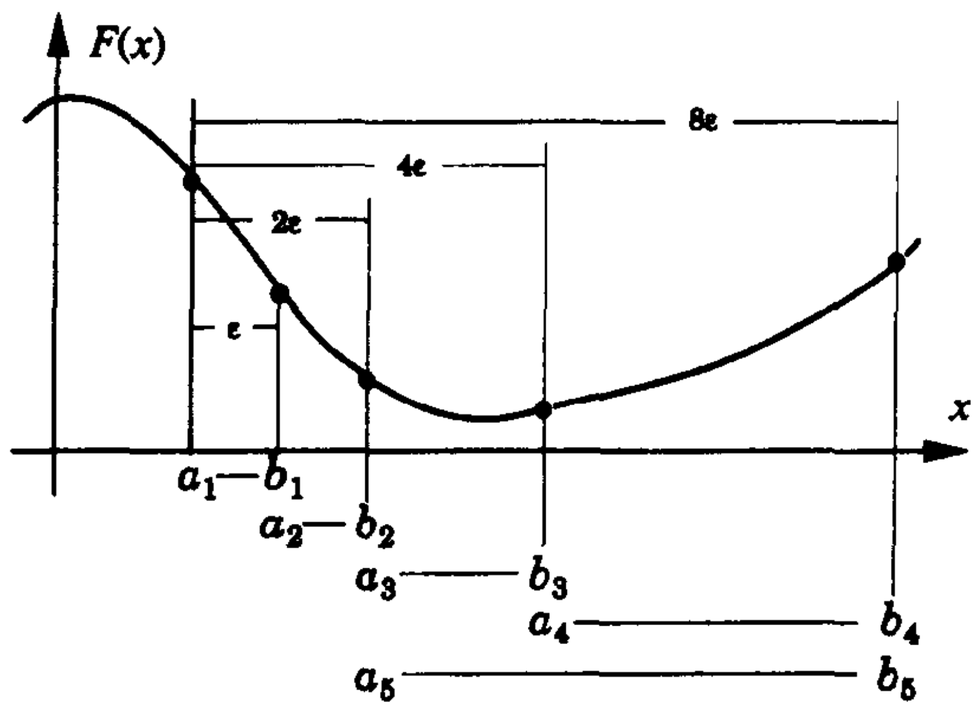
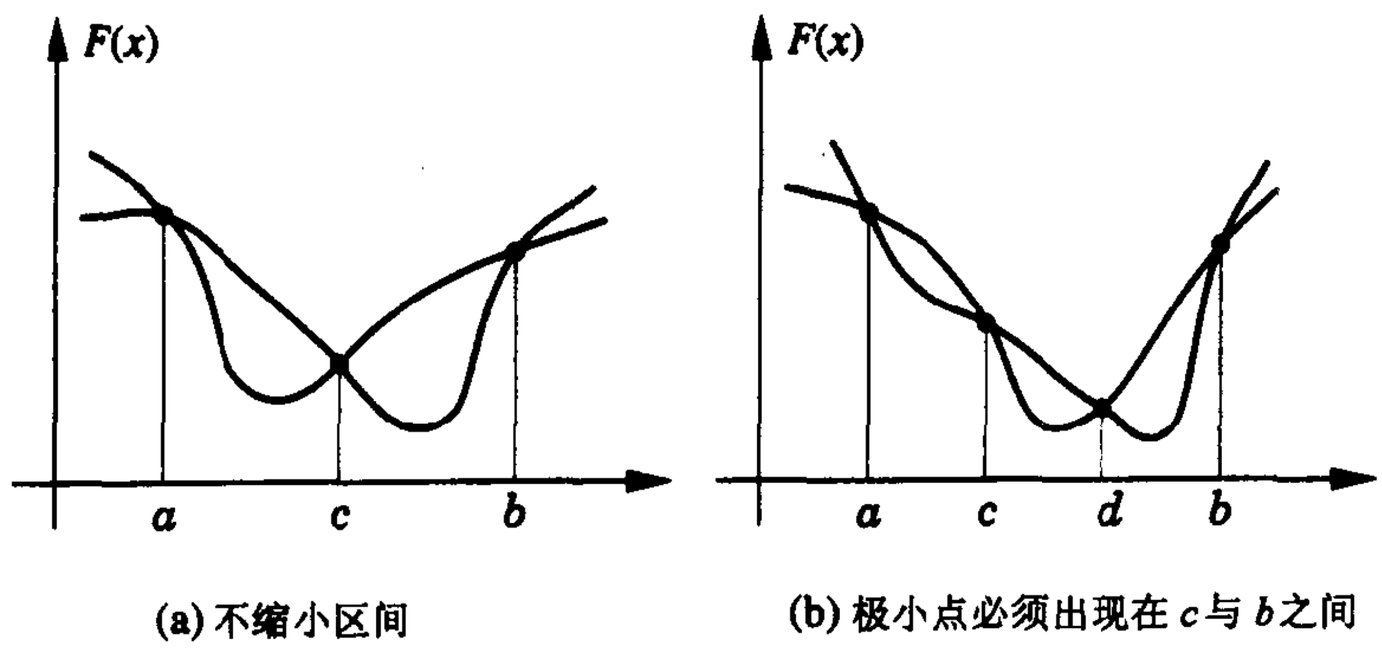



图 12-13　区间定位

出至少必须计算两个内部点的函数值以减少不确定区间的尺寸。图 12-14(a)指出一个区间的函数值计算不能提供极小点定位的任何信息。但是，如果计算了两个点 $c$ 和 $d$(如图 12-14(b))，可以缩小不确定的区间。如果 $F(c) > F(d)$(如图 12-14(b)所示)，则极小点必定在 12-16
$[c, b]$区间内。反之，如果 $F(c) < F(d)$，则极小点在$[a, d]$区间内。(注意，假设在初始区间中只有一个极小点。后面要作进一步讨论。)




(a) 不缩小区间　　(b) 极小点必须出现在 $c$ 与 $b$ 之间

图 12-14　减少不确定区间的大小

**黄金分割搜索**　上述过程描述了减少不确定区间尺寸的方法。现在需要确定如何找到内部点 $c$ 点 $d$ 位置的方法。有一些方法能实现它(见[Scal85])。我们使用一种称为*黄金分割搜索*的方法，它可以减少函数计算的次数。每次迭代只需要计算一次函数值。例如，在图 12-14(b)的例子中，点 $a$ 可以丢弃而点 $c$ 成为外部点。于是一个新的点 $c$ 将在原来的点 $c$ 和点 $d$ 之间。技巧是放置新的点以便尽快减少不确定性区间。

黄金分割搜索算法如下所示[Scal85]：

$\tau = 0.618$

set　　$c_1 = a_1 + (1-\tau)(b_1 - a_1)$，$F_c = F(c_1)$

　　$d_1 = b_1 - (1-\tau)(b_1 - a_1)$，$F_d = F(d_1)$

for $k = 1, 2, \ldots$ repeat

　　If $F_c < F_d$ then

set $a_{k+1} = a_k$; $b_{k+1} = d_k$; $d_{k+1} = c_k$

$c_{k+1} = a_{k+1} + (1-\tau)(b_{k+1} - a_{k+1})$

$F_d = F_c$; $F_c = F(c_{k+1})$

12-17 else

set $a_{k+1} = c_k$; $b_{k+1} = b_k$; $c_{k+1} = d_k$

$d_{k+1} = b_{k+1} - (1-\tau)(b_{k+1} - a_{k+1})$

$F_c = F_d$; $F_d = F(d_{k+1})$

end

end until $b_{k+1} - a_{k+1} < tol$

其中 $tol$ 是用户给定的精度上限。(关于区间定位和区间缩小过程的数值例子请见例题 P12.4。)

要使共轭梯度法应用于神经网络训练，有若干地方需要改进。对二次函数算法将至多在 $n$ 次迭代 内收敛到极小点，其中 $n$ 是被优化的参数数目。由于多层网络的平方误差性能指数不是二次函数，所以算法一般不能在 $n$ 次迭代内收敛。共轭梯度法的发展并不意味着在同一搜索方向下包含 $n$ 次迭代过程的一个周期就可以结束。这可能有多个过程，但最简单的方法是在 $n$ 次迭代之后将搜索方向重新设置为最速的下降方向[Scal 85]。我们将使用这一方法。

让我们现在把共轭梯度法应用于解释其他神经网络训练算法的函数逼近例子中。我们将用反向传播算法计算梯度(用式(11.23)和(11.24))，并用共扼梯度法决定权值的更新。这里仍采用批处理算法，即梯度是在整个训练集都应用于网络后才计算的。

图 12-15 显示了 CGBP 算法在三次迭代后的中间步。区间定位过程由小空心圆圈表示；每一个点表示一次函数的计算，最终区间由较大的空心圆圈表示，图 12-15 中的小黑点表示用黄金分割搜索的新的内部点，每一个对应于一次迭代过程。最终点是由大黑点表示。

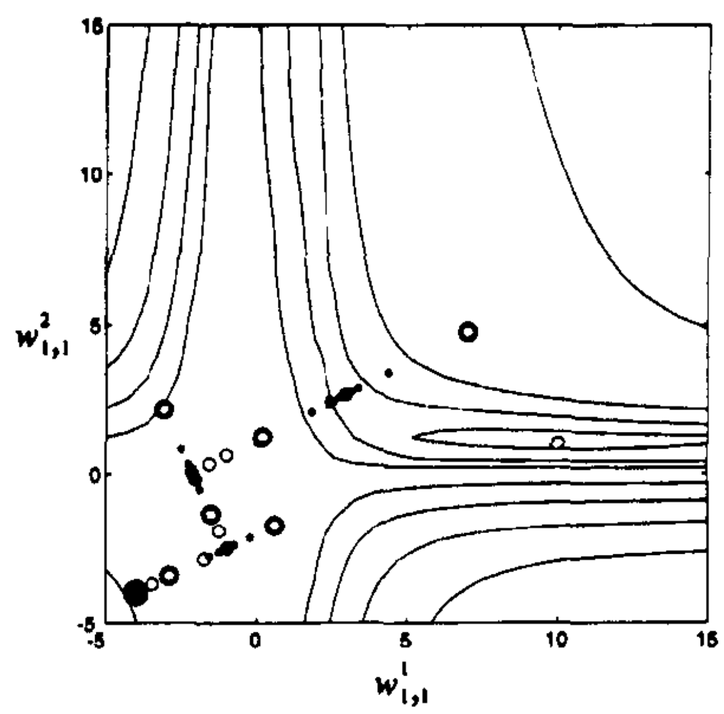



图 12-15 CGBP 的中间步骤

图 12-16 表示收敛的整个轨迹。注意，CGBP 算法要比我们试过的所有其他算法都少的迭代次数收敛。这有一点欺骗性，因为 CGBP 的每次迭代较其他方法需要更多的计算；在 CGBP 的每次迭代中包括了多次函数计算。即使如此，CBGP 算法也是多层网络批处理训练

12-18 算法中最快速的方法之一[Char92]。

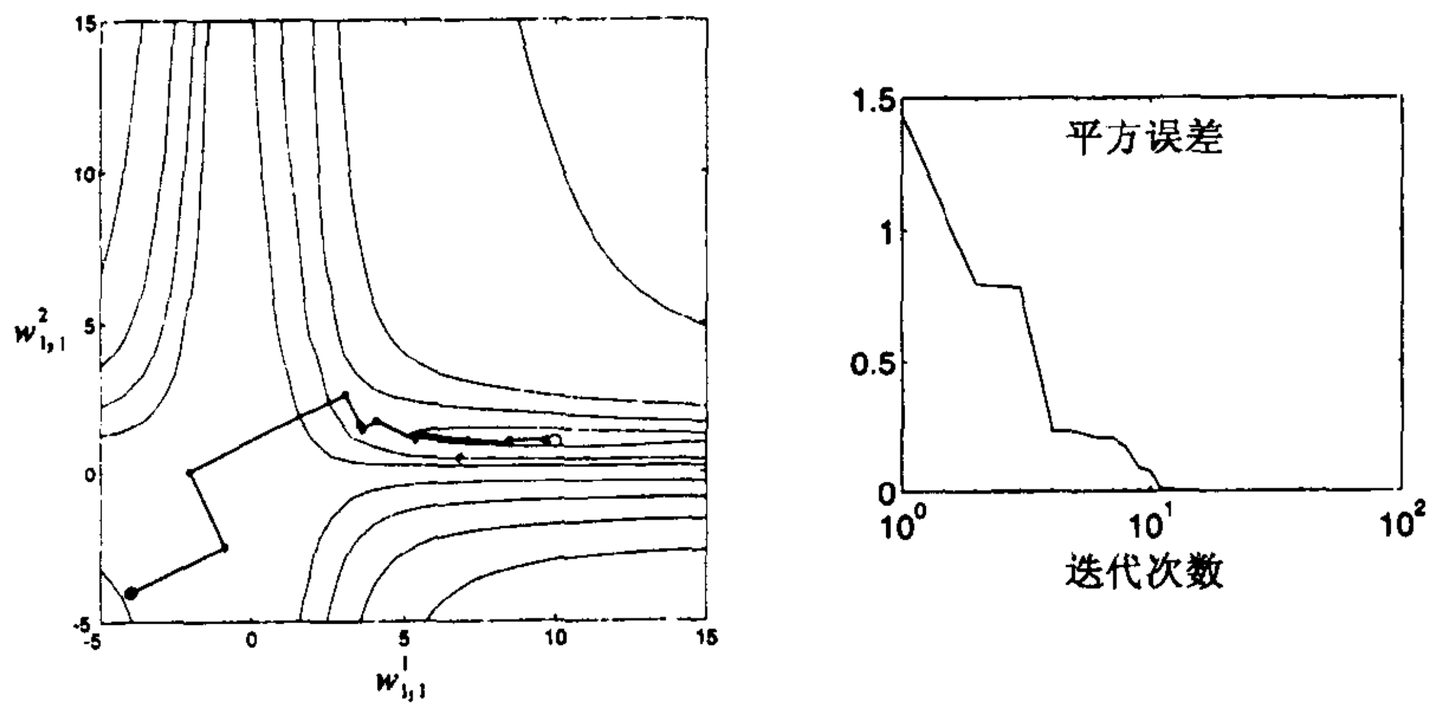
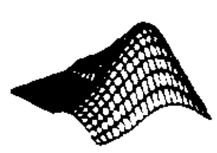



图 12-16 共轭梯度的轨迹



试验 CGBP 请用 *Neural Network Design Demonstration Conjugate Gradient Line Search*(**nnd12ls**)和 *Conjugate Gradient Backpropagation*(**nnd12cg**)。

2. Levenberg-Marquardt 算法

Levenberg-Marquardt 算法是牛顿法的变形，用以最小化那些作为其他非线性函数平方和的函数。这非常适合于性能指数是均方误差的神经网络训练。

**基本算法**

让我们从考虑性能指数是一组平方和的牛顿方法的形式开始。由第 9 章知，优化性能指数 $F(\mathbf{x})$的牛顿方法是 12-19

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1}\mathbf{g}_k \tag{12.19}$$

其中 $\mathbf{A}_k \equiv \nabla^2 F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k}$，$\mathbf{g}_k \equiv \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k}$。

**雅可比矩阵** 如果假设 $F(\mathbf{x})$是平方函数之和，即

$$F(\mathbf{x}) = \sum_{i=1}^{N} v_i^2(\mathbf{x}) = \mathbf{v}^T(\mathbf{x})\mathbf{v}(\mathbf{x}) \tag{12.20}$$

那么第 $j$ 个梯度分量为

$$[\nabla F(\mathbf{x})]_j = \frac{\partial F(\mathbf{x})}{\partial x_j} = 2\sum_{i=1}^{N} v_i(\mathbf{x})\frac{\partial v_i(\mathbf{x})}{\partial x_j} \tag{12.21}$$

因此梯度可以写成矩阵形式：

$$\nabla F(\mathbf{x}) = 2\mathbf{J}^T(\mathbf{x})\mathbf{v}(\mathbf{x}) \tag{12.22}$$

其中

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial v_1(\mathbf{x})}{\partial x_1} & \frac{\partial v_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial v_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial v_2(\mathbf{x})}{\partial x_1} & \frac{\partial v_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial v_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial v_N(\mathbf{x})}{\partial x_1} & \frac{\partial v_N(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial v_N(\mathbf{x})}{\partial x_n} \end{bmatrix} \tag{12.23}$$

J-x 为雅可比矩阵。

下一步计算赫森矩阵。赫森矩阵的第 $k$，$j$ 元素为

$$[\Delta^2 F(\mathbf{x})]_{k,j} = \frac{\partial^2 F(\mathbf{x})}{\partial x_k \partial x_j} = 2\sum_{i=1}^{N}\left\{\frac{\partial v_i(\mathbf{x})}{\partial x_k}\frac{\partial v_i(\mathbf{x})}{\partial x_j} + v_i(\mathbf{x})\frac{\partial^2 v_i(\mathbf{x})}{\partial x_k \partial x_j}\right\} \tag{12.24}$$

赫森矩阵于是可以表示为

12-20

$$\nabla^2 F(\mathbf{x}) = 2\mathbf{J}^T(\mathbf{x})\mathbf{J}(\mathbf{x}) + 2\mathbf{S}(\mathbf{x}) \tag{12.25}$$

其中

$$\mathbf{S}(\mathbf{x}) = \sum_{i=1}^{N} v_i(\mathbf{x})\nabla^2 v_i(\mathbf{x}) \tag{12.26}$$

如果假设 $\mathbf{S}(\mathbf{x})$很小，可以将赫森矩阵近似表示为

$$\nabla^2 F(\mathbf{x}) \cong 2\mathbf{J}^T(\mathbf{x})\mathbf{J}(\mathbf{x}) \tag{12.27}$$

**高斯-牛顿方法** 将(12.27)式和(12.22)式代入(12.19)式，可以得到高斯-牛顿方法：

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{x}_k - [2\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k)]^{-1}2\mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k)\\ &= \mathbf{x}_k - [\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k)]^{-1}\mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k)\end{aligned} \tag{12.28}$$

注意高斯-牛顿方法较标准牛顿法的优点是不需计算二阶导数。

高斯-牛顿方法的一个问题是矩阵 $\mathbf{H} = \mathbf{J}^T\mathbf{J}$ 可能不可逆。这可以用下述近似赫森矩阵改进：

$$\mathbf{G} = \mathbf{H} + \mu\mathbf{I} \tag{12.29}$$

为看出这个矩阵是可逆的，设 $\mathbf{H}$ 的特征值和特征向量为$\{\lambda_1, \lambda_2, \cdots, \lambda_n\}$和$\{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n\}$，则有

$$\mathbf{G}\mathbf{z}_i = [\mathbf{H} + \mu\mathbf{I}]\mathbf{z}_i = \mathbf{H}\mathbf{z}_i + \mu\mathbf{z}_i = \lambda_i\mathbf{z}_i + \mu\mathbf{z}_i = (\lambda_i + \mu)\mathbf{z}_i \tag{12.30}$$

因此 $\mathbf{G}$ 的特征向量与 $\mathbf{H}$ 的特征向量相同，且 $\mathbf{G}$ 的特征值为 $\lambda_i + \mu$。对所有 $i$，增加 $\mu$ 以保证 $\lambda_i + \mu > 0$，可使 $\mathbf{G}$ 成为正定的，所以矩阵可逆。

**Levenberg-Marquardt 算法** 由此可导出 Levenberg-Marquardt 算法[Scal85]：

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k) + \mu_k\mathbf{I}]^{-1}\mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k) \tag{12.31}$$

或

$$\Delta\mathbf{x}_k = -[\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k) + \mu_k\mathbf{I}]^{-1}\mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k) \tag{12.32}$$

这个算法的一个非常有用的特点是：当 $\mu_k$ 增加时，它接近于有小的学习速度的最速下

12-21 降算法：

$$\mathbf{x}_{k+1} \cong \mathbf{x}_k - \frac{1}{\mu_k}\mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k) = \mathbf{x}_k - \frac{1}{2\mu_k}\nabla F(\mathbf{x}),\text{对于大的 } \mu_k \tag{12.33}$$

当 $\mu_k$ 下降到 0 的时候，算法变成了高斯-牛顿方法。

算法开始时 $\mu_k$ 取小值(例如 $\mu_k = 0.01$)。如果某一步不能减少 $F(\mathbf{x})$值，则将 $\mu_k$ 乘以一个因子 $\theta > 1$(例如 $\theta = 1.0$)后再重复这一步。最后 $F(\mathbf{x})$会下降，因为使用最速下降方向的一小步。如果某一步产生了更小的 $F(\mathbf{x})$，则 $\mu_k$ 在下一步被除以 $\theta$，这样算法就接近于高斯-牛顿方法，该方法能提高收敛速度。这个算法提供牛顿法的速度和保证收敛的最速下降法之间的一个折衷。

现在来看如何将Levenberg-Marquardt算法应用于多层网络训练问题。多层网络训练的

性能指数是均方误差(见式(11.11))。如果每一个目标以相同的概率出现，均方误差就正比于训练集中下述所有 $Q$ 个目标的平方误差之和：

$$F(\mathbf{x}) = \sum_{q=1}^{Q}(\mathbf{t}_q - \mathbf{a}_q)^T(\mathbf{t}_q - \mathbf{a}_q)$$
$$= \sum_{q=1}^{Q}\mathbf{e}_q^T\mathbf{e}_q = \sum_{q=1}^{Q}\sum_{j=1}^{S^M}(\mathbf{e}_{j,q})^2 = \sum_{i=1}^{N}(v_i)^2 \tag{12.34}$$

其中 $e_{j,q}$是第 $q$ 个输入/目标对的误差的第 $j$ 项元素。

式(12.34)等价于性能指数式(12.20)(Levenberg-Marquardt 方法)。所以，为网络训练调整算法将是很直观的。结果表明这在概念上是正确的，但在细节上要加以注意。

**雅可比计算**

Levenberg-Marquardt 算法中的关键一步是雅可比矩阵的计算。用一种 BP 算法的变形来进行计算。回忆在标准 BP 算法中，以网络的权值和偏置值计算平方误差的导数。为了产生雅可比矩阵，需要用误差的导数来代替平方误差的导数。

从概念上说，修改 BP 算法以计算雅可比矩阵的元素是很容易的。但是，虽然概念上很简单，实现上却需要一些技巧。因此，在第一次阅读时你可以先跳过本节的其余部分以获得算法流程的总体概念，而后再返回来看细节。在继续看下去之前，先复习第 11 章中 BP 算法的推导是有益的。 12-22

在介绍计算雅可比阵的过程之前，先仔细观察它的形式(式(12.23))。注意误差向量为

$$\mathbf{v}^T = [v_1\ v_2 \cdots\ v_N] = [e_{1,1}\ e_{2,1} \cdots\ e_{S^M,1}\ e_{1,2} \cdots\ e_{S^M,Q}] \tag{12.35}$$

参数向量为

$$\mathbf{x}^T = [x_1\ x_2 \cdots\ x_n] = [w_{1,1}^1\ w_{1,2}^1 \cdots\ w_{S^1,R}^1\ b_1^1 \cdots\ b_{S^1}^1\ w_{1,1}^2 \cdots\ b_{S^M}^M] \tag{12.36}$$

其中 $N = Q \times S^M$，$n = S^1(R+1) + S^2(S^1+1) + \cdots + S^M(S^{M-1}+1)$。

因此，可以把这些式子代入式(12.23)中，多层网络训练的雅可比矩阵可以写为

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial w_{1,1}^1} & \frac{\partial e_{1,1}}{\partial w_{1,2}^1} & \cdots & \frac{\partial e_{1,1}}{\partial w_{S^1,R}^1} & \frac{\partial e_{1,1}}{\partial b_1^1} & \cdots \\ \frac{\partial e_{2,1}}{\partial w_{1,1}^1} & \frac{\partial e_{2,1}}{\partial w_{1,2}^1} & \cdots & \frac{\partial e_{2,1}}{\partial w_{S^1,R}^1} & \frac{\partial e_{2,1}}{\partial b_1^1} & \cdots \\ \vdots & \vdots & & \vdots & \vdots & \\ \frac{\partial e_{S^M,1}}{\partial w_{1,1}^1} & \frac{\partial e_{S^M,1}}{\partial w_{1,2}^1} & \cdots & \frac{\partial e_{S^M,1}}{\partial w_{S^1,R}^1} & \frac{\partial e_{S^M,1}}{\partial b_1^1} & \cdots \\ \frac{\partial e_{1,2}}{\partial w_{1,1}^1} & \frac{\partial e_{1,2}}{\partial w_{1,2}^1} & \cdots & \frac{\partial e_{1,2}}{\partial w_{S^1,R}^1} & \frac{\partial e_{1,2}}{\partial b_1^1} & \cdots \\ \vdots & \vdots & & \vdots & \vdots & \end{bmatrix} \tag{12.37}$$

雅可比矩阵中的元素，可以由 BP 算法的简单改进计算。标准 BP 算法的计算公式为：

$$\frac{\partial \hat{F}(\mathbf{x})}{\partial x_l} = \frac{\partial \mathbf{e}_q^T\mathbf{e}_q}{\partial x_l} \tag{12.38}$$ 12-23

对 Levenberg-Marquardt 算法中所需的雅可比矩阵的元素，我们需要计算如下的项：

$$[\mathbf{J}]_{h,l} = \frac{\partial v_h}{\partial x_l} = \frac{\partial e_{k,q}}{\partial x_l} \tag{12.39}$$

回忆 BP 算法中的导数(式 11.18)

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial w_{i,j}^m} \tag{12.40}$$

上式右边第一项被定义为敏感度

$$s_i^m \equiv \frac{\partial \hat{F}}{\partial n_i^m} \tag{12.41}$$

**Marquardt 敏感度** BP 算法中用递归关系从最后一层回到第一层计算敏感度。可以用同样的概念计算雅可比矩阵的各项(式(12.37))，如果定义新的 Marquardt 敏感度：

$$\tilde{s}_{i,h}^m \equiv \frac{\partial v_h}{\partial n_{i,q}^m} = \frac{\partial e_{k,q}}{\partial n_{i,q}^m} \tag{12.42}$$

其中，由式(12.35)，$h = (q-1)S^M + k$。

可以用下式计算雅可比矩阵的元素：

$$[\mathbf{J}]_{h,l} = \frac{\partial v_h}{\partial x_l} = \frac{\partial e_{k,q}}{\partial w_{i,j}^m} = \frac{\partial e_{k,q}}{\partial n_{i,q}^m} \times \frac{\partial n_{i,q}^m}{\partial w_{i,j}^m} = \tilde{s}_{i,h}^m \times \frac{\partial n_{i,q}^m}{\partial w_{i,j}^m} = \tilde{s}_{i,h}^m \times a_{j,q}^{m-1} \tag{12.43}$$

如果 $x_l$ 是偏置值

$$[\mathbf{J}]_{h,l} = \frac{\partial v_h}{\partial x_l} = \frac{\partial e_{k,q}}{\partial b_i^m} = \frac{\partial e_{k,q}}{\partial n_{i,q}^m} \times \frac{\partial n_{i,q}^m}{\partial b_i^m} = \tilde{s}_{i,h}^m \times \frac{\partial n_{i,q}^m}{\partial b_i^m} = \tilde{s}_{i,h}^m \tag{12.44}$$

Marquardt 敏感度可以通过标准敏感度同样的递归关系计算(式(11.35))，只是在最后一层有所修改：标准 BP 算法由式(11.40)计算。对 Marquardt 敏感度，有

12-24

$$\tilde{s}_{i,h}^M = \frac{\partial v_h}{\partial n_{i,q}^M} = \frac{\partial e_{k,q}}{\partial n_{i,q}^M} = \frac{\partial (t_{k,q} - a_{k,q}^M)}{\partial n_{i,q}^M} = -\frac{\partial a_{k,q}^M}{\partial n_{i,q}^M}$$

$$= \begin{cases} -\dot{f}^M(n_{i,q}^M), & i = k \\ 0, & i \neq k \end{cases} \tag{12.45}$$

所以当输入 $\mathbf{p}_q$ 作用于网络且对应的网络输出 $\mathbf{a}_q^M$ 计算出后，Levenbery-Marquardt 反向传播被初始化为

$$\tilde{\mathbf{S}}_q^M = -\dot{\mathbf{F}}^M(\mathbf{n}_q^M) \tag{12.46}$$

其中 $\dot{\mathbf{F}}^M(\mathbf{n}^M)$由(11.34)式定义。矩阵 $\tilde{\mathbf{S}}_q^M$ 的每一列用式(11.35)通过网络进行反向传播产生雅可比矩阵的一行。各个列也可以用下式进行反向传播：

$$\tilde{\mathbf{S}}_q^m = \dot{\mathbf{F}}^m(\mathbf{n}_q^m)(\mathbf{W}^{m+1})^T \tilde{\mathbf{S}}_q^{m+1} \tag{12.47}$$

每层的总体 Marquardt 敏感度矩阵可以由增广每个输入计算出的矩阵而创建：

$$\tilde{\mathbf{S}}^m = [\tilde{\mathbf{S}}_1^m \mid \tilde{\mathbf{S}}_2^m \mid \cdots \mid \tilde{\mathbf{S}}_Q^m] \tag{12.48}$$

注意，对每个提交给网络的输入，将反向传播 $S^M$ 的敏感度向量。这是由于计算每个单独误差的导数，而非平方误差和的导数。对每一个作用于网络的输入都有 $S^M$ 个误差(每个误差都对应于网络的一个输出)。对每个误差都有雅可比矩阵的一行。

当敏感度被反向传播后，雅可比矩阵由式(12.43)和(12.44)计算。请见例题 P12.5 关于雅可比计算的数值示例。

**LMBP**　Levenberg-Marquardt BP 算法(LMBP)的迭代过程概括如下：

1) 将所有输入提交网络并用式(11.41)和(11.42)计算相应的网络输出和误差 $\mathbf{e}_q = \mathbf{t}_q - \mathbf{a}_q^M$。用式(12.34)计算所有输入的平方误差和 $F(\mathbf{x})$。 12-25
2) 计算雅可比矩阵式(12.37)。首先用式(12.46)初始化敏感度，再用式(12.47)递归计算敏感度。用式(12.48)将各个单独的矩阵增广到 Marquardt 敏感度中。用式(12.43)和(12.44)计算雅可比阵的元素。
3) 解式(12.32)求得 $\Delta\mathbf{x}_k$。
4) 用 $\mathbf{x}_k + \Delta\mathbf{x}_k$ 重复计算平方误差的和。如果新的和小于第 1 步中计算的和，则用 $\mu$ 除以 $\theta$，并设 $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}_k$，转第 1 步；如果和没有减少，则用 $\mu$ 乘以 $\theta$，转第 3 步。

当梯度的模((12.22)式)小于给定值，或平方误差和减小到某个目标误差时，算法被认为收敛。

为了说明 LMBP，将它应用到本章开始时介绍的函数逼近的例子中。首先看一下基本的 Levenberg-Marquardt 计算步骤。图 12-17 说明在第一次迭代中 LMBP 算法可能产生的计算步骤。

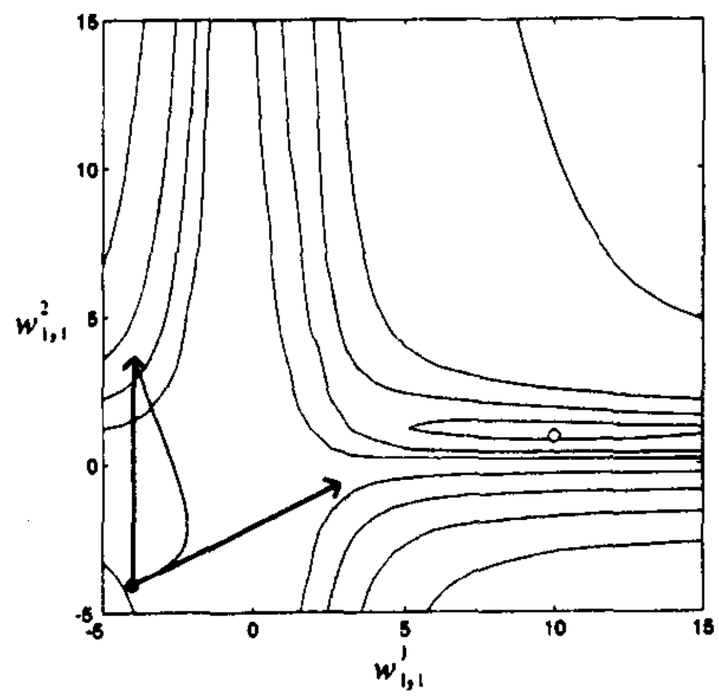



图 12-17　Levenberg-Marquardt 计算步骤

向上箭头表示较小 $\mu_k$ 所取的方向，对应于高斯-牛顿法的方向。偏右方箭头表示较大 $\mu_k$ 所取的方向，对应于最速下降法。(这是前面讨论过的算法的初始取向。)两箭头之间的线表示中等大小 $\mu_k$ 的 Levenberg-Marquardt 步骤。注意，当 $\mu_k$ 增加时算法向最速下降法的方向移动一小步。这意味着算法的每次迭代都能减少平方误差和。 12-26

图 12-18 显示 LMBP 轨迹的收敛路径($\mu_0 = 0.01$，$\theta = 5$)。注意，算法收敛的迭代次数较前面讨论的所有算法都少。当然这个算法在每次迭代时的计算量比任何其他算法大(因为要求矩阵的逆)。但是，对于中等数量的网络参数，即使要作大量计算，LMBP 算法依然是最快的神经网络训练算法[HaMe94]。

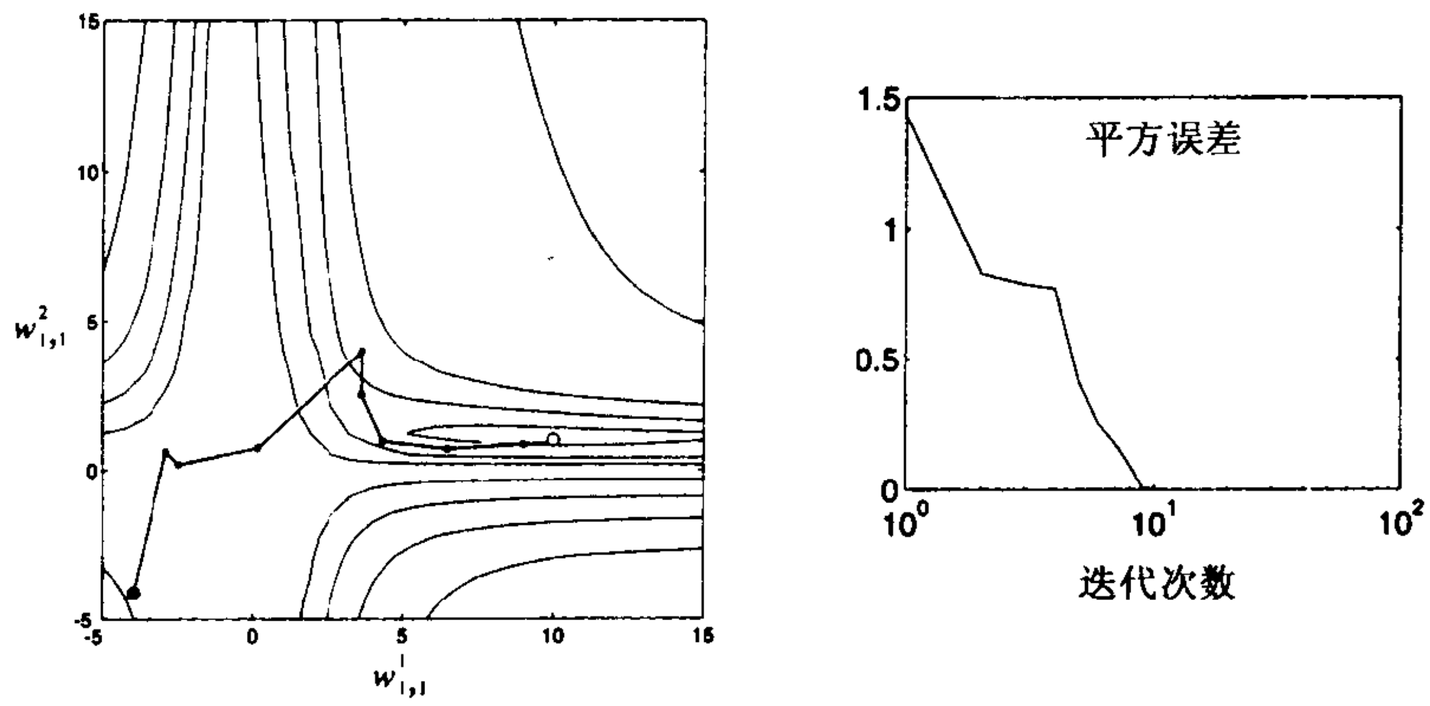



图 12-18 LMBP 轨迹

试验 LMBP 算法请用 *Neural Network Design Demonstration Marquardt Step* (**nnd12ms**)和 *Marquardt Backpropagation*(**nnd12m**)。

LMBP 算法的主要缺点是存储需求。算法需要存储近似赫森矩阵 $\mathbf{J}^T\mathbf{J}$，这是一个 $n \times n$ 矩阵，其中 $n$ 是网络中的参数(权值和偏置值)数目。回忆其他算法只要存储一个 $n$ 维向量的梯度。当参数数目非常大的时候，Levenberg-Marquardt 算法可能是不实用的(“非常大”依赖于你计算机上的存储器，但典型的上限是几千个参数)。

12-27

## 12.3 小结

### 启发式 BP 算法改进

**批处理**

在整个训练集都提交网络后才更新参数。平均每个样本计算出的梯度以得到更精确的梯度估计。(如果训练集是完全的，即覆盖了所有可能的输入/输出对，则梯度估计是精确的。)

**动量 BP 算法(MOBP)**

$$\Delta\mathbf{W}^m(k) = \gamma\Delta\mathbf{W}^m(k-1) - (1-\gamma)\alpha\mathbf{s}^m(\mathbf{a}^{m-1})^T$$

$$\Delta\mathbf{b}^m(k) = \gamma\Delta\mathbf{b}^m(k-1) - (1-\gamma)\alpha\mathbf{s}^m$$

**可变学习速度的 BP 算法(VLBP)**

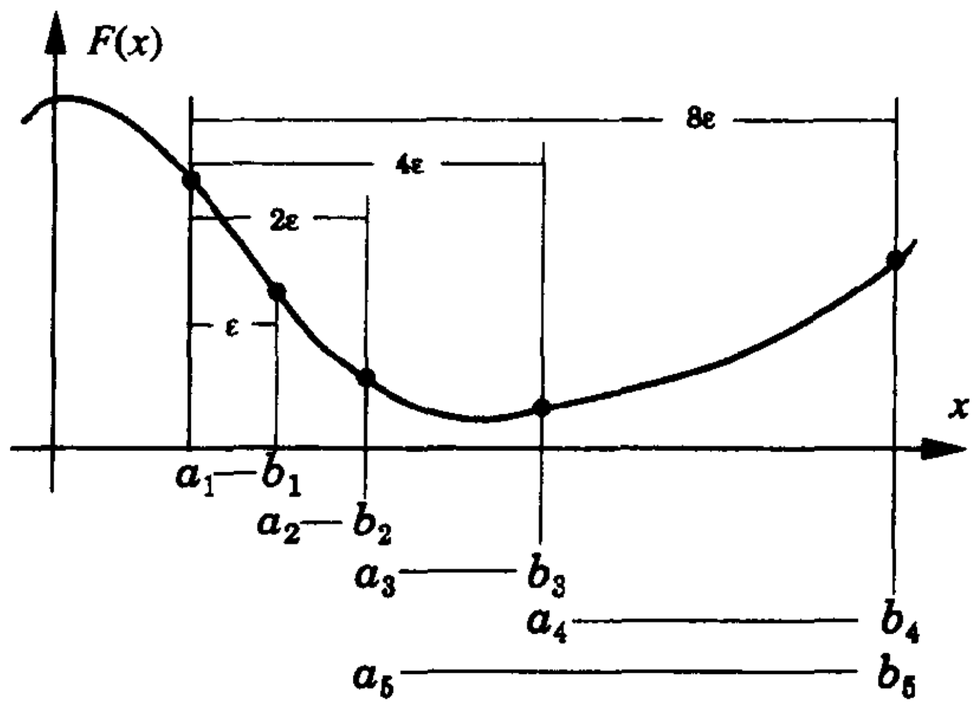
1) 如果一次权值改变后平方误差(在整个训练集上)的递增超过某个百分数 $\zeta$(典型值为 1% ~ 5%)，则权值改变被取消，学习速度乘以一个小于 1 的因子 $\rho$，动量系数 $\gamma$(如果有的话)设为 0。
2) 如果数值改变后的平方误差递减，则接受权值更新，学习速度乘以大于 1 的因子 $\eta$。如果 $\gamma$ 过去设置为 0，则恢复到原来的值。
3) 如果平方误差递增不超过 $\zeta$，则接受权值更新，但学习速度和动量系度都不变。

12-28

## 数值优化技术

### 共扼梯度法

区间位置




区间缩小(黄金分割搜索)

$$\tau = 0.618$$

set　$c_1 = a_1 + (1-\tau)(b_1 - a_1)$,　$F_c = F(c_1)$

　　$d_1 = b_1 - (1-\tau)(b_1 - a_1)$,　$F_d = F(d_1)$

for $k = 1, 2, \cdots$ repeat

　if $F_c < F_d$ then

　　set　$a_{k+1} = a_k$;　$b_{k+1} = d_k$;　$d_{k+1} = c_k$

　　$c_{k+1} = a_{k+1} + (1-\tau)(b_{k+1} - a_{k+1})$

　　$F_d = F_c$;　$F_c = F(c_{k+1})$

　else

　　set　$a_{k+1} = c_k$;　$b_{k+1} = b_k$;　$c_{k+1} = d_k$

　　$d_{k+1} = b_{k+1} - (1-\tau)(b_{k+1} - a_{k+1})$

　　$F_c = F_d$;　$F_d = F(d_{k+1})$

　end　　12-29

end until $b_{k+1} - a_{k+1} < tol$

### Levenberg - Marquardt BP(LMBP)算法

$$\Delta\mathbf{x}_k = -[\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k) + \mu_k\mathbf{I}]^{-1}\mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k)$$

$$\mathbf{v}^T = [v_1\ v_2 \cdots\ v_N] = [e_{1,1}\ e_{2,1} \cdots\ e_{S^M,1}\ e_{1,2} \cdots\ e_{S^M,Q}]$$

$$\mathbf{x}^T = [x_1\ x_2 \cdots\ x_n] = [w^1_{1,1}\ w^1_{1,2} \cdots\ w^1_{S^1,R}\ b^1_1 \cdots\ b^1_{S^1}\ w^2_{1,1} \cdots\ b^{M}_{S^M}]$$

$$N = Q \times S^M \text{ and } n = S^1(R+1) + S^2(S^1+1) + \cdots + S^M(S^{M-1}+1)$$

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial w^1_{1,1}} & \frac{\partial e_{1,1}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{1,1}}{\partial w^1_{S^1,R}} & \frac{\partial e_{1,1}}{\partial b^1_1} & \cdots \\ \frac{\partial e_{2,1}}{\partial w^1_{1,1}} & \frac{\partial e_{2,1}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{2,1}}{\partial w^1_{S^1,R}} & \frac{\partial e_{2,1}}{\partial b^1_1} & \cdots \\ \vdots & \vdots & & \vdots & \vdots & \\ \frac{\partial e_{S^M,1}}{\partial w^1_{1,1}} & \frac{\partial e_{S^M,1}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{S^M,1}}{\partial w^1_{S^1,R}} & \frac{\partial e_{S^M,1}}{\partial b^1_1} & \cdots \\ \frac{\partial e_{1,2}}{\partial w^1_{1,1}} & \frac{\partial e_{1,2}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{1,2}}{\partial w^1_{S^1,R}} & \frac{\partial e_{1,2}}{\partial b^1_1} & \cdots \\ \vdots & \vdots & & \vdots & \vdots & \end{bmatrix}$$

对权值 $x_l$

$$[\mathbf{J}]_{h,l} = \frac{\partial v_h}{\partial x_l} = \frac{\partial e_{k,q}}{\partial w^m_{i,j}} = \frac{\partial e_{k,q}}{\partial n^m_{i,q}} \times \frac{\partial n^m_{i,q}}{\partial w^m_{i,j}} = \tilde{s}^m_{i,h} \times \frac{\partial n^m_{i,q}}{\partial w^m_{i,j}} = \tilde{s}^m_{i,h} \times a^{m-1}_{j,q}$$

对偏置值 $x_l$

$$[\mathbf{J}]_{h,l} = \frac{\partial v_h}{\partial x_l} = \frac{\partial e_{k,q}}{\partial b^m_i} = \frac{\partial e_{k,q}}{\partial n^m_{i,q}} \times \frac{\partial n^m_{i,q}}{\partial b^m_i} = \tilde{s}^m_{i,h} \times \frac{\partial n^m_{i,q}}{\partial b^m_i} = \tilde{s}^m_{i,h}$$

$$\tilde{s}^m_{i,h} \equiv \frac{\partial v_h}{\partial n^m_{i,q}} = \frac{\partial e_{k,q}}{\partial n^m_{i,q}} \text{(Marquardt 敏感度)，其中 } h = (q-1)S^M + k$$

$$\tilde{\mathbf{S}}^M_q = -\dot{\mathbf{F}}^M(\mathbf{n}^M_q)$$

$$\tilde{\mathbf{S}}^m_q = \dot{\mathbf{F}}^m(\mathbf{n}^m_q)(\mathbf{W}^{m+1})^T\tilde{\mathbf{S}}^{m+1}_q$$

12-30

$$\tilde{\mathbf{S}}^m = \left[\tilde{\mathbf{S}}^m_1 \,|\, \tilde{\mathbf{S}}^m_2 \,|\, \cdots \,|\, \tilde{\mathbf{S}}^m_Q\right]$$

Levenberg-Marquardt 迭代

1) 将所有输入提交网络并用式(11.41)和(11.42)计算相应的网络输出和误差 $\mathbf{e}_q = \mathbf{t}_q - \mathbf{a}^M_q$。用式(12.34)计算所有输入的平方误差之和 $F(\mathbf{x})$。
2) 用式(12.37)计算雅可比矩阵。首先用(12.46)式初始化敏感度，再用(12.47)递归计算，用式(12.48)将各个矩阵增广到 Marquardt 敏感度中。用式(12.43)和(12.44)计算雅可比阵的元素。
3) 解(12.32)式求 $\Delta\mathbf{x}_k$。
4) 用 $\mathbf{x}_k + \Delta\mathbf{x}_k$ 重复计算平方误差之和。如果新的和小于第 1 步中计算所得的和，则把 $\mu$ 除以 $\theta$，并设 $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}_k$，转第 1 步。如果平方误差和没有减少，则 $\mu$ 乘以 $\theta$，转第 3 步。

12-31

## 12.4 例题

**P12.1** 用训练集$\{(\mathbf{p}_1 = [-3]), (\mathbf{t}_1 = [0.5])\}$，$\{(\mathbf{p}_2 = [2]), (\mathbf{t}_2 = [1])\}$训练图 12-19

中的网络，初始值为 $w(0)=0.4$，$b(0)=0.15$。用批处理和非批处理的 SDBP 方法，说明第一步方向计算中批处项的影响。

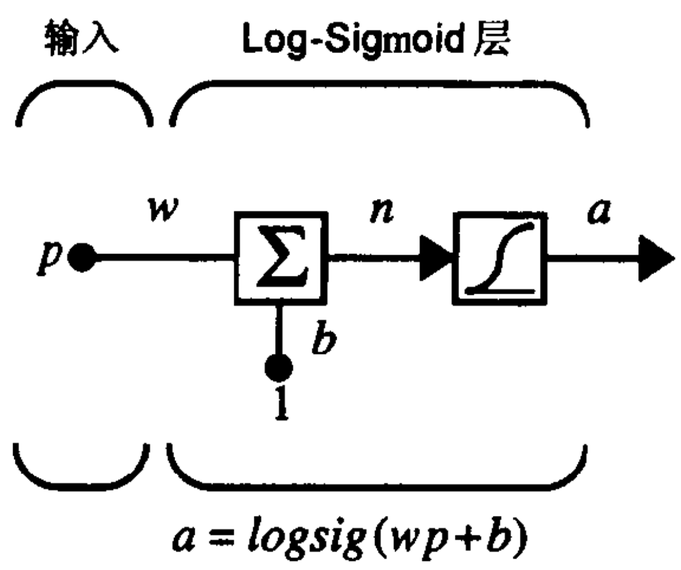



图 12-19　例题 P12.1 的网络

**解**

首先计算不用批处理的初始步的方向。此时第 1 步由第一个输入/目标对计算。前向传播步和反向传播步为

$$a = logsig(wp+b) = \frac{1}{1+\exp(-(0.4(-3)+0.15))} = 0.2592$$

$$e = t - a = 0.5 - 0.2592 = 0.2408$$

$$s = -2\dot{f}(n)e = -2a(1-a)e = -2(0.2592)(1-0.2592)0.2408 = -0.0925$$

初始步的方向是梯度的反方向。此时权值为

$$-sp = -(-0.0925)(-3) = -0.2774$$

对于偏置值有

$$-s = -(-0.0925) = 0.0925$$

12-32

因此在$(w, b)$平面中初始步的方向为

$$\begin{bmatrix} -0.2774 \\ 0.0925 \end{bmatrix}$$

现在考虑批处理算法的初始步的方向。此时的梯度是由两个输入/目标对集合的梯度迭加而成。因此，需要将第 2 个输入提交网络并进行前向和反向传播处理步骤：

$$a = logsig(wp+b) = \frac{1}{1+\exp(-(0.4(2)+0.15))} = 0.7211$$

$$e = t - a = 1 - 0.7211 = 0.2789$$

$$s = -2\dot{f}(n)e = -2a(1-a)e = -2(0.7211)(1-0.7211)0.2789 = -0.1122$$

操作步的方向是梯度的反方向。对权值这将是

$$-sp = -(-0.1122)(2) = 0.2243$$

对偏置值有

$$-s = -(-0.1122) = 0.1122$$

因此第二个输入/目标对的部分梯度为

$$\begin{bmatrix} 0.2243 \\ 0.1122 \end{bmatrix}$$

如果将两个输入/目标对的结果相加，可以得到批处理模式下 SDBP 第 1 步的方向为

$$\frac{1}{2}\left(\begin{bmatrix}-0.2774\\0.0925\end{bmatrix}+\begin{bmatrix}0.2243\\0.1122\end{bmatrix}\right)=\frac{1}{2}\begin{bmatrix}-0.0531\\0.2047\end{bmatrix}=\begin{bmatrix}-0.0265\\0.1023\end{bmatrix}$$

结果如图 12-20 所示。黑圆圈指示初始点。两边的箭头表示两个输入/目标对的部分梯度方向，中间的箭头表示总梯度的方向。画出的函数是整个训练集的平方误差之和。注意单个梯度分量可以指向与真实梯度完全不同的方向。但是，一般说来，在若干次迭代后，路径将沿着最速下降轨迹。

12-33

批处理模式对逐渐逼近的相对影响是强烈依赖于特定问题的。逐渐逼近不需要更多的存储，而且如果提交给网络的输入是随机的，轨迹也会是随机的，这使算法有时会落入局部极小点，并且较批处理算法花费更多的时间。

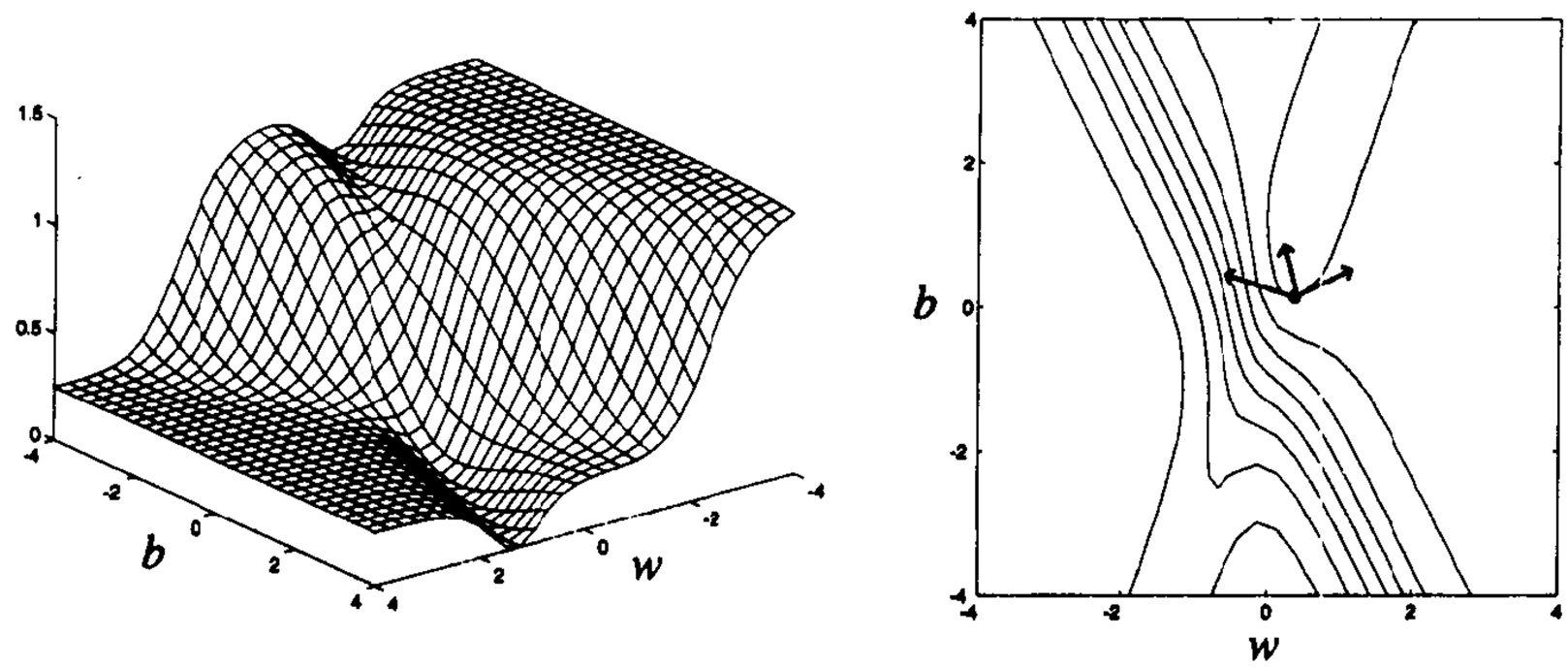



图 12-20 批处理对例题 P12.1 的影响

**P12.2** 在第 9 章中证明了将最速下降法应用于二次函数时，如果学习速度小于 2 除以赫森矩阵的最大特征值，则算法是稳定的。如果将动量项加入到最速下降法，证明总是由动量系数决定算法的稳定性而非学习速度。遵循 9.2.1 节的“稳定的学习速度”一段的格式进行证明。

**解**

标准最速下降算法为

$$\Delta\mathbf{x}_k = -\alpha\nabla F(\mathbf{x}_k) = -\alpha\mathbf{g}_k$$

如果加上动量项，它变成

$$\Delta\mathbf{x}_k = \gamma\Delta\mathbf{x}_{k-1} - (1-\gamma)\alpha\mathbf{g}_k$$

由第 8 章，二次函数的形式为

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{d}^T\mathbf{x} + c$$

二次函数的梯度为

12-34

$$\nabla F(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{d}$$

将该式代入有动量项的最速下降算法中，得到

$$\Delta\mathbf{x}_k = \gamma\Delta\mathbf{x}_{k-1} - (1-\gamma)\alpha(\mathbf{A}\mathbf{x}_k + \mathbf{d})$$

使用定义 $\Delta\mathbf{x}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$，上式可写为

$$\mathbf{x}_{k+1} - \mathbf{x}_k = \gamma(\mathbf{x}_k - \mathbf{x}_{k-1}) - (1-\gamma)\alpha(\mathbf{A}\mathbf{x}_k + \mathbf{d})$$

或

$$\mathbf{x}_{k+1} = [(1+\gamma)\mathbf{I} - (1-\gamma)\alpha\mathbf{A}]\mathbf{x}_k - \gamma\mathbf{x}_{k-1} - (1-\gamma)\alpha\mathbf{d}$$

现在定义一个新向量

$$\tilde{\mathbf{x}}_k = \begin{bmatrix} \mathbf{x}_{k-1} \\ \mathbf{x}_k \end{bmatrix}$$

带动量的最速下降法变形可写成

$$\tilde{\mathbf{x}}_{k+1} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\gamma\mathbf{I} & [(1+\gamma)\mathbf{I} - (1-\gamma)\alpha\mathbf{A}] \end{bmatrix}\tilde{\mathbf{x}}_k + \begin{bmatrix} \mathbf{0} \\ -(1-\gamma)\alpha\mathbf{d} \end{bmatrix} = \mathbf{W}\tilde{\mathbf{x}}_k + \mathbf{v}$$

如果 $\mathbf{W}$ 的特征值的模都小于 1，则该线性系统是稳定的。我们将找到 $\mathbf{W}$ 的特征值。首先，重写 $\mathbf{W}$ 为

$$\mathbf{W} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\gamma\mathbf{I} & \mathbf{T} \end{bmatrix}, \quad \text{其中 } \mathbf{T} = [(1+\gamma)\mathbf{I} - (1-\gamma)\alpha\mathbf{A}]$$

$\mathbf{W}$ 的特值和特征向量应满足

$$\mathbf{W}\mathbf{z}^w = \lambda^w\mathbf{z}^w \quad \text{或} \quad \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\gamma\mathbf{I} & \mathbf{T} \end{bmatrix}\begin{bmatrix} \mathbf{z}_1^w \\ \mathbf{z}_2^w \end{bmatrix} = \lambda^w\begin{bmatrix} \mathbf{z}_1^w \\ \mathbf{z}_2^w \end{bmatrix}$$

即

$$\mathbf{z}_2^w = \lambda^w\mathbf{z}_1^w \quad \text{和} \quad -\gamma\mathbf{z}_1^w + \mathbf{T}\mathbf{z}_2^w = \lambda^w\mathbf{z}_2^w$$

此时选择 $\mathbf{z}_2^w$ 作为矩阵 $\mathbf{T}$ 的特征向量，对应的特征值为 $\lambda^t$。(如果这个选择不恰当，将会导出矛盾。)因此上式变为 12-35

$$\mathbf{z}_2^w = \lambda^w\mathbf{z}_1^w \quad \text{和} \quad -\gamma\mathbf{z}_1^w + \lambda^t\mathbf{z}_2^w = \lambda^w\mathbf{z}_2^w$$

将第一个式子代入第二个式子有

$$-\frac{\gamma}{\lambda^w}\mathbf{z}_2^w + \lambda^t\mathbf{z}_2^w = \lambda^w\mathbf{z}_2^w \quad \text{或} \quad [(\lambda^w)^2 - \lambda^t(\lambda^w) + \gamma]\mathbf{z}_2^w = 0$$

因此，对于 $\mathbf{T}$ 的每个特征值 $\lambda^t$ 都有 $\mathbf{W}$ 的两个特征值 $\lambda^w$ 满足二次方程

$$(\lambda^w)^2 - \lambda^t(\lambda^w) + \gamma = 0$$

由二次方程求根公式

$$\lambda^w = \frac{\lambda^t \pm \sqrt{(\lambda^t)^2 - 4\gamma}}{2}$$

如果算法要稳定，则要求每个特征值的模都小于 1。我们将说明总是存在 $\gamma$ 的一个范围满足这个条件。

注意，如果特征值 $\lambda^w$ 为复数，则它的模为$\sqrt{\gamma}$：

$$|\lambda^w| = \sqrt{\frac{(\lambda^t)^2}{4} + \frac{4\gamma - (\lambda^t)^2}{4}} = \sqrt{\gamma}$$

(这仅当 $\lambda^t$ 为实数时成立。下面将说明 $\lambda^t$ 为实数。)由于 $\gamma$ 在 0 和 1 之间，所以特征值的模必小于 1。下面将说明，存在 $\gamma$ 的一个范围使所有的特征值都为复数。

为了使 $\lambda^w$ 为复数，必须有

$$(\lambda^t)^2 - 4\gamma < 0 \quad \text{或} \quad |\lambda^t| < 2\sqrt{\gamma}$$

考虑 **T** 的特征值 $\lambda^t$。这些特征值可以由 **A** 的特征值表示。设$\{\lambda_1, \lambda_2, \cdots, \lambda_n\}$ 和$\{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n\}$为赫森矩阵的特征值和特征向量，则

12-36

$$\begin{aligned}\mathbf{T}\mathbf{z}_i &= [(1+\gamma)\mathbf{I} - (1-\gamma)\alpha\mathbf{A}]\mathbf{z}_i = (1+\gamma)\mathbf{z}_i - (1-\gamma)\alpha\mathbf{A}\mathbf{z}_i \\ &= (1+\gamma)\mathbf{z}_i - (1-\gamma)\alpha\lambda_i\mathbf{z}_i = \{(1+\gamma) - (1-\gamma)\alpha\lambda_i\}\mathbf{z}_i = \lambda_i^t\mathbf{z}_i\end{aligned}$$

因此 **T** 的特征向量与 **A** 的特征向量相同，且 **T** 的特征值为

$$\lambda_i^t = \{(1+\gamma) - (1-\gamma)\alpha\gamma_i\}$$

(注意对于对称矩阵 **A** 而言，由于 $\gamma$，$\alpha$ 和 $\lambda_i$ 均为实数，所以 $\lambda_i^t$也是实数。)为了使 $\lambda^w$ 为复数，必需有

$$|\lambda^t| < 2\sqrt{\gamma} \quad 或 \quad |(1+\gamma) - (1-\gamma)\alpha\lambda_i| < 2\sqrt{\gamma}$$

当 $\gamma = 1$ 时，不等式两边均为 2。不等式右端作为 $\gamma$ 的函数，在 $\lambda = 1$ 的斜率为 1。不等式左端的函数的斜率为 $1 + \alpha\lambda_i$。由于赫森矩阵的特征值在函数有一个强极小点时将是正实数，且学习速度为正数，此斜率必大于 1。这说明当 $\lambda$ 足够接近于 1 时该不等式总是成立。

作为结论，我们证明了如果将动量项加到二次函数的最速下降算法中，则总有一个动量系数将使整个算法稳定，而不管学习速度如何。另外证明了如果 $\lambda$ 趋近于 1，则 **W** 的特征值的模为$\sqrt{\gamma}$。可以证明[Brog91]特征值的模决定了算法的收敛速度。模越小，收敛速度越快。当模趋近于 1 时，收敛时间加快。

可以用图 9-3 的例子来说明这些结果。在那里证明了当学习速度 $\alpha > 0.4$ 时，最速下降法对函数 $F(\mathbf{x}) = x_1^2 + 25x_2^2$ 是不稳定的。在图 12-21 中可以看到当 $\alpha = 0.041$，$\gamma = 0.2$ 时具有动量项的最速下降法轨迹。将这个轨迹与图 9-3(它有相同的学习速度，但没有动量项)比较。

12-37

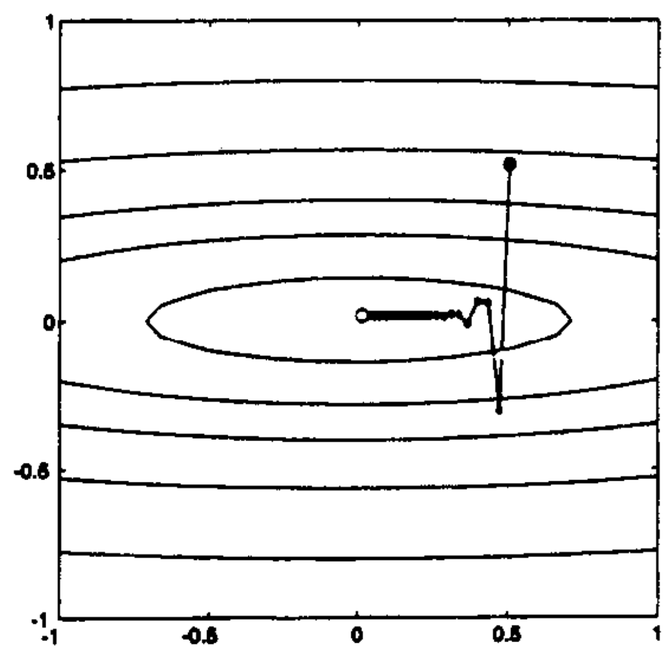



图 12-21 $\alpha = 0.041$ 和 $\gamma = 0.2$ 时的轨迹

**P12.3** 对下述函数执行 3 次可变学习速度算法的迭代：

$$F(\mathbf{x}) = x_1^2 + 25x_2^2$$

(该函数取自第 9 章 9.2.1 节中“稳定的学习速度”的例子）初始值为

$$\mathbf{x}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

算法的参数为：

$$\alpha = 0.05, \quad \gamma = 0.2, \quad \eta = 1.5, \quad \rho = 0.5, \quad \zeta = 5\%$$

**解**

第 1 步是计算初始点的函权值：

$$F(\mathbf{x}_0) = \frac{1}{2}\mathbf{x}_0^T\begin{bmatrix}2 & 0\\0 & 50\end{bmatrix}\mathbf{x}_0 = \frac{1}{2}[0.5 \quad 0.5]\begin{bmatrix}2 & 0\\0 & 50\end{bmatrix}\begin{bmatrix}0.5\\0.5\end{bmatrix} = 6.5$$

下一步是求梯度

$$\nabla F(\mathbf{x}) = \begin{bmatrix}\frac{\partial}{\partial x_1}F(\mathbf{x})\\ \frac{\partial}{\partial x_2}F(\mathbf{x})\end{bmatrix} = \begin{bmatrix}2x_1\\50x_2\end{bmatrix}$$

如果计算初始点的梯度，则有 12-38

$$\mathbf{g}_0 = \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0} = \begin{bmatrix}1\\25\end{bmatrix}$$

在初始学习速度为 $\alpha = 0.05$ 时，算法的第一步尝试为

$$\Delta\mathbf{x}_0 = \gamma\Delta\mathbf{x}_{-1} - (1-\gamma)\alpha\mathbf{g}_0 = 0.2\begin{bmatrix}0\\0\end{bmatrix} - 0.8(0.05)\begin{bmatrix}1\\25\end{bmatrix} = \begin{bmatrix}-0.04\\-1\end{bmatrix}$$

$$\mathbf{x}_1^t = \mathbf{x}_0 + \Delta\mathbf{x}_0 = \begin{bmatrix}0.5\\0.5\end{bmatrix} + \begin{bmatrix}-0.04\\-1\end{bmatrix} = \begin{bmatrix}0.46\\-0.5\end{bmatrix}$$

为验证这一步的有效性，计算在这个新点的函权值：

$$F(\mathbf{x}_1^t) = \frac{1}{2}(\mathbf{x}_1^t)^T\begin{bmatrix}2 & 0\\0 & 50\end{bmatrix}\mathbf{x}_1^t = \frac{1}{2}[0.46 \ -0.5]\begin{bmatrix}2 & 0\\0 & 50\end{bmatrix}\begin{bmatrix}0.46\\-0.5\end{bmatrix} = 6.4616$$

它小于 $F(\mathbf{x}_0)$。因此，这一试验步被接受，而且学习速度增加：

$$\mathbf{x}_1 = \mathbf{x}_1^t = \begin{bmatrix}0.46\\-0.5\end{bmatrix}, F(\mathbf{x}_1) = 6.4616 \text{ 和 } \alpha = \eta\alpha = 1.5(0.05) = 0.075$$

算法第二步试验的计算为：

$$\Delta\mathbf{x}_1 = \gamma\Delta\mathbf{x}_0 - (1-\gamma)\alpha\mathbf{g}_1 = 0.2\begin{bmatrix}-0.04\\-1\end{bmatrix} - 0.8(0.075)\begin{bmatrix}0.92\\-25\end{bmatrix} = \begin{bmatrix}-0.0632\\1.3\end{bmatrix}$$

$$\mathbf{x}_2^t = \mathbf{x}_1 + \Delta\mathbf{x}_1 = \begin{bmatrix}0.46\\-0.5\end{bmatrix} + \begin{bmatrix}-0.0632\\1.3\end{bmatrix} = \begin{bmatrix}0.3968\\0.8\end{bmatrix}$$

计算这一点的函权值：

$$F(\mathbf{x}_2^t) = \frac{1}{2}(\mathbf{x}_2^t)^T\begin{bmatrix}2 & 0\\0 & 50\end{bmatrix}\mathbf{x}_2^t = \frac{1}{2}[0.3968 \ 0.8]\begin{bmatrix}2 & 0\\0 & 50\end{bmatrix}\begin{bmatrix}0.3968\\0.8\end{bmatrix} = 16.157$$

由于它比 $F(\mathbf{x}_1)$大 5%，放弃这一步，减少学习速度，并设动量系数为 0。

$$\mathbf{x}_2 = \mathbf{x}_1, \quad F(\mathbf{x}_2) = F(\mathbf{x}_1) = 6.4616, \quad \alpha = \rho\alpha = 0.5(0.075) = 0.0375, \quad \gamma = 0$$ 12-39

现在试验新一步的计算(动量为 0)：

$$\Delta\mathbf{x}_2 = -\alpha\mathbf{g}_2 = -(0.0375)\begin{bmatrix}0.92\\-25\end{bmatrix} = \begin{bmatrix}-0.0345\\0.9375\end{bmatrix}$$

$$\mathbf{x}_3^t = \mathbf{x}_2 + \Delta\mathbf{x}_2 = \begin{bmatrix}0.46\\-0.5\end{bmatrix} + \begin{bmatrix}-0.0345\\0.9375\end{bmatrix} = \begin{bmatrix}0.4255\\0.4375\end{bmatrix}$$

$$F(\mathbf{x}_3^t) = \frac{1}{2}(\mathbf{x}_3^t)^T\begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix}\mathbf{x}_3^t = \frac{1}{2}[0.4255\ 0.4375]\begin{bmatrix} 2 & 0 \\ 0 & 50 \end{bmatrix}\begin{bmatrix} 0.4255 \\ 0.4375 \end{bmatrix} = 4.966$$

它小于 $F(\mathbf{x}_2)$。因此这一步被接受，动量系数恢复到初始值，且学习速度增加。

$$\mathbf{x}_3 = \mathbf{x}_3^t, \quad \gamma = 0.2, \quad \alpha = \eta\alpha = 1.5(0.0375) = 0.05625$$

这就完成了第 3 次迭代。

**P12.4** 回忆第 9 章中用以说明共轭梯度算法的例子见(9.2.3 节)：

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}\mathbf{x}$$

初始点为

$$\mathbf{x}_0 = \begin{bmatrix} 0.8 \\ -0.25 \end{bmatrix}$$

执行共轭梯度算法的一次迭代。在线性极小化中用函数计算进行区间定位，并用黄金分割搜索算法进行区间缩小。

**解**

函数的梯度为

$$\nabla F(\mathbf{x}) = \begin{bmatrix} 2x_1 + x_2 \\ x_1 + 2x_2 \end{bmatrix}$$

12-40 由最速下降法，共轭梯度法的第一个搜索方向是梯度的反方向：

$$\mathbf{p}_0 = -\mathbf{g}_0 = -\nabla F(\mathbf{x})^T|_{\mathbf{x}=\mathbf{x}_0} = \begin{bmatrix} -1.35 \\ -0.3 \end{bmatrix}$$

在第一次迭代中，要沿着下述直线极小化 $F(\mathbf{x})$：

$$\mathbf{x}_1 = \mathbf{x}_0 + \alpha_0\mathbf{p}_0 = \begin{bmatrix} 0.8 \\ -0.25 \end{bmatrix} + \alpha_0\begin{bmatrix} -1.35 \\ -0.3 \end{bmatrix}$$

第 1 步是区间定位。假设初始步长 $\varepsilon = 0.075$。区间定位过程如下：

$$F(a_1) = F\left(\begin{bmatrix} 0.8 \\ -0.25 \end{bmatrix}\right) = 0.5025$$

$$b_1 = \varepsilon = 0.075, \quad F(b_1) = F\left(\begin{bmatrix} 0.8 \\ -0.25 \end{bmatrix} + 0.075\begin{bmatrix} -1.35 \\ -0.3 \end{bmatrix}\right) = 0.3721$$

$$b_2 = 2\varepsilon = 0.15, \quad F(b_2) = F\left(\begin{bmatrix} 0.8 \\ -0.25 \end{bmatrix} + 0.15\begin{bmatrix} -1.35 \\ -0.3 \end{bmatrix}\right) = 0.2678$$

$$b_3 = 4\varepsilon = 0.3, \quad F(b_3) = F\left(\begin{bmatrix} 0.8 \\ -0.25 \end{bmatrix} + 0.3\begin{bmatrix} -1.35 \\ -0.3 \end{bmatrix}\right) = 0.1373$$

$$b_4 = 8\varepsilon = 0.6, \quad F(b_4) = F\left(\begin{bmatrix} 0.8 \\ -0.25 \end{bmatrix} + 0.6\begin{bmatrix} -1.35 \\ -0.3 \end{bmatrix}\right) = 0.1893$$

由于函数在连续两次计值之间递增，所以我们知道极小值在[0.15　0.6]区间内。该过程由图 12-22 中的小圆圈表示，最后的区间由大的圆圈表示。

下一步线性极小化是用黄金分割搜索算法进行区间缩小。过程如下：

$$c_1 = a_1 + (1-\tau)(b_1 - a_1) = 0.15 + (0.382)(0.6 - 0.15) = 0.3219$$

$$d_1 = b_1 - (1-\tau)(b_1 - a_1) = 0.6 - (0.382)(0.6 - 0.15) = 0.4281$$

$$F_a = 0.2678, \quad F_b = 0.1893, \quad F_c = 0.1270, \quad F_d = 0.1085$$

由于 $F_c > F_d$，我们有 12-41

$$a_2 = c_1 = 0.3219, \quad b_2 = b_1 = 0.6, \quad c_2 = d_1 = 0.4281$$

$$d_2 = b_2 - (1-\tau)(b_2 - a_2) = 0.6 - (0.382)(0.6 - 0.3219) = 0.4938$$

$$F_a = F_c = 0.1270, \quad F_c = F_d = 0.1085, \quad F_d = F(d_2) = 0.1232$$

此时 $F_c < F_d$，因此

$$a_3 = a_2 = 0.3219, \quad b_3 = d_2 = 0.4938, \quad d_3 = c_2 = 0.4281$$

$$c_3 = a_3 + (1-\tau)(b_3 - a_3) = 0.3219 + (0.382)(0.4938 - 0.3219) = 0.3876$$

$$F_b = F_d = 0.1232, \quad F_d = F_c = 0.1085, \quad F_c = F(c_3) = 0.1094$$

该过程继续直至 $b_{k+1} - a_{k+1} < tol$。图 12-22 中的小黑点表示每次迭代过程的一个新内部点的位置。最后的点由大黑点表示。将结果与图 9-10 中显示的第一次迭代结果比较。

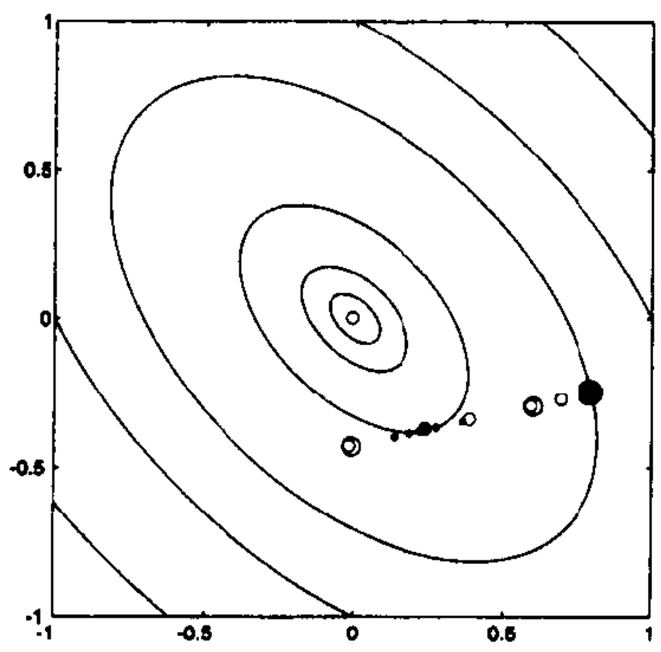



图 12-22　线性极小化的例子

**P12.5** 为说明 Levenberg-Marquardt 方法中雅可比矩阵的计算过程，考虑使用图 12-23 中求解函数逼近的网络。选择的网络传输函数为

$$f^1(n) = (n)^2, \quad f^2(n) = n$$

它们的导数分别为 12-42

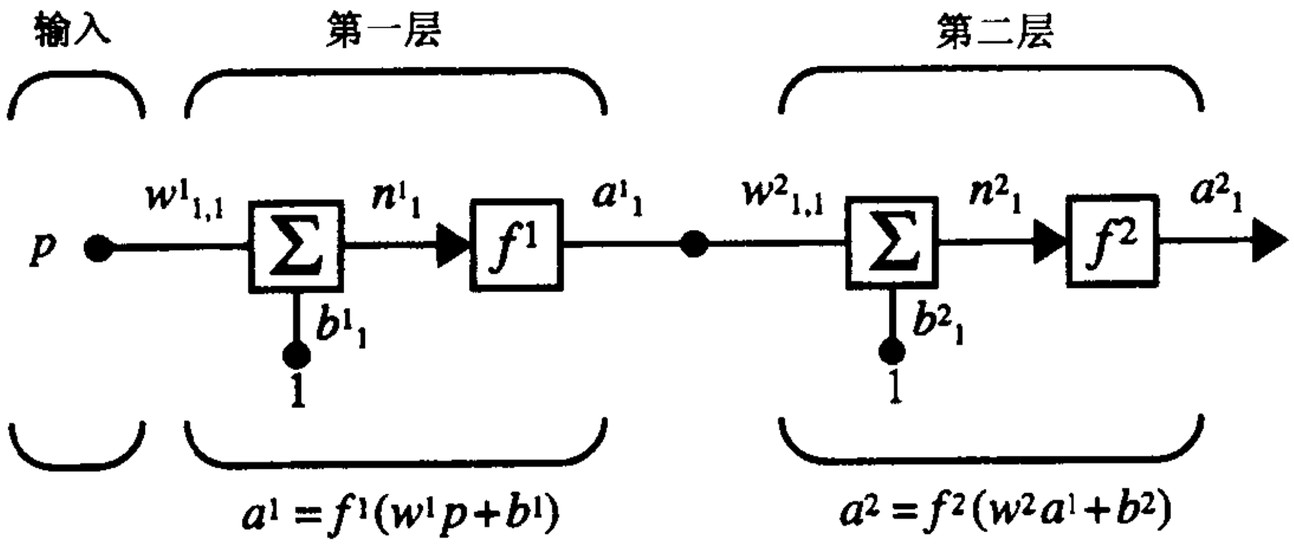



图 12-23　说明 LMBP 的两层网络

$$\dot{f}^1(n) = 2n, \quad \dot{f}^2(n) = 1$$

假设训练集包括

$$\{(\mathbf{p}_1 = [1]),(\mathbf{t}_1 = [1])\}, \quad \{(\mathbf{p}_2 = [2]),(\mathbf{t}_2 = [2])\}$$

参数被初始化为

$$\mathbf{W}^1 = [1], \quad \mathbf{b}^1 = [0], \quad \mathbf{W}^2 = [2], \quad \mathbf{b}^1 = [1]$$

计算 Levenberg-Marquardt 方法中第一步的雅可比矩阵。

**解**

第 1 步是在网络中传播输入并计算误差：

$$\mathbf{a}_1^0 = \mathbf{p}_1 = [1]$$

$$\mathbf{n}_1^1 = \mathbf{W}^1\mathbf{a}_1^0 + \mathbf{b}^1 = [1][1] + [0] = [1], \mathbf{a}_1^1 = \mathbf{f}^1(\mathbf{n}_1^1) = ([1])^2 = [1]$$

$$\mathbf{n}_1^2 = \mathbf{W}^2\mathbf{a}_1^1 + \mathbf{b}^2 = ([2][1] + [1]) = [3], \mathbf{a}_1^2 = \mathbf{f}^2(\mathbf{n}_1^2) = ([3]) = [3]$$

$$\mathbf{e}_1 = (\mathbf{t}_1 - \mathbf{a}_1^2) = ([1] - [3]) = [-2]$$

$$\mathbf{a}_2^0 = \mathbf{p}_2 = [2]$$

$$\mathbf{n}_2^1 = \mathbf{W}^1\mathbf{a}_2^0 + \mathbf{b}^1 = [1][2] + [0] = [2], \mathbf{a}_2^1 = \mathbf{f}^1(\mathbf{n}_2^1) = ([2])^2 = [4]$$

$$\mathbf{n}_2^2 = \mathbf{W}^2\mathbf{a}_2^1 + \mathbf{b}^2 = ([2][4] + [1]) = [9], \mathbf{a}_2^2 = \mathbf{f}^2(\mathbf{n}_2^2) = ([9]) = [9]$$

12-43

$$\mathbf{e}_1 = (\mathbf{t}_1 - \mathbf{a}_1^2) = ([2] - [9]) = [-7]$$

第 2 步是用式(12.46)和(12.47)初始化并反向传播 Marquardt 敏感度。

$$\tilde{\mathbf{S}}_1^2 = -\dot{\mathbf{F}}^2(\mathbf{n}_1^2) = -[1]$$

$$\tilde{\mathbf{S}}_1^1 = \dot{\mathbf{F}}^1(\mathbf{n}_1^1)(\mathbf{W}^2)^T\tilde{\mathbf{S}}_1^2 = [2n_{1,1}^1][2][-1] = [2(1)][2][-1] = [-4]$$

$$\tilde{\mathbf{S}}_2^2 = -\dot{\mathbf{F}}^2(\mathbf{n}_2^2) = -[1]$$

$$\tilde{\mathbf{S}}_2^1 = \dot{\mathbf{F}}^1(\mathbf{n}_2^1)(\mathbf{W}^2)^T\tilde{\mathbf{S}}_2^2 = [2n_{1,2}^2][2][-1] = [2(2)][2][-1] = [-8]$$

$$\tilde{\mathbf{S}}^1 = \left[\tilde{\mathbf{S}}_1^1 | \tilde{\mathbf{S}}_2^1\right] = [-4 \quad -8], \quad \tilde{\mathbf{S}}^2 = \left[\tilde{\mathbf{S}}_1^2 | \tilde{\mathbf{S}}_2^2\right] = [-1 \quad -1]$$

现在用式(12.43)，(12.44)和(12.37)计算雅可比矩阵。

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial v_1}{\partial x_1} & \frac{\partial v_1}{\partial x_2} & \frac{\partial v_1}{\partial x_3} & \frac{\partial v_1}{\partial x_4} \\ \frac{\partial v_2}{\partial x_1} & \frac{\partial v_2}{\partial x_2} & \frac{\partial v_2}{\partial x_3} & \frac{\partial v_2}{\partial x_4} \end{bmatrix} = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial w_{1,1}^1} & \frac{\partial e_{1,1}}{\partial b_1^1} & \frac{\partial e_{1,1}}{\partial w_{1,1}^2} & \frac{\partial e_{1,1}}{\partial b_1^2} \\ \frac{\partial e_{1,2}}{\partial w_{1,1}^1} & \frac{\partial e_{1,2}}{\partial b_1^1} & \frac{\partial e_{1,2}}{\partial w_{1,1}^2} & \frac{\partial e_{1,2}}{\partial b_1^2} \end{bmatrix}$$

$$[\mathbf{J}]_{1,1} = \frac{\partial v_1}{\partial x_1} = \frac{\partial e_{1,1}}{\partial w_{1,1}^1} = \frac{\partial e_{1,1}}{\partial n_{1,1}^1} \times \frac{\partial n_{1,1}^1}{\partial w_{1,1}^1} = \tilde{s}_{1,1}^1 \times \frac{\partial n_{1,1}^1}{\partial w_{1,1}^1} = \tilde{s}_{1,1}^1 \times a_{1,1}^0$$

$$= (-4)(1) = -4$$

$$[\mathbf{J}]_{1,2} = \frac{\partial v_1}{\partial x_2} = \frac{\partial e_{1,1}}{\partial b_1^1} = \frac{\partial e_{1,1}}{\partial n_{1,1}^1} \times \frac{\partial n_{1,1}^1}{\partial b_1^1} = \tilde{s}_{1,1}^1 \times \frac{\partial n_{1,1}^1}{\partial b_1^1} = \tilde{s}_{1,1}^1 = -4$$

$$[\mathbf{J}]_{1,3} = \frac{\partial v_1}{\partial x_3} = \frac{\partial e_{1,1}}{\partial n_{1,1}^2} \times \frac{\partial n_{1,1}^2}{\partial w_{1,1}^2} = \tilde{s}_{1,1}^2 \times \frac{\partial n_{1,1}^2}{\partial w_{1,1}^2} = \tilde{s}_{1,1}^2 \times a_{1,1}^1 = (-1)(1) = -1$$

12-44

$$[\mathbf{J}]_{1,4} = \frac{\partial v_1}{\partial x_4} = \frac{\partial e_{1,1}}{\partial n_{1,1}^2} \times \frac{\partial n_{1,1}^1}{\partial b_1^2} = \tilde{s}_{1,1}^2 \times \frac{\partial n_{1,1}^2}{\partial b_1^2} = \tilde{s}_{1,1}^2 = -1$$

$$[\mathbf{J}]_{2,1} = \frac{\partial v_2}{\partial x_1} = \frac{\partial e_{1,2}}{\partial n_{1,2}^1} \times \frac{\partial n_{1,2}^1}{\partial w_{1,1}^1} = \tilde{s}_{1,2}^1 \times \frac{\partial n_{1,2}^1}{\partial w_{1,1}^1} = \tilde{s}_{1,2}^1 \times a_{1,2}^0 = (-8)(2) = -16$$

$$[\mathbf{J}]_{2,2} = \frac{\partial v_2}{\partial x_2} = \frac{\partial e_{1,2}}{\partial b_1^1} = \frac{\partial e_{1,2}}{\partial n_{1,2}^1} \times \frac{\partial n_{1,2}^1}{\partial b_1^1} = \tilde{s}_{1,2}^1 \times \frac{\partial n_{1,2}^1}{\partial b_1^1} = \tilde{s}_{1,2}^1 = -8$$

$$[\mathbf{J}]_{2,3} = \frac{\partial v_2}{\partial x_3} = \frac{\partial e_{1,2}}{\partial n_{1,2}^2} \times \frac{\partial n_{1,2}^2}{\partial w_{1,1}^2} = \tilde{s}_{1,2}^2 \times \frac{\partial n_{1,2}^2}{\partial w_{1,1}^2} = \tilde{s}_{1,2}^2 \times a_{1,2}^1 = (-1)(4) = -4$$

$$[\mathbf{J}]_{2,4} = \frac{\partial v_2}{\partial x_4} = \frac{\partial e_{1,2}}{\partial b_1^2} = \frac{\partial e_{1,2}}{\partial n_{1,2}^2} \times \frac{\partial n_{1,2}^2}{\partial b_1^2} = \tilde{s}_{1,2}^2 \times \frac{\partial n_{1,2}^2}{\partial b_1^2} = \tilde{s}_{1,2}^2 = -1$$

所以雅可比矩阵为

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} -4 & -4 & -1 & -1 \\ -16 & -8 & -4 & -1 \end{bmatrix} \qquad \boxed{12\text{-}45}$$

## 12.5 结束语

基本 BP 算法(最速下降 BP 算法——SDBP)的一个主要问题是训练时间长。对于某些问题 SDBP 在大型机上也要花费数星期进行训练，因此并不适合于实际问题。由于 BP 算法是最先流行的算法，所以有许多提高算法收敛性能的改进。本章中讨论了 SDBP 算法收敛速度慢的原因，并介绍了几种提高算法性能的技术。

加速收敛的技术分成两类：启发式方法和标准的数值优化方法。我们讨论了两种启发式方法：动量方法(MOBP)和改变学习速度方法(VLBP)。MOBP 易于实现，并可以用批处理或增量处理模式，并且它的速度明显快于 SDBP。它需要选择动量系数，但 $\gamma$ 的取值范围限于[0，1]内，并且算法对它的选择并不敏感。

VLBP 方法比 MOBP 快，但只能用批处理方式。所以，它需要更多的存储空间。VLBP 需要选择 5 个参数，算法是相当鲁棒的，但参数的选择能影响收敛速度，并且是与实际问题相关的。

另外还介绍了两种标准的数值优化技术：共轭梯度法(CGBP)和 Levenberg-Marquardt 方法(LMBP)。CGBP 一般快于 VLBP。这是一种批处理方法，在每次迭代时要进行线性搜索，但它的存储需求与 VLBP 相仿。共轭梯度法还有许多用于神经网络应用的变化，我们只介绍了一种。

即使 LMBP 在每次迭代的时候都要求矩阵的逆，它还是所讨论过的中等规模的多层神经网络训练算法中最快的一种。它需要选择两个参数，但算法对参数的选择关并不敏感。LMBP 的主要缺点是存储需求大，需要求 $\mathbf{J}^T\mathbf{J}$ 矩阵的逆，而该矩阵是 $n \times n$ 的，其中 $n$ 是网络中权值和偏置值的总数。如果神经网络中的参数多于几千个，LMBP 在当前机器上就无法实现。

BP 算法还有许多其他变型，本章都没有讨论到。关于其他技术的某些参考文献在第 19 章给出。 12-46

## 参考文献

[Barn92] E. Barnard, "Optimization for training neural nets," *IEEE Trans. on Neural Networks*, vol. 3, no. 2, pp. 232 - 240, 1992.

这篇文章中讨论了许多有前途的神经网络训练的优化算法。

[Batt92] R. Battiti, "First-and second-order methods for learning : Between steepest descent and Newton's method," *Neural Computation*, vol. 4, no. 2, pp. 141 - 166, 1992.

这篇文章很好地总结了当前适合于神经网络训练的优化算法。

[Char92] C. Charalambous, "Conjugate gradient algorithm for efficient training of artificial neural networks," *IEE Proceedings*, vol. 139, no. 3, pp. 301 - 310, 1992.

本文说明了共轭梯度算法如何被用于训练多层神经网络，并且与其他训练算法作了比较。

[Fahl88] S. E. Fahlman, "Faster-learning variations on back-propagation : An empirical study," In D. Touretsky, G. Hinton &T. Sejnowski, eds., *Proceedings of the 1988 Connectionist Models Summer School*, San Mateo, CA: Morgan Kaufmann, pp. 38 - 51, 1988.

这篇文章介绍了 QuickProp 算法。这是对 BP 算法更常用的启发式变型中的一种。它假定误差曲线可以用抛物线逼近，同时每个权值可以看成是独立的。在许多问题上，该算法比标准的 BP 算法能够大大提高速度

[HaMe94] M. T. Hagan and M. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Tansactions on Neural Networks*, vol. 5, no. 6, 1994.

这篇文章描述用 Levenberg-Marquardt 算法训练多层神经网络，并比较这种算法与可变学习速度的 BP 算法和共轭梯度算法的性能。Levenberg-Marquardt 算法比较快，
12-47 但是需要更多的存储空间。

[Jaco88] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks*, vol. 1, no. 4, pp. 295 - 308, 1988.

这是早期讨论使用可变学习速度的 BP 算法的另外一篇文章。主要介绍了一种称为 delta-bar-delta 的学习规则，其中每个网络参数在每次迭代中都有不同的学习速度。

[NgWi90] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," *Proceedings of the IJCNN*, vol. 3, pp. 21 - 26, July 1990.

介绍为 BP 算法设置初始权值和偏置值的过程。它用 S 型传输函数的形状和输入变量的范围决定权值的大小，然后用偏置值将 S 形函数的形状移到操作区域的中央。BP 算法的收敛性在这里得到了显著的改进。

[RiIr90] A. K. Rigler, J. M. Irvine and T. P. Vogl, "Rescaling of variables in back propagation learning," *Neural Networks*, vol. 3, no. 5, pp. 561 - 573, 1990.

本文注意到了 S 形函数的导数在靠近尾部的值很小。这意味着网络中与前面几层相关的梯度元素通常要比最后一层的小。因此放大梯度中的项达到与它们相等。

[Scal85] L. E. Scales, *Introduction to Non-Linear Optimization*. New York: Springer-Verlag, 1985.

这是 Scales 写的一本很好读的关于算法优化方面的书。该书强调优化的方法比存在定理和收敛性证明更重要。书中的算法都给出了直观意义上的解释。大部分的算法都给
12-48 出了伪代码。

[Shan90] D. F. Shanno, "Recent advances in numerical techniques for large-scale optimization," *Neural Networks for Control*, Miller, Sutton and Werbos, eds., Cambridge MA : MIT

Press, 1990.

这篇文章讨论了一些可用于神经网络训练的共轭梯度算法和拟牛顿优化算法。

[Toll90] T. Tollenaere, "SuperSAB : Fast adaptive back propagation with good scaling properties," *Neural Networks*, vol. 3, no. 5, pp. 561 - 573, 1990.

这篇文章展示了一种具有不同学习速度的 BP 算法，其中每个权值都用不同的学习速度。

[VoMa88] T. P. Vogl, J. K. Mangis, A. K. Zigler, W. T. Zink and D. L. Alkon, "Accelerating the convergence of the backpropagation method," *Biological Cybernetics.*, vol. 59, pp. 256 - 264, Sept. 1988.

这是首先引入启发式技术来加速 BP 算法收敛速度的文章之一。文中包含对批处理、动量和可变学习速度的讨论。

12-49

## 习题

**E12.1** 要训练图 12-24 中的网络，训练集为

$$\{(\mathbf{p}_1 = [-2]),(\mathbf{t}_1 = [0.8])\},\quad \{(\mathbf{p}_2 = [2]),(\mathbf{t}_2 = [1])\}$$

其中每对的出现是等可能的。

写一个 MATLAB 的 M-文件画出均方误差性能指数的轮廓线图。

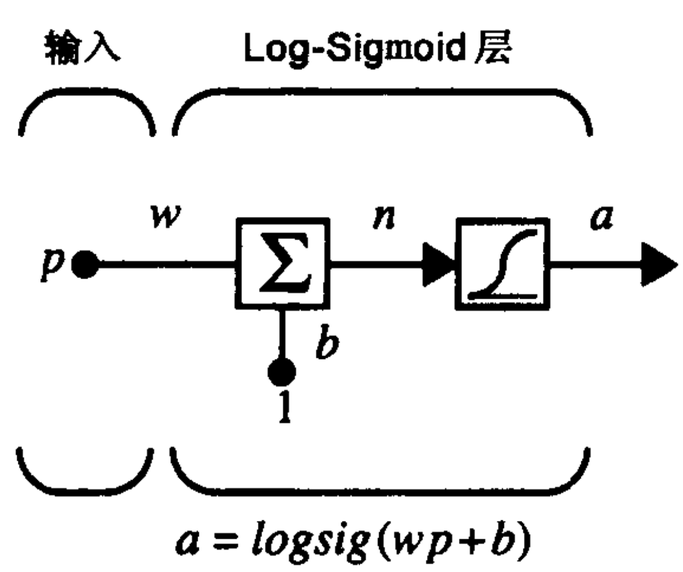



图 12-24　习题 E12.1 的网络

**E12.2** 用批处理模式和非批处理模式计算习题 E12.1 所述问题的初始步的方向，说明批处理模式的作用，初始条件为

$$w(0) = 0,\quad b(0) = 0.5$$

**E12.3** 回忆例题 P9.1 中的二次函数

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix}\mathbf{x} + [4 \quad 4]\mathbf{x}$$

用带动量的最速下降法求该函数极小点。

(i) 假设学习速度 $\alpha = 0.2$。求能使算法稳定的动量系数 $\gamma$(用例题 P12.2 中的思想)。

(ii) 假设学习速度，$\alpha = 20$。求能使算法稳定的动量系数 $\gamma$。

12-50

(iii) 写一个 MATLAB 程序，在 $F(\mathbf{x})$轮廓线图上画出(i)(ii)两种学习速度和动量系数的算法轨迹，初始值为

$$\mathbf{x}_0 = \begin{bmatrix} -1 \\ -2.5 \end{bmatrix}$$

**E12.4** 对习题 E12.3 的函数，执行 3 次可变学习速度算法的迭代过程，初始值为

$$\mathbf{x}_0 = \begin{bmatrix} -1 \\ -2.5 \end{bmatrix}$$

在 $F(\mathbf{x})$的轮廓线图上画出算法轨迹。算法的参数为 $\alpha = 0.4$，$\gamma = 0.1$，$\eta = 1.5$，$\rho = 0.5$，$\zeta = 5\%$。

**E12.5** 对习题 E12.3 的函数，执行一次共扼梯度法的迭代过程，初始值为

$$\mathbf{x}_0 = \begin{bmatrix} -1 \\ -2.5 \end{bmatrix}$$

在线性优化中，对区间定位用函数求值法，对区间缩小用黄金分割搜索法。在 $F(\mathbf{x})$轮廓线图上画出搜索路径。

**E12.6** 用图 12-25 的网络逼近函数

$$g(p) = 1 + \sin\left(\frac{\pi}{4}p\right), \ -2 \leq p \leq 2$$

初始网络参数选为

$$\mathbf{w}^1(0) = \begin{bmatrix} -0.27 \\ -0.41 \end{bmatrix}, \quad \mathbf{b}^1(0) = \begin{bmatrix} -0.48 \\ -0.13 \end{bmatrix}, \quad \mathbf{w}^2(0) = \begin{bmatrix} 0.09 \\ -0.17 \end{bmatrix}, \quad \mathbf{b}^2(0) = \begin{bmatrix} 0.48 \\ 0.13 \end{bmatrix}$$

试用在 $p = 0$ 和 $p = 1$ 的函数 $g(p)$建立训练集。计算 LMBP 算法中第一步的雅可比矩阵。(一些需要的信息在 11.2.3 节的例子中。)

12-51

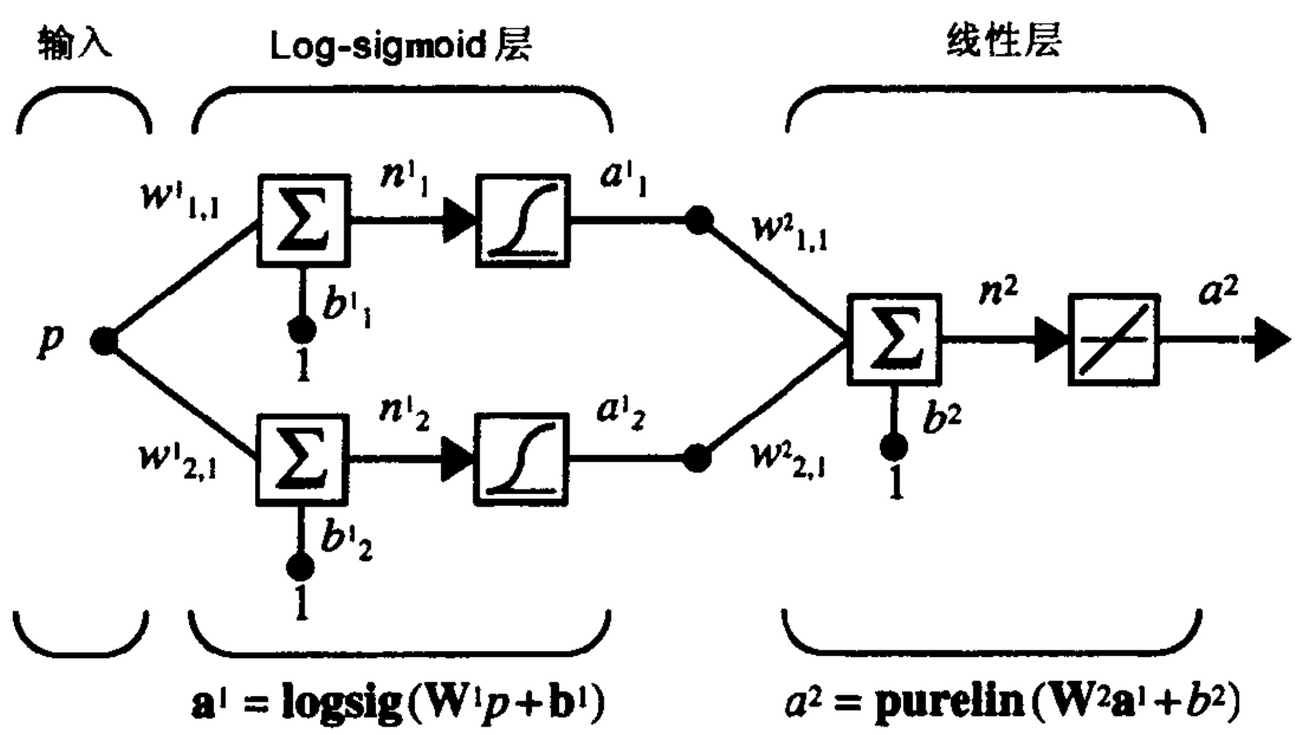



图 12-25 习题 E12.6 的网络

**E12.7** 对一线性网络，证明当 $\mu = 0$ 时 LMBP 算法在一次迭代内将收敛到最优解。

**E12.8** 在习题 E11.11 中你已用 MATLAB 编写了图 12-25 的 1-2-1 网络的 SDBP 训练算法，并且已经训练了逼的函数

$$g(p) = 1 + \sin\left(\frac{\pi}{8}p\right), \ -2 \leq p \leq 2$$

重做上述习题，改进你的程序以实现本章讨论过的算法：批处理模式的 SDBP，MOBP，VLBP，CGBP 和 LMBP。比较不同算法的收敛法果。

12-52

# 第13章 联想学习

## 13.1 目的

前面所讨论的神经网络（第4,7,10~12章)都是在监督模式下训练的。每个网络需要一个目标信号来定义正确的网络行为。

相反地，本章中介绍的一组简单规则允许无监督学习，这使网络具有在经常一同出现的模式之间学习其中关联的能力。一旦学习成功，关联能力将使网络能执行有用的任务，如模式识别和回忆。

尽管本章中的规则很简单，但它们是构成第14~16章中强大神经网络的基础。 13-1

## 13.2 理论和实例

本章是讨论联想的：联想是怎样在网络中表示的？网络怎样学习新的联想？

**刺激/响应** 什么是联想？联想是指系统中输入和输出之间的任何联系，其中当模式A输入到系统时，将产生模式B的反应。当两个模式关联时，输入模式被称为刺激(stimlus)。类似地，输出模式被称为响应(response)。

关联是很基本的概念，并构成了行为心理学派的基础。这个心理学的分支，试图利用联想和学习联想规则解释动物和人类的各种形为。

最早的形为心理学派的影响之一是巴甫洛夫的经典实验，他利用喂食时摇铃训练狗对铃声的反应，这是一个现在称为典型条件反射的例子。B. F. Skimer是最具影响的形为心理学派的支持者之一。他的经典实验包括训练老鼠按下一根棒以获得食物丸，这是一个用仪器作为条件的例子。

为了提供这些行为的生物学解释，Donald Hebb提出了他的假设(如在第7章所引用的)[Hebb49]：

"当细胞A的轴突触到细胞B的距离近到足够激励它，且反复地或持续地刺激B，那么在这两个细胞或一个细胞中会发生某种增长过程或代谢作用，增加A对细胞B的刺激效果。"

第7章中我们分析了基于Hebb律的有监督学习的性能。本章我们将讨论无监督的Hebb学习以及其他相关的联想学习规则。有许多学者对联想学习的发展作出了贡献，特别是Tuevo Kohonen，James Anderson和Stephen Grossberg都非常有影响。Anderson和Kohonen在20世纪60年代末和70年代初独立地提出了线性联想器网络([Ande72]，[Koho72])。Grossberg同时引入了非线性连续联想网络([Gross68])，这些学者与其他许多学者一起持续推动联想学习的发展直至今天。

本章中我们将讨论一些基本的联想学习规则。接着在第14~16章，要介绍一些将联想学习作为基本部件的复杂网络。第14章讲述Kohonen网络，第15章和第16章将讨论Grossberg网络。 13-2

### 13.2.1 简单联想网络

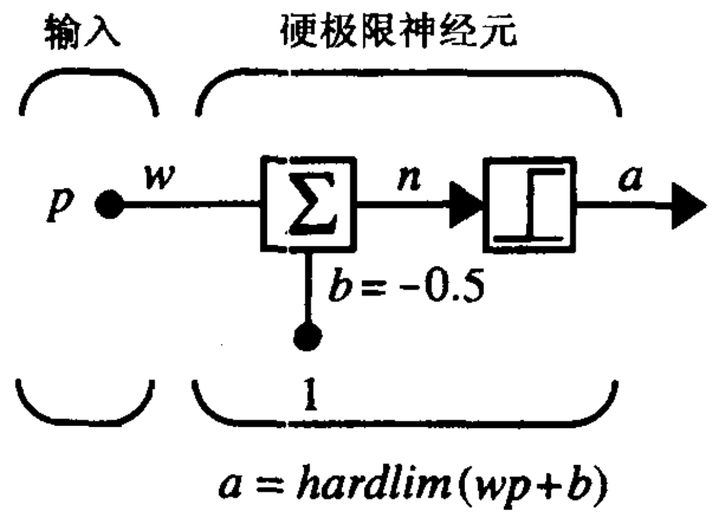



图 13-1 单输入硬极限联想器

让我们看一看可以实现一个联想的最简单的网络。一个单输入硬极限神经元如图 13-1 所示。

神经元的输出 $a$ 由输入 $p$ 按下式决定：

$$a = hardlim\ (wp + b) = hardlim\ (wp - 0.5) \tag{13.1}$$

为简化起见，这里限制 $p$ 的值为 1 或 0，以表示是否有刺激。注意到由于使用硬极限函数，$a$ 被限定为同样的值。它表明网络是否有响应。

$$p = \begin{cases} 1, \text{有刺激} \\ 0, \text{无刺激} \end{cases} \qquad a = \begin{cases} 1, \text{有响应} \\ 0, \text{无响应} \end{cases} \tag{13.2}$$

有刺激 $p = 1$ 和有响应 $a = 1$ 之间的联想是由 $w$ 的值决定的。仅当 $w$ 大于 $-b$(此例为 0.5)时，网络对刺激将有响应。

本章讨论的学习规则一般在大型网络的框架中使用，如第 14 ~ 16 章中的竞争网络。为了避免使用复杂网络来说明联想学习规则的操作，我们将使用有两类输入类型的简单网络。

**无条件刺激　条件刺激**　一类输入集称为无条件刺激，这类似于巴甫洛夫实验中给狗的食物。另一类输入集称为条件刺激，类似于巴甫洛夫实验中的铃声。一开始狗只在有食物时才分泌唾液，这是一种无需学习的先天的特征。然而当铃声与食物重复地同时出现时，狗会在仅有铃声的条件下分泌唾液，而不论是否有食物。

13-3

我们用 $\mathbf{p}^0$ 表示无条件刺激，用 $\mathbf{p}$ 表示条件刺激。首先假设关于 $\mathbf{p}^0$ 的权值固定，但与 $\mathbf{p}$ 有关的权值根据相关的学习规则改变。

图 13-2 表示一个识别香蕉的例子。网络中有无条件刺激(香蕉的形象)和有条件刺激(香蕉的气味)。这并不是暗示嗅觉比视觉更有条件性。本章的例子中对条件刺激和无条件刺激的选择是任意的，只是用以说明学习规则的性能。我们在下节中将用这个网络显示 Hebb 规则的操作过程。

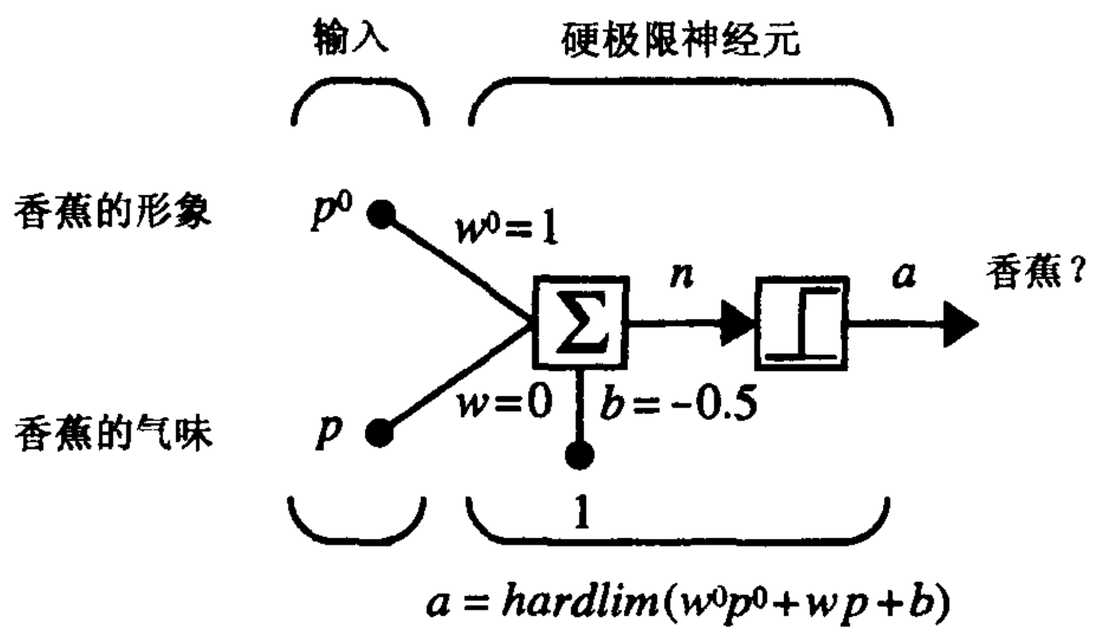



图 13-2 香蕉联想器

该网络中有条件和无条件输入的定义为：

$$p^0 = \begin{cases} 1, \text{检测形状} \\ 0, \text{不检测形状} \end{cases} \qquad p = \begin{cases} 1, \text{检测气味} \\ 0, \text{不检测气味} \end{cases} \tag{13.3}$$

此时我们希望网络用指明水果是香蕉的响应将香蕉的形状关联，而不是气味(请见图 13-3)。这可以将大于 $-b$ 的值赋与 $w^0$ 和将小于 $-b$ 的值赋于 $w$ 来解决这一问题。下述值满足这些要求：

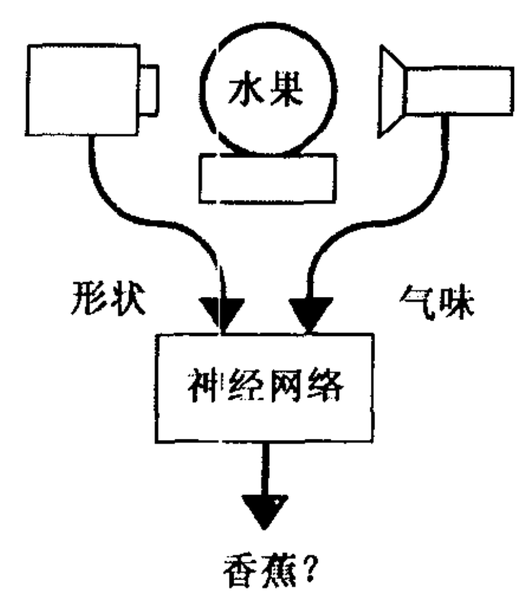
$$w^0 = 1, \quad w = 0 \tag{13.4}$$

香蕉联想器的输入/输出函数现简化为

$$a = hardlim(p^0 - 0.5) \tag{13.5}$$

所以，网络只对看到香蕉($p^0 = 1$)产生响应，而不论是否闻到香蕉的气味($p = 1$ 或 $p = 0$)。

在下一节中我们用这个网络说明一些联想学习规则的性能。




13-4

图 13-3

### 13.2.2 无监督的 Hebb 规则

对简单问题设计具有固定联想集的网络并不困难，但有用的网络必须能学习联想关系。

那么要在何时学习联想？一般来说，当几个事件同时发生的时候人和动物倾向于将它们联系起来。Hebb 规则表认为：当香蕉的气味刺激与香蕉概念响应(由其他刺激引起，如香蕉形状)同时产生的时候，网络将加强它们之间的联系。以后，当只有香蕉气味刺激时也能产生香蕉概念的响应。

无监督 Hebb 规则将根据神经元的输入 $p_j$ 和输出 $a_i$ 与它们的乘积之间的比例增加权值 $w_{ij}$：

$$w_{ij}(q) = w_{ij}(q-1) + \alpha a_i(q) p_j(q) \tag{13.6}$$

(也可参见式(7.5)。)学习速度 $\alpha$ 决定着联想关系建立前刺激和响应同时发生的次数。在图 13-2 的网络中，当 $w > -b = 0.5$ 时联想关系建立，因为此时 $p = 1$ 会产生响应 $a = 1$ 而不论 $p^0$ 的值是多少。

**局部学习** 注意到式(3.6)中只用到了包含被更新权值的层的信号。满足这个条件规则被称为局部学习规则。这与 BP 算法不同，例如 BP 算法中要将敏感度从最后一层反向传播。本章中介绍的规则都是局部学习规则。

无监督的 Hebb 规则也可以写成向量形式：

$$\mathbf{W}(q) = \mathbf{W}(q-1) + \alpha \mathbf{a}(q) \mathbf{p}^T(q) \tag{13.7}$$

**训练序列** 对所有的无监督规则，学习是由对一个按时间的输入序列(训练序列)的响应决定的：

$$\mathbf{p}(1), \quad \mathbf{p}(2), \quad \cdots, \quad \mathbf{p}(Q) \tag{13.8}$$

13-5

(注意：使用记号 $\mathbf{p}(q)$ 代替 $\mathbf{p}_q$ 是为了强调输入的时序性质。)在每次迭代中，根据对输入 $\mathbf{p}$ 的响应计算输出 $\mathbf{a}$，接着权值 $\mathbf{W}$ 根据 Hebb 规则更新。

让我们将无监督 Hebb 规则应用于香蕉联想器。联想器的初始权值由前面的例子给定，所以它最初对形状响应，而不响应气味。

$$w^0 = 1, w(0) = 0 \tag{13.9}$$

联想器重复地受到香蕉的作用。然而网络的气味传感器很可靠，形状传感器却只间断地工作(在偶数步)。所以，训练序列将重复下述两组输入：

$$\{p^0(1) = 0, \quad p(1) = 1\}, \quad \{p^0(2) = 1, \quad p(2) = 1\}, \cdots \tag{13.10}$$

第一个权值 $w^0$(表示无条件刺激 $p^0$ 的权值)保持为常数，而 $w$ 在每次迭代时，根据无监督 Hebb 规则更新，其中学习速度为 1。

$$w(q) = w(q-1) + a(q)p(q) \tag{13.11}$$

第一次迭代($q=1$)的输出为

$$\begin{aligned} a(1) &= hardlim(w^0 p^0(1) + w(0)p(1) - 0.5) \\ &= hardlim(1 \times 0 + 0 \times 1 - 0.5) = 0 \quad (\text{无响应}) \end{aligned} \tag{13.12}$$

单独的气味并不产生响应。无响应时，Hebb 规则并不改变 $w$。

$$w(1) = w(0) + a(1)p(1) = 0 + 0 \times 1 = 0 \tag{13.13}$$

第二次迭代时，香蕉的形状和气味都被检测到，网络的响应如下：

$$\begin{aligned} a(2) &= hardlim(w^0 p^0(2) + w(1)p(2) - 0.5) \\ &= hardlim(1 \times 1 + 0 \times 1 - 0.5) = 1 \quad (\text{香蕉}) \end{aligned} \tag{13.14}$$

因为气味刺激和响应同时发生，Hebb 规则将增加它们之间的权值。

13-6

$$w(2) = w(1) + a(2)p(2) = 0 + 1 \times 1 = 1 \tag{13.15}$$

第三次迭代中，视觉检测器再次失败，网络依然响应。此时已产生了香蕉气味和对它的反应之间的有用联想。

$$\begin{aligned} a(3) &= hardlim(w^0 p^0(3) + w(2)p(3) - 0.5) \\ &= hardlim(1 \times 0 + 1 \times 1 - 0.5) = 1 \quad (\text{香蕉}) \end{aligned} \tag{13.16}$$

$$w(3) = w(2) + a(3)p(3) = 1 + 1 \times 1 = 2 \tag{13.17}$$

此时，网络已能对香蕉的形状和气味的检测都作出响应。甚至在一个检测系统出现故障的时候，网络在大部分也能正常工作。

试验无监督的 Hebb 规则请用 *the Neural Network Design Demonstration* ***Unsupervised Hebb Rule*** (**nnd13uh**)。

我们看到 Hebb 规则能学习有用的联想。然而式(13.6)中定义的 Hebb 规则有一些缺点。首先，当我们在上例中连续地提交输入并更新 $w$ 时，权值 $w$ 将趋于无限大，这与导出 Hebb 规则的生物系统矛盾。突触不能无限制地增大。

第二个问题是没有机制使权值递减。如果 Hebb 网络的输入或输出有噪声，每次权值都会增加(但是很缓慢)，直至最后网络对任何刺激都作出响应。

### 带衰减的 Hebb 规则

**衰减速度** 改进 Hebb 规则的一种方法是加入权值的衰减项(式(7.45))：

$$\begin{aligned} \mathbf{W}(q) &= \mathbf{W}(q-1) + \alpha \mathbf{a}(q)\mathbf{p}^T(q) - \gamma \mathbf{W}(q-1) \\ &= (1-\gamma)\mathbf{W}(q-1) + \alpha \mathbf{a}(q)\mathbf{p}^T(q) \end{aligned} \tag{13.18}$$

其中 $\gamma$ 为衰减速度，是一个小于 1 的正数。当 $\gamma$ 趋近于 0 时，学习规则就变成了标准规则。当 $\gamma$ 趋近于 1 时，学习只能记住当前的模式而很快忘了前面的输入。这保证权值矩阵无界地增加。(过滤权值改变的思想在第 12 章中已讨论过，在那里称为动量。)

最大权值 $w_{ij}^{max}$ 由 $\gamma$ 决定。在式(13.18)的标量形式中，对所有 $q$ 将 $a_i$ 和 $p_j$ 都设为 1(最大化学习)，然后求解稳态权值(此时，新旧权值相同)，就可以求出这个最大权值。

13-7

$$\begin{aligned} w_{ij} &= (1-\gamma)w_{ij} + \alpha a_i p_j \\ w_{ij} &= (1-\gamma)w_{ij} + \alpha \\ w_{ij} &= \frac{\alpha}{\gamma} \end{aligned} \tag{13.19}$$

让我们来检验上述香蕉联想器问题中带衰试的 Hebb 规则。设衰减率 $\gamma$ 为 0.1。第一次迭代只有气味刺激，结果与前面相同：

$$a(1) = 0(\text{无响应}), \quad w(1) = 0 \tag{13.20}$$

第二次迭代也有相同的结果，此时两种刺激都产生了，并且网络对形状刺激产生响应。此时气味刺激和响应的同时出现产生了新的联想：

$$a(2) = 1(\text{香蕉}), \quad w(2) = 1 \tag{13.21}$$

第三次迭代的情况有所不同。网络已经学习了对气味的响应，权值也持续增加。然而，这次权值的增加仅为 0.9 而非 1.0。

$$w(3) = w(2) + a(3)p(3) - 0.1\ w(2) = 1 + 1 \times 1 - 0.1 \times 1 = 1.9 \tag{13.22}$$

由于衰减项限制权值的值，使得无论怎样多次强制联想，$w$ 也不会超过 $w_{ij}$。

$$w_{ij}^{max} = \frac{\alpha}{\gamma} = \frac{1}{0.1} = 10 \tag{13.23}$$

新规则也保证网络已学习的联想不会成为人为的噪声。任何小的随机增加将很快地衰减掉。

图 13-4 显示了香蕉识别例子中，有衰减和无衰减的 Hebb 规则的响应。在无衰减时，权值以神经元每次激活时同样的值连续增加。当增加衰减后，其权值以指数方式逼近最大值（$w_{ij}^{max} = 10$）。

试验带衰减的 Hebb 规则请用 *Neural Network Design Demonstration* ***Hebb with Decay*** (**nnd13hd**)和 ***Effect of Decay Rate*** (**nnd13edr**)。 13-8

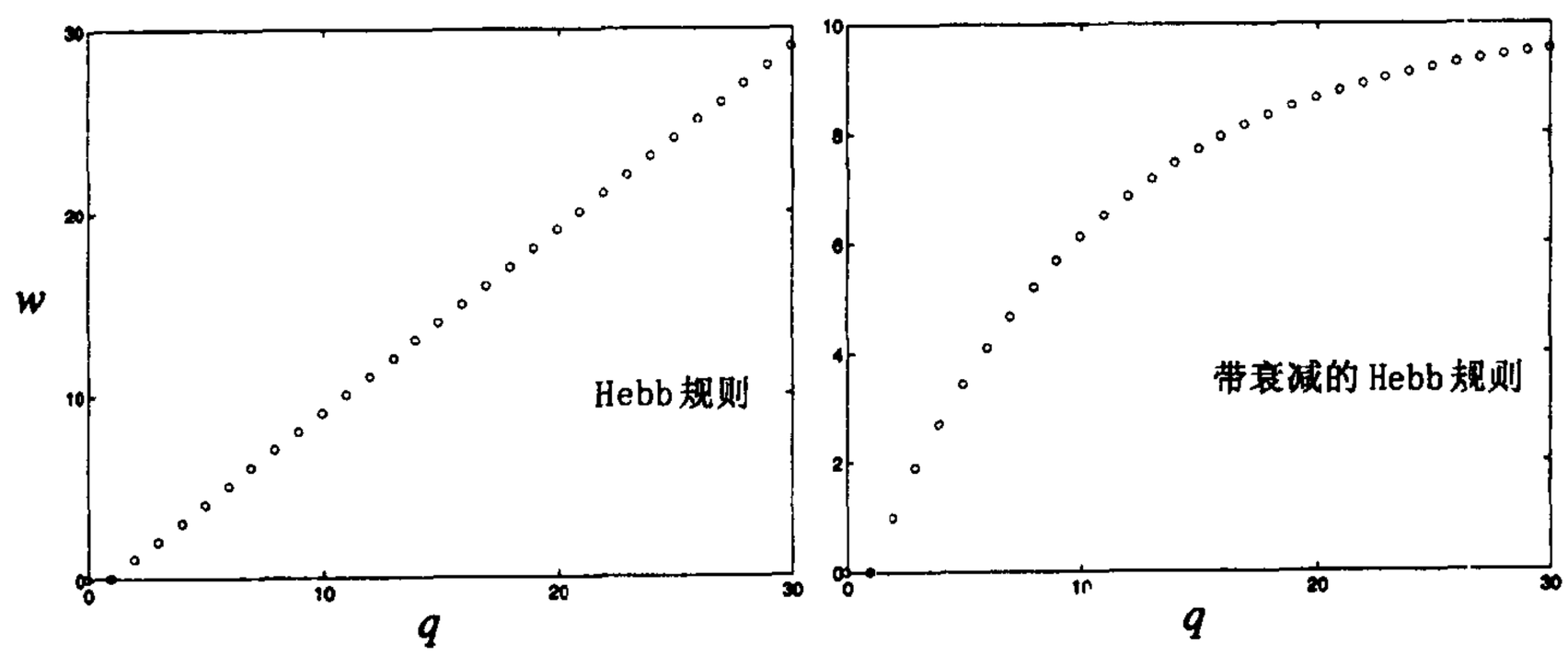



图 13-4 带衰减的和不带衰减的 Hebb 规则的响应

带衰减的 Hebb 规则解决了大权值的问题，然而这是有代价的。环境必须考虑到有时会出现具有联想的所有刺激，否则的话，联想将衰减。

为了说明这种情况，考虑 $a_i = 0$ 时的式(13.18)：

$$w_{ij}(q) = (1 - \gamma)w_{ij}(q - 1) \tag{13.24}$$

如果 $\gamma = 0.1$，它变为

$$w_{ij}(q) = (0.9)w_{ij}(q - 1) \tag{13.25}$$

因此 $w_{ij}$ 在每次 $a_i = 0$ 时将减少 10%。任何已学习的联想，最终将丢失。我们在下节中将讨论这个问题的一种解决方案。

### 13.2.3 简单的识别网络

**instar** 前面我们讨论的仅限于标量输入/输出之间的联想。这里将检验有向量输入的神经元(见图 13-5)。这个神经元有时被称为 instar，是最简单的模式识别网络，我们将简单地说明。

13-9

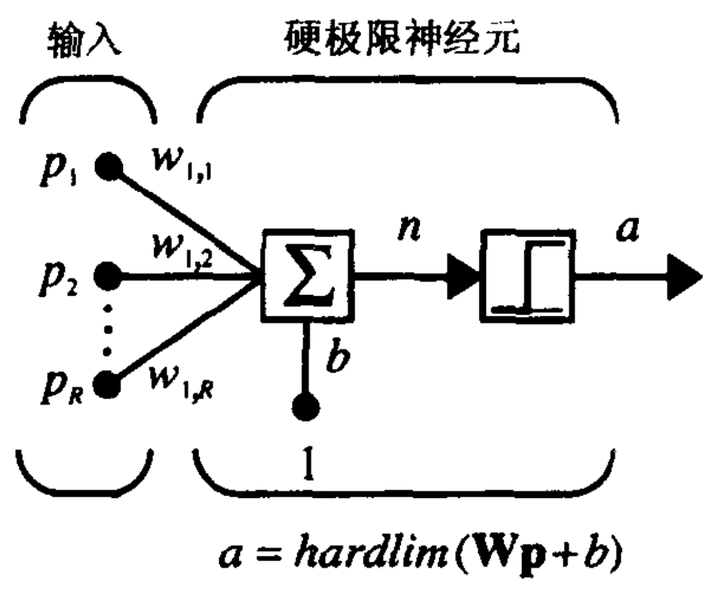



图 13-5 instar 神经元

我们注意到图 13-5 的 instar 神经元与图 4-3 的感知机(图 10-2 的 ADALINE 以及图 7-5 的线性联想器)的相似性。给这些网络以不同的名字，部分是因为历史的原因(因为它们产生于不同的时期和环境)，并且因为它们有不同的功能，以及用不同的方法分析。例如，虽然判定边界是感知机的重要概念，但是并不在 instar 中直接考虑。相反，我们将分析 instar 进行模式识别的能力，这类似于 Hamming 网络的第一层神经元(参见 3.2.3 节)。

instar 的输入/输出表达式为

$$a = hardlim(\mathbf{Wp} + b) = hardlim({}_1\mathbf{w}^T\mathbf{p} + b) \tag{13.26}$$

instar 神经元在输向量和权值向量的内积大于等于 $-b$ 时成为活跃的：

$${}_1\mathbf{w}^T\mathbf{p} \geq -b \tag{13.27}$$

从 3.2.3 节关于 Hamming 网络的讨论可以知道，对两个定长的向量，它们的内积在其方向相同时最大。也可以使用式(5.15)表示这一点：

$${}_1\mathbf{w}^T\mathbf{p} = \|{}_1\mathbf{w}\| \|\mathbf{p}\| \cos\theta \geq -b \tag{13.28}$$

其中 $\theta$ 是两个向量的夹角。易见当 $\theta=0$ 时，内积最大。如果 $\mathbf{p}$ 和 ${}_1\mathbf{w}$ 有相同的长度($\|\mathbf{p}\| = \|{}_1\mathbf{w}\|$)，则内积在 $\mathbf{p} = {}_1\mathbf{w}$ 时达到最大。

基于上述讨论，图 13-5 的 instar 神经元在 $\mathbf{p}$"接近"于 ${}_1\mathbf{w}$ 时将是活跃的。设置合适的偏置值 $b$，就可以选择输入向量和权值向量的接近程度，使 instar 神经元被激活。

13-10

如果设

$$b = -\|{}_1\mathbf{w}\| \|\mathbf{p}\| \tag{13.29}$$

则 instar 神经元只有在 $\mathbf{p}$ 的方向精确等于 ${}_1\mathbf{w}$ 的方向($\theta=0$)时才活跃。因而，我们就有了一个只能识别模式 ${}_1\mathbf{w}$ 的神经元。

如果想让 instar 能响应任何接近 ${}_1\mathbf{w}$($\theta$ 很小)的模式，那么可以增加 $b$ 到大于 $-\|{}_1\mathbf{w}\| \|\mathbf{p}\|$ 的值。$b$ 值越大，就有越多的模式能激活 instar 神经元，也就使它具有更小的分辨率。

应该注意，这里的分析假设所有的输入向量都有相同的长度(模)。我们将在第 14 ~ 16 章中重新考察规格化问题。

如果我们知道要识别那个向量，现在可以设计出一个向量识别网络。然而，如果网络是无监督地学习一个向量，将需要一个新的规则，因为 Hebb 规则的任何形式都不产生规格化的权值。

### 13.2.4 instar 规则

带衰减的 Hebb 规则的一个问题是要求刺激不断重复，否则联想就会丢失。一个更好的规则可能只在 instar 神经元是活跃时允许权值衰减。这样权值仍被限制，但遗忘被减到最小。再次考虑 Hebb 规则的原始形式。

$$w_{ij}(q) = w_{ij}(q-1) + \alpha a_i(q) p_j(q) \tag{13.30}$$

**instar 规则** 为了在获得权值衰减的同时限制遗忘问题，可以加上一个与 $a_i(q)$ 成比例的衰减项：

$$w_{ij}(q) = w_{ij}(q-1) + \alpha a_i(q) p_j(q) - \gamma a_i(q) w_{ij}(q-1) \tag{13.31}$$

可以设置 $\gamma$ 等于 $\alpha$ 来简化(13.31)式(这样新权值的学习速度与旧值的衰减速度相同)，并合并同类项

$$w_{ij}(q) = w_{ij}(q-1) + \alpha a_i(q)(p_j(q) - w_{ij}(q-1)) \tag{13.32}$$

该等式称为 instar 规则，写成向量形式：

$${}_i\mathbf{w}(q) = {}_i\mathbf{w}(q-1) + \alpha a_i(q)(p(q) - {}_i\mathbf{w}(q-1)) \tag{13.33}$$

如果考虑 instar 活跃($a_i = 1$)的情况，就可以很好地理解 instar 规则的性能特点。式(13.33)可以写成 13-11

$$\begin{aligned} {}_i\mathbf{w}(q) &= {}_i\mathbf{w}(q-1) + \alpha(\mathbf{p}(q) - {}_i\mathbf{w}(q-1)) \\ &= (1-\alpha)\,{}_i\mathbf{w}(q-1) + \alpha\mathbf{p}(q) \end{aligned} \tag{13.34}$$

该运算可以由图 13-6 表示。

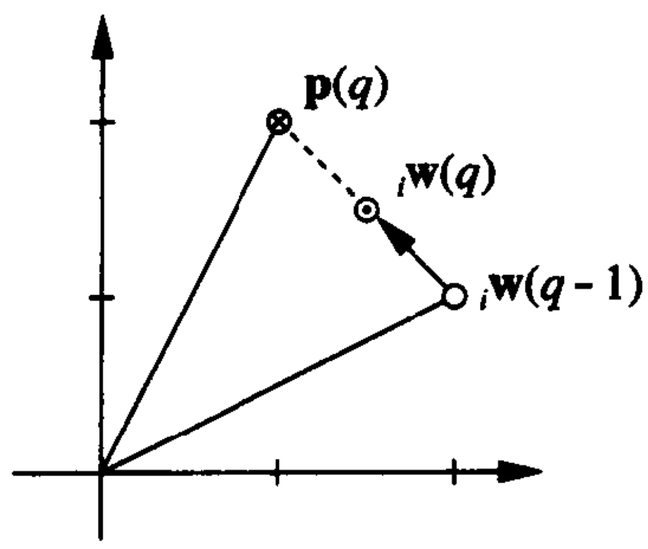



图 13-6 instar 规则的图形表示

当 instar 神经元活跃的时候，权值向量沿着旧的权值向量和输入向量连线向输入向量方向移动，权值向量移动的距离决定于 $\alpha$ 的值。当 $\alpha = 0$ 时，新的权值向量等于旧的权值向量(没有移动)。当 $\alpha = 1$ 时，新的权值向量等于输入向量(最大移动)。如果 $\alpha = 0.5$，则新的权值向量移动到旧权值向量和输入向量正中间。

instar 规则的一个有用特性是如果输入向量是规格化的，则一旦 ${}_i\mathbf{w}$ 学习了一个特定的向量 $\mathbf{p}$ 后也会成为规格化的。可以发现这个规则不仅能使遗忘减到最小程度，并且在输入向量是规格化时使权值向量也是规格化的。

将 instar 规则应用于图 13-8 的网络中。由图 13-6，它有两个输入：一个表示水果是在视觉上是否作为橘子识别(无条件刺激)；另一个包含了水果的其他三种度量(条件刺激)。

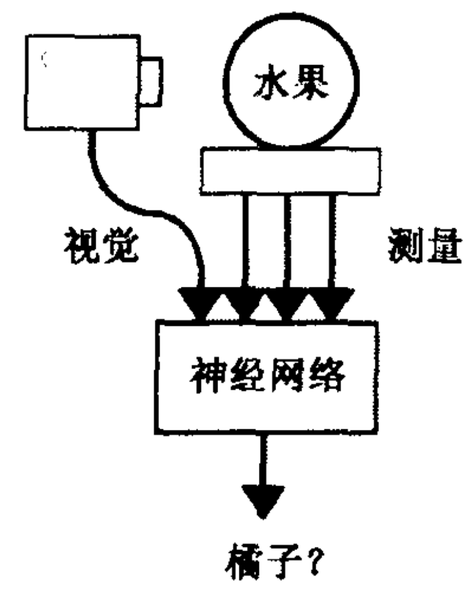



图 13-7

网络的输出为

$$a = hardlim(w^0 p^0 + \mathbf{W}\mathbf{p} + b) \tag{13.35}$$

输入 $\mathbf{p}$ 的元素被限制为 ±1(如第 3 章式(3.2)所定义)。这个限制

13-12 保证 $\mathbf{p}$ 是规格化向量，其长度为 $\|\mathbf{p}\| = \sqrt{3}$。$p^0$ 和 $\mathbf{p}$ 的定义为

$$p^0 = \begin{cases} 1, \text{视觉上探测到橘子} \\ 0, \text{没有探测到橘子} \end{cases} \qquad \mathbf{p} = \begin{bmatrix} \text{形状} \\ \text{质地} \\ \text{重量} \end{bmatrix} \tag{13.36}$$

偏置值 $b$ 为 $-2$，比 $-\|\mathbf{p}^2\| = -3$ 稍大一些(见式(13.29))。

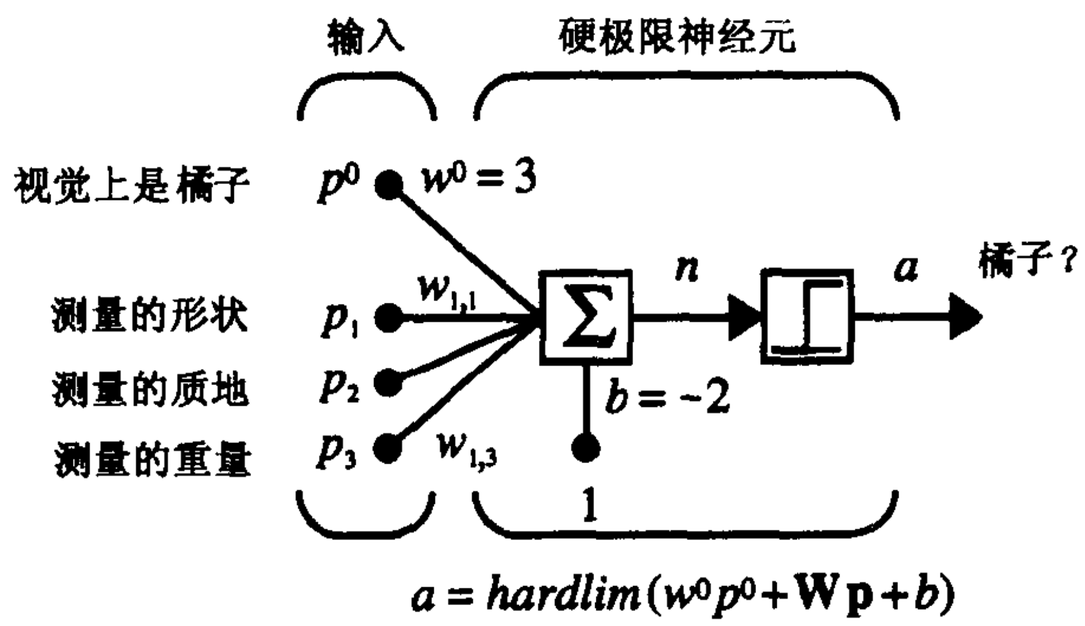



图 13-8 橘子识别器

我们希望网络在橘子的视觉和响应之间有一个固定的联想，故可以设置 $w^0$ 大于 $-b$。但一开始，网络不应该对任何水果测量值的组合作出响应，所以测量权值设置为全 0。

$$w^0 = 3, \quad \mathbf{W}(0) = {}_1\mathbf{w}^T(0) = [0\ 0\ 0] \tag{13.37}$$

测量权值由 instar 规则更新，其中学习速度 $\alpha = 1$。

$${}_1\mathbf{w}(q) = {}_1\mathbf{w}(q-1) + a(q)(\mathbf{p}(q) - {}_1\mathbf{w}(q-1)) \tag{13.38}$$

训练序列包括了重复出现的橘子信号，测量值每次都给出。但为了说明 instar 规则的操作，我们假设视觉系统由于构造上的问题只在偶数步骤运行正常。

$$\left\{ p^0(1) = 0, \mathbf{p}(1) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \right\}, \left\{ p^0(2) = 1, \mathbf{p}(2) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \right\}, \cdots \tag{13.39}$$

13-13 由于 $\mathbf{W}$ 初始化为余 0，故 instar 神经元在第一次迭代的时候并不响应橘子的测量值。

$$a(1) = hardlim(w^0 p^0(1) + \mathbf{W}\mathbf{p}(1) - 2)$$

$$a(1) = hardlim\left(3 \times 0 + [0\ 0\ 0]\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - 2\right) = 0 \quad (\text{无响应}) \tag{13.40}$$

由于神经元没有响应，权值 ${}_1\mathbf{w}$ 并不由 instar 改变。

$$
\begin{aligned}
{}_1\mathbf{w}(1) &= {}_1\mathbf{w}(0) + a(1)(\mathbf{p}(1) - {}_1\mathbf{w}(0)) \\
&= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + 0\left(\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}
\end{aligned} \tag{13.41}
$$

但在第二次迭代中，除对橘子测量之外，神经元对橘子的视觉刺激产生了响应。

$$
\begin{aligned}
a(2) &= hardlim(w^0 p^0(2) + \mathbf{W}\mathbf{p}(2) - 2) \\
&= hardlim\left(3 \times 1 + [0\ 0\ 0]\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - 2\right) = 1 \quad (\text{橘子})
\end{aligned} \tag{13.42}
$$

结果神经元学习到橘子的测量向量和响应之间的联想。权值向量${}_1\mathbf{w}$成了橘子测量向量的拷贝。

$$
\begin{aligned}
{}_1\mathbf{w}(2) &= {}_1\mathbf{w}(1) + a(2)(\mathbf{p}(2) - {}_1\mathbf{w}(1)) \\
&= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + 1\left(\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}
\end{aligned} \tag{13.43}
$$

网络现在可以根据橘子的测量值来识别橘子了。在第三次迭代中即使当视觉检测系统再次失效时，神经元依然产生响应。 13-14

$$
\begin{aligned}
a(3) &= hardlim(w^0 p^0(3) + \mathbf{W}\mathbf{p}(3) - 2) \\
a(3) &= hardlim\left(3 \times 0 + [1\ -1\ -1]\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - 2\right) = 1 \quad (\text{橘子})
\end{aligned} \tag{13.44}
$$

在完全学习了测量后，权值向量停止了改变。(低的学习速度将需要更多的迭代次数。)

$$
\begin{aligned}
{}_1\mathbf{w}(3) &= {}_1\mathbf{w}(2) + a(3)(\mathbf{p}(3) - {}_1\mathbf{w}(2)) \\
&= \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + 1\left(\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}
\end{aligned} \tag{13.45}
$$

此时网络在视觉检测系统失效的情况下，也能根据测量值识别橘子。

试验 instar 规则请用 *Neural Design Demonstration Instar* (**nnd13is**)和 *Graphical Instar* (**nnd13gis**)。

Kohonen 规则

此时可以引入另一种与 instar 规则相关的联想学习规则，即 Kohonen 规则：

$$
{}_i\mathbf{w}(q) = {}_i\mathbf{w}(q-1) + \alpha(\mathbf{p}(q) - {}_i\mathbf{w}(q-1)), \quad i \in X(q) \tag{13.46}
$$

类似于 instar 规则，Kohonen 规则允许神经元的权值学习输入向量，因此适合于识别应用。与 instar 规则不同的是学习并不正比于神经元的输出 $a_i(q)$。它的学习发生在神经元的下标 $i$ 是集合 $X(q)$的元素时。

如果 instar 规则应用于仅返回值为 0 或 1 的传输函数(如硬极限函数)的一层神经元时，Kohonen 规则可以通过将 $X(q)$定义为满足 $a_i(q) = 1$ 的所有 $i$ 的集合，从而与 instar 规则等

价。Kohonen 规则的优点是也可以用于其他定义。它对于训练如像自组织特性映射这样的网络(第 14 章中介绍)是有用的。

13-15

### 13.2.5 简单回忆网络

**outstar** 前面已经看到 instar 网络(有一个向量输入和一个标量输出)可以利用将特定向量刺激与响应相联想来实现模式识别。图 13-9 中所示的 outstar 网有一个标量输入和一个向量输出。它可以利用一个刺激和向量响应之间的联想完成模式回忆。

这个网络的输入输出表达式是

$$\mathbf{a} = \mathbf{satlins}(\mathbf{W}p) \tag{13.47}$$

之所以选择对称饱和函数(**satlins**)是为了把网络用于回忆包含 1 或 -1 的向量。

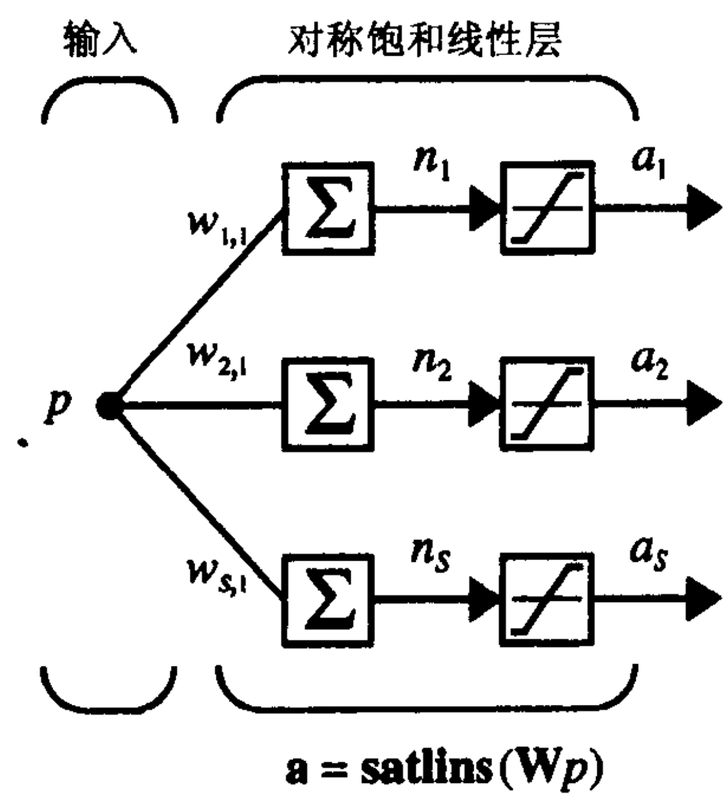



图 13-9 outstar 网络

如果我们希望网络将某种刺激(输入 1)和一个特定输出向量 $\mathbf{a}^*$ 联想，可以简单地设置 $\mathbf{W}$(它仅包含一个单列向量)等于 $\mathbf{a}^*$。这时如果 $p = 1$，输出将是 $\mathbf{a}^*$：

$$\mathbf{a} = \mathbf{satlins}(\mathbf{W}p) = \mathbf{satlins}(\mathbf{a}^* \cdot 1) = \mathbf{a}^* \tag{13.48}$$

(其中假设 $\mathbf{a}^*$ 的元素都是小于或等于 1 的。)

注意，我们通过把权值矩阵的一列设置为目标向量来构造一个回忆网络，而在前面则是通过设置权值矩阵的一行为目标向量来设计识别网络的。

现在可以设计一个回忆已知向量 $\mathbf{a}^*$ 的网络，但需要一个在无监督条件下学习向量的学习规则。我们将在下节中描述该学习规则。

13-16

### 13.2.6 outstar 规则

为了推导 instar 规则，遗忘由 Hebb 规则中的权值衰减项限制为与网络的输出 $a_i$ 成比例。相反，为了得到 outstar 学习规则，我们限制权值衰减项与网络输入 $p_j$ 成比例：

$$w_{ij}(q) = w_{ij}(q-1) + \alpha a_i(q) p_j(q) - \gamma p_j(q) w_{ij}(q-1) \tag{13.49}$$

如果设置衰减速度 $\gamma$ 等于学习速度 $\alpha$ 并合并同类项，有

$$w_{ij}(q) = w_{ij}(q-1) + \alpha(a_i(q) - w_{ij}(q-1)) p_j(q) \tag{13.50}$$

outstar 规则的特性类似于 instar 规则。学习发生在 $p_j$ 不等 0(代替 $a_i$)。当学习发生的时

候，列 $\mathbf{w}_j$ 向输出向量方向移动。

**outstar 规则** 正如 instar 规则，outstar 规则可以写成向量形式：

$$\mathbf{w}_j(q) = \mathbf{w}_j(q-1) + \alpha(\mathbf{a}(q) - \mathbf{w}_j(q-1))p_j(q) \tag{13.51}$$

这里 $\mathbf{w}_j$ 是矩阵 $\mathbf{W}$ 的第 $j$ 列。

为了检验 outstar 规则，我们将训练图 13-10 所示的网络。

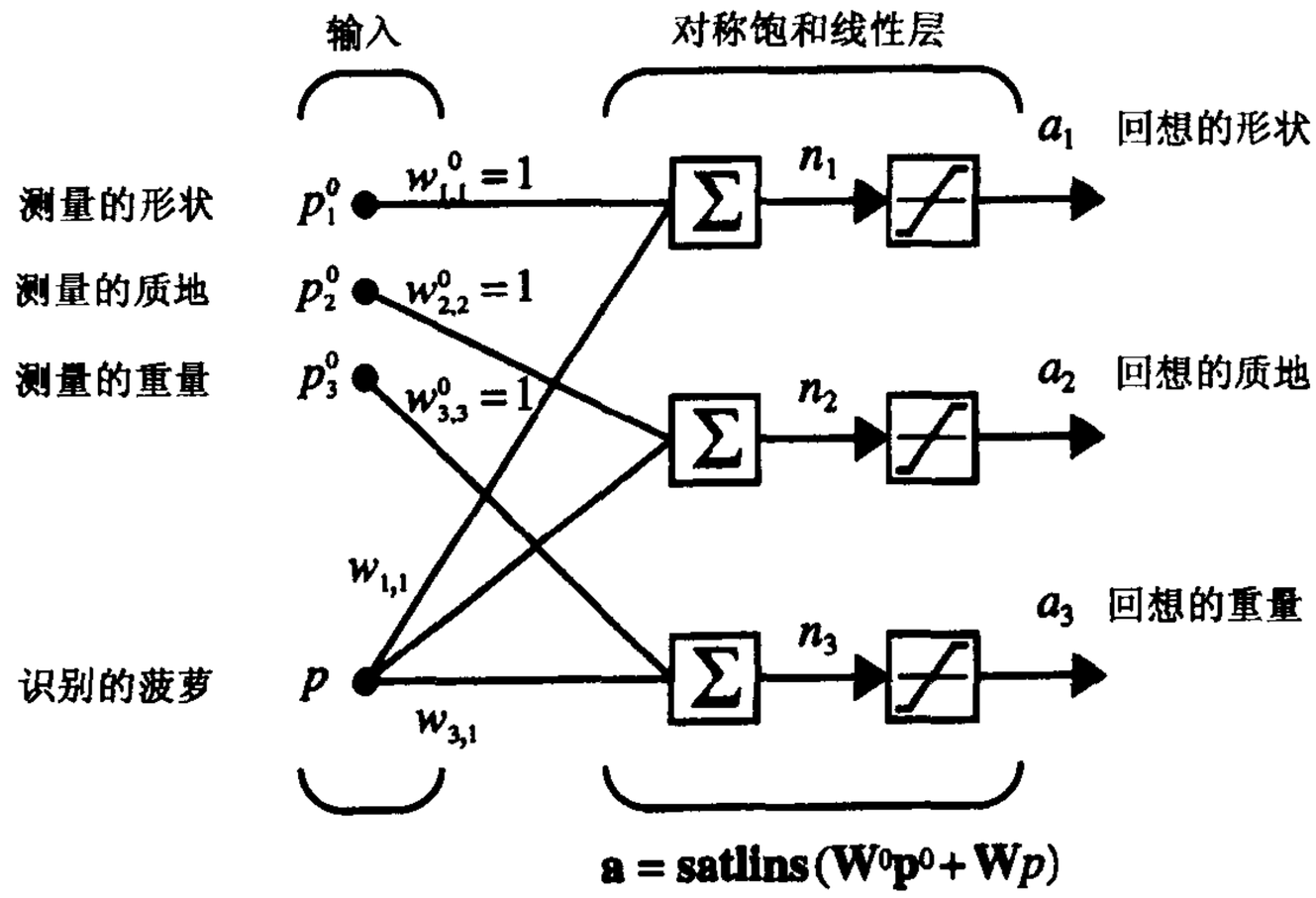



图 13-10 菠萝回忆器

13-17

网络输出由

$$\mathbf{a} = \mathbf{satlins}(\mathbf{W}^0\mathbf{p}^0 + \mathbf{W}p) \tag{13.52}$$

计算，其中

$$\mathbf{W}^0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{13.53}$$

由图 13-11，网络的两个输入提供了对水果的测量 $\mathbf{p}^0$(无条件刺激)，以及表示通过视觉确认菠萝的信号 $p$(条件刺激)。

$$\mathbf{p}^0 = \begin{bmatrix} \text{形状} \\ \text{质地} \\ \text{重量} \end{bmatrix}, p = \begin{cases} 1, \text{看见一个菠萝} \\ 0, \text{其他} \end{cases} \tag{13.54}$$

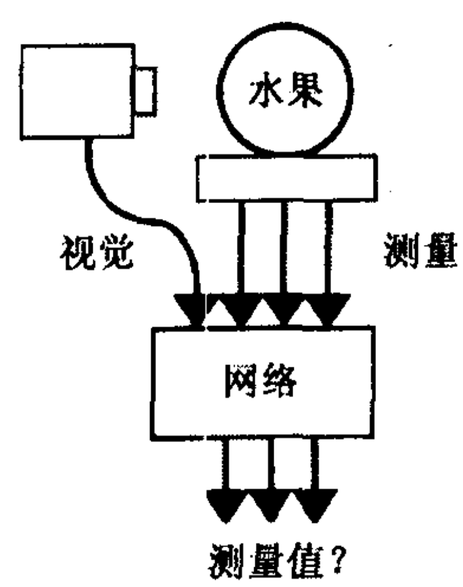



图 13-11

网络的输出反映了对当前被检验水果的测量值，所用的是无论什么可用的输入。

无条件刺激的权值矩阵 $\mathbf{W}^0$ 被设置成单位矩阵，所以任何测量值集合 $\mathbf{p}^0$(取 ±1 值)可以拷贝到输出 $\mathbf{a}$。条件刺激权值矩阵 $\mathbf{W}$ 一开始被设置为 0，这样 $p$ 为 1 时就不产生响应，而 $\mathbf{W}$ 用 outstar 规则进行更新，其中学习速度为 1：

$$\mathbf{w}_j(q) = \mathbf{w}_j(q-1) + (\mathbf{a}(q) - \mathbf{w}_j(q-1)p(q) \tag{13.55}$$

训练序列包括了对菠萝的视觉和测量的重复表示。菠萝的测量值为

$$\mathbf{p}^{菠萝} = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \tag{13.56}$$

但是由于测量系统的误差，测量值只在偶数次迭代时才有效。

13-18

$$\left\{\mathbf{p}^0(1) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, p(1) = 1\right\}, \left\{\mathbf{p}^0(2) = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}, p(2) = 1\right\}, \cdots \tag{13.57}$$

第一次迭代时看到了菠萝，但测量值不对。

$$\mathbf{a}(1) = \mathbf{satlins}(\mathbf{W}^0\mathbf{p}^0(1) + \mathbf{W}p(1)) \tag{13.58}$$

$$\mathbf{a}(1) = \mathbf{satlins}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}1\right) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (\text{无响应}) \tag{13.59}$$

网络看见了菠萝，但不能输出合适的测量值。这是因为它还没有学习到，且测量系统没有开始工作。更新后的权值保持不变。

$$\mathbf{w}_1(1) = \mathbf{w}_1(0) + (\mathbf{a}(1) - \mathbf{w}_1(0))p(1) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right)1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{13.60}$$

第二次迭代时，菠萝被看见，而且获得正确的测量值。

$$\mathbf{a}(2) = \mathbf{satlins}\left(\begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}1\right) = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \quad (\text{给出测量值}) \tag{13.61}$$

测量值是可用的，所以网络正确地输出这些测量值。权值更新如下：

$$\begin{aligned} \mathbf{w}_1(2) &= \mathbf{w}_1(1) + (\mathbf{a}(2) - \mathbf{w}_1(1))p(2) \\ &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \left(\begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right)1 = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \end{aligned} \tag{13.62}$$

因为可同时获得菠萝的视觉和测量值，因此网络形成了两者之间的联想。权值矩阵现在是测量值的拷贝，所以测量值在以后就可以回忆。 13-19

在第三次迭代中，测量值再次成为不可用的，但是输出为

$$\mathbf{a}(3) = \mathbf{satlins}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}1\right) = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \quad (\text{回忆的测量值}) \tag{13.63}$$

网络现在即使在测量系统失效时，也能回忆菠萝的测量值。从现在起，权值只有在菠萝被看到且具有不同的测量值时才会发生变化。

$$\begin{aligned} \mathbf{w}_1(3) &= \mathbf{w}_1(2) + (\mathbf{a}(2) - \mathbf{w}_1(2))p(2) \\ &= \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} + \left(\begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} - \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}\right)1 = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \end{aligned} \tag{13.64}$$

试验带衰减的 outstar 规则请用 *Neural Network Design Demonstration* **Outstar Rule**(**nnd13os**)

第 16 章中，我们将介绍 ART 网络，其中用到了 instar 规则和 outstar 规则。 13-20

## 13.3 小结

### 联想

联想是网络输入和输出之间的一种联系，即当某个刺激 A 提交网络后，将输出一个响应 B。

### 联想学习规则

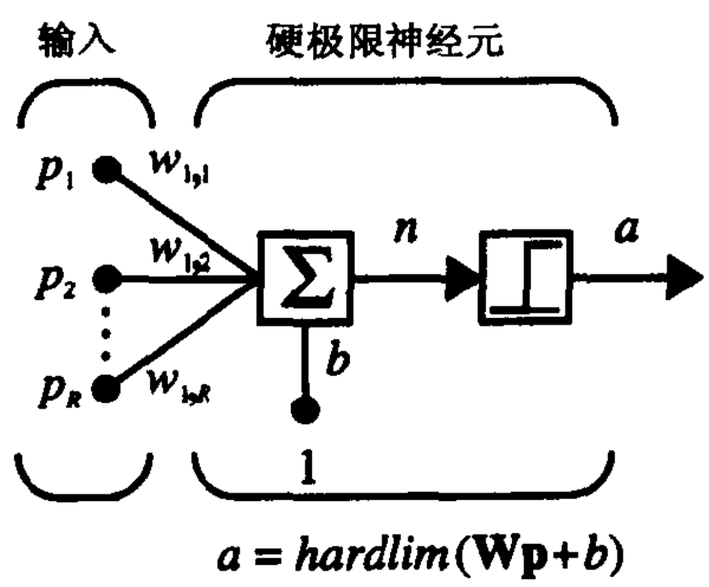
**无监督的 Hebb 规则**

$$\mathbf{W}(q) = \mathbf{W}(q-1) + \alpha\mathbf{a}(q)\mathbf{p}^T(q)$$

**带衰减的 Hebb 规则**

$$\mathbf{W}(q) = (1-\gamma)\mathbf{W}(q-1) + \alpha\mathbf{a}(q)\mathbf{p}^T(q)$$

**instar**



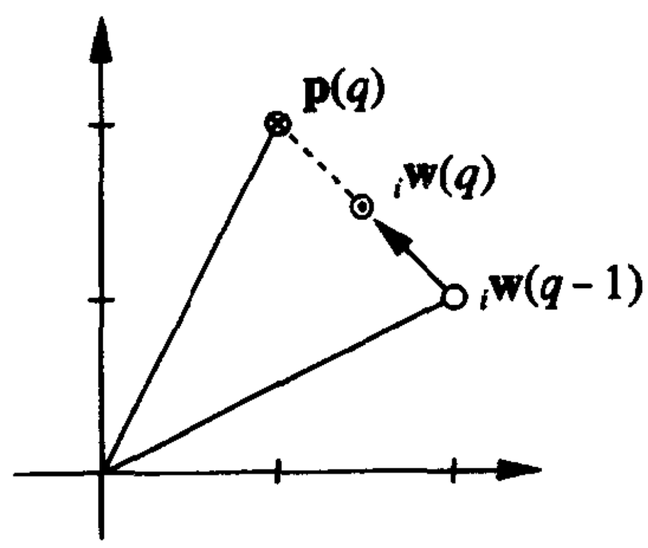

$$a = hardlim({}_1\mathbf{w}^T\mathbf{p} + b)$$

当 ${}_1\mathbf{w}^T\mathbf{p} = \|{}_1\mathbf{w}\|\,\|\mathbf{p}\|\cos\theta \geqslant -b$ 时 instar 神经元被激活，其中 $\theta$ 是 $\mathbf{p}$ 和 ${}_1\mathbf{w}$ 之间的夹角。

**instar 规则**

$${}_i\mathbf{w}(q) = {}_i\mathbf{w}(q-1) + \alpha a_i(q)(\mathbf{p}(q) - {}_i\mathbf{w}(q-1))$$

$${}_i\mathbf{w}(q) = (1-\alpha)\mathbf{w}(q-1) + \alpha\mathbf{p}(q), \quad 如果(a_i(q) = 1)$$ 13-21

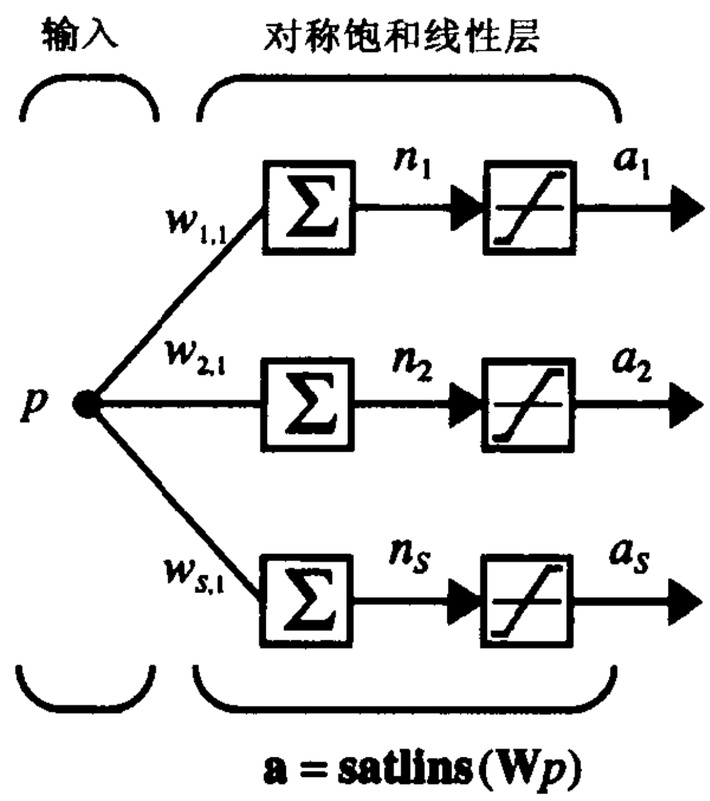



instar 规则的图形表示($a_i(q) = 1$)

**Kohonen 规则**

$$ {}_i\mathbf{w}(q) = {}_i\mathbf{w}(q-1) + \alpha(\mathbf{p}(q) - {}_i\mathbf{w}(q-1)), \quad i \in X(q) $$

**outstar**




**outstar 规则**

13-22

$$ \mathbf{w}_j(q) = \mathbf{w}_j(q-1) + \alpha(\mathbf{a}(q) - \mathbf{w}_j(q-1))p_j(q) $$

## 13.4 例题

**P13.1** 式(13.19)计算了带衰减的 Hebb 规则的最大权值，其中假设 $p_j$ 和 $a_i$ 在每个时间步都为 1。当 $p_j$ 和 $a_j$ 为 0 和 1 之间的值时，计算最大权值。

**解**

我们从带衰减的 Hebb 规则的标量形式开始

$$ w_{ij}(q+1) = (1-\gamma)w_{ij}(q-1) + \alpha a_i(q)p_j(q) $$

当权值在两个时间步更新时，可以用 $q$ 作为下标重写这个表达式两次。

$$ w_{ij}(q+1) = (1-\gamma)w_{ij}(q) + \alpha a_i(q)p_j(q) $$

$$ w_{ij}(q+2) = (1-\gamma)w_{ij}(q+1) + \alpha a_i(q+1)p_j(q+1) $$

将第一个式子代入第二个式子，得到 $w_{ij}$在两个时间步更新的单一表达式。

$$ w_{ij}(q+2) = (1-\gamma)((1-\gamma)w_{ij}(q) + \alpha a_i(q)p_j(q)) + \alpha a_i(q+1)p_j(q+1) $$

此时可以代入 $p_i$ 和 $a$ 的值。由于我们在计算最大权值，设 $p_j(q)$和 $a_i(q)$为 0，$p_j(q+1)$和 $a_i(q+1)$为 1。这意味着在第一步权值减少，而第二步权值增加，以保证 $w_{ij}(q+2)$为两个权值中的最大值。如果求解 $w_{ij}(q+2)$，有

$$ w_{ij}(q+2) = (1-\gamma)^2 w_{ij}(q) + \alpha $$

假设 $w_{ij}$最终将达到稳定状态，或可以设 $w_{ij}(q+2)$和 $w_{ij}(q)$都为 $w_{ij}^{max}$来求解，并解

$$ w_{ij}^{max} = (1-\gamma)^2 w_{ij}^{max} + \alpha $$

$$ w_{ij}^{max} = \frac{\alpha}{2\gamma - \gamma^2} $$

我们用 MATLAB 绘制这个关系图。这个图表示在相间 0.025 的各个区间上的学习速度和衰

减速度。

```
lr = 0:0.025:1;
dr = 0.025:0.025:1;
```

13-23

下面是建立最大权值的格网图的命令，最大权值是学习速度和衰减速度的函数。

```
[LR, DR] = meshgrid(dr, lr);
MW = LR ./ (DR .* (2 - DR));
mesh(DR, LR, MW);
```

图中显示了当衰减速度与学习速度 $\alpha$ 相比为很小的值时，$w_{ij}^{max}$ 趋向无限(见图 13-12)。

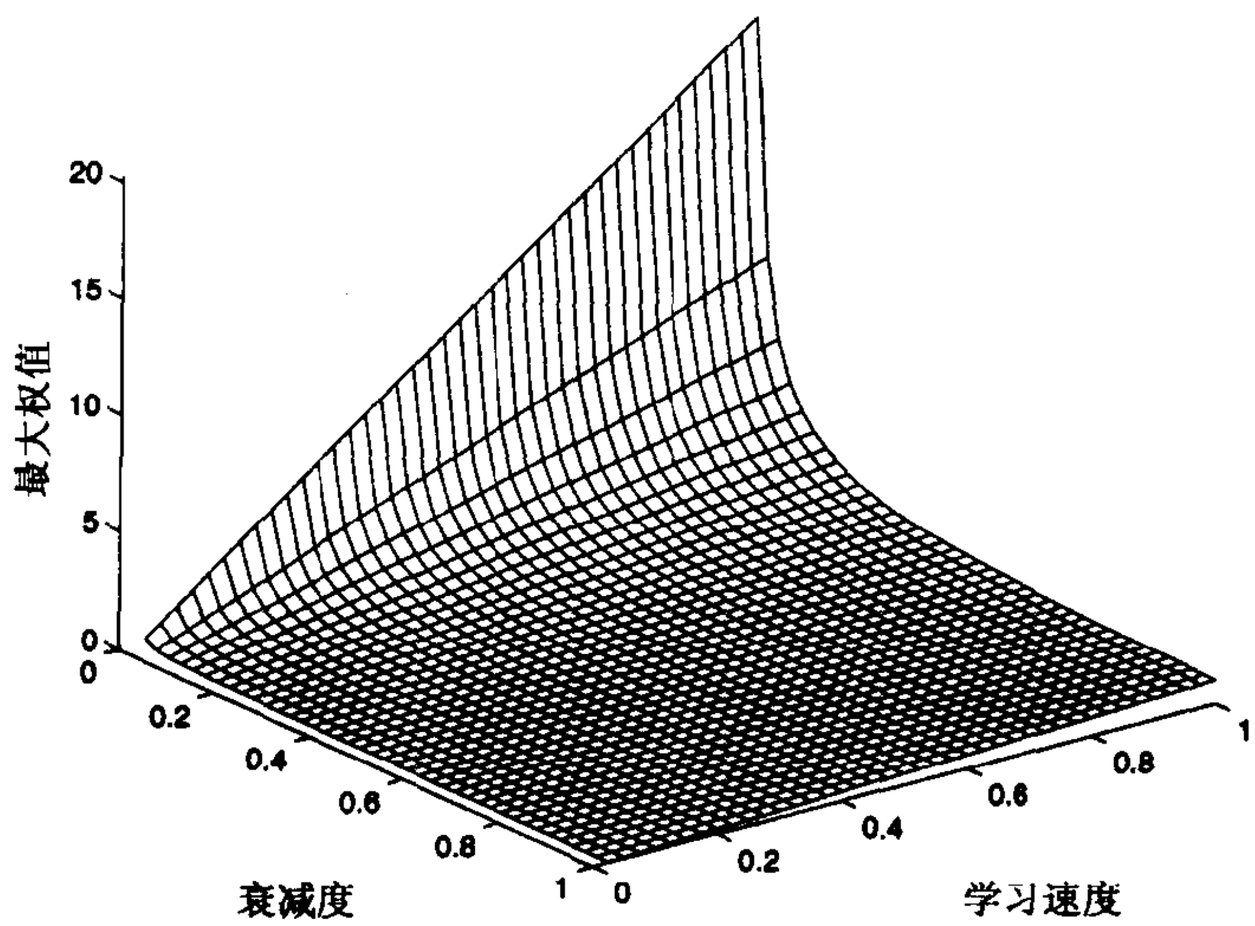



图 13-12 最大权值 $w_{ij}^{max}$

**P13.2** 设学习速度为 0.4，用 instar 规则训练图 13-8 中的橘子识别网络。使用相同的学习序列，需要多少步网络才能根据测量值识别一个橘子?

**解**

下面是训练序列。它一直重复直到网络即使在视觉系统失效($p^0=0$)时也能对橘子的测量值($\mathbf{p}=[1\ -1\ -1]^T$)做出响应。

$$\left\{p^0(1)=0,\mathbf{p}(1)=\begin{bmatrix}1\\-1\\-1\end{bmatrix}\right\},\left\{p^0(2)=1,\mathbf{p}(2)=\begin{bmatrix}1\\-1\\-1\end{bmatrix}\right\},\cdots \tag{13-24}$$

我们用 MATLAB 来求解。下面两行代码设置权值的初始值。

```
w0 = 3;
W = [0 0 0];
```

然后可以模拟网络的第 1 步。

```
p0 = 0;
```

```
p = [1; -1; -1];
a = hardlim(w0*p0+W*p-2)
a =
     0
```

神经元还不能识别橘子，因此输出为 0。当采用 instar 规则时，权值并不改变。

```
W = W + 0.4*a*(p'-W)
W =
     0 0 0
```

神经元开始学习第 2 次迭代中的测量值。

```
p0 = 1;
p = [1; -1; -1];
a = hardlim(w0*p0+W*p-2)
a =
     1
W = W + 0.4*a*(p'-W)
W =
     0.4000   -0.4000   -0.4000
```

但在第 3 次迭代中联想仍然不够强而不能作出响应。

```
p0 = 0;
p = [1; -1; -1];
a = hardlim(w0*p0+W*p-2)
a =
     0
W = W + 0.4*a*(p'-W)
W =
     0.4000   -0.4000   -0.4000
```

13-25

这是第 4 次迭代的结果：

```
a =
     1
W =
     0.6400   -0.6400   -0.6400
```

第 5 次迭代：

```
a =
     0
W =
     0.6400   -0.6400   -0.6400
```

第6次迭代：

```
a =
     1
W =
     0.7840   -0.7840   -0.7840
```

在第7次迭代中网络能够单独根据测量值识别橘子。

```
p0 = 0;
p = [1; -1; -1];
a = hardlim(w0*p0+W*p-2)
a =
     1
W = W + 0.4*a*(p'-W)
W =
     0.8704  -0.8704  -0.8704
```

由于学习速度较低，网络必须经过3次测量值和响应的匹配(偶数次迭代)才能在两者之间建立强的联想。

**P13.3** 本章的识别和回忆网络都只能学习一个向量。画一个能识别和响应下述两个向量的网络图并确定网络的参数：

$$\mathbf{p}_1 = \begin{bmatrix} 5 \\ -5 \\ 5 \end{bmatrix} \quad \mathbf{p}_2 = \begin{bmatrix} -5 \\ 5 \\ 5 \end{bmatrix} \qquad \boxed{13\text{-}26}$$

网络只能响应与上述向量之一相同的一个输入向量。

**解**

我们知道因为要识别三元输入向量，网络必须有三个输入。同时还知道网络有两个输出，分别对应两个响应。

这样的网络可以由两个 instar 神经元组合到一个单层网络而成，如图 13-13 所示。

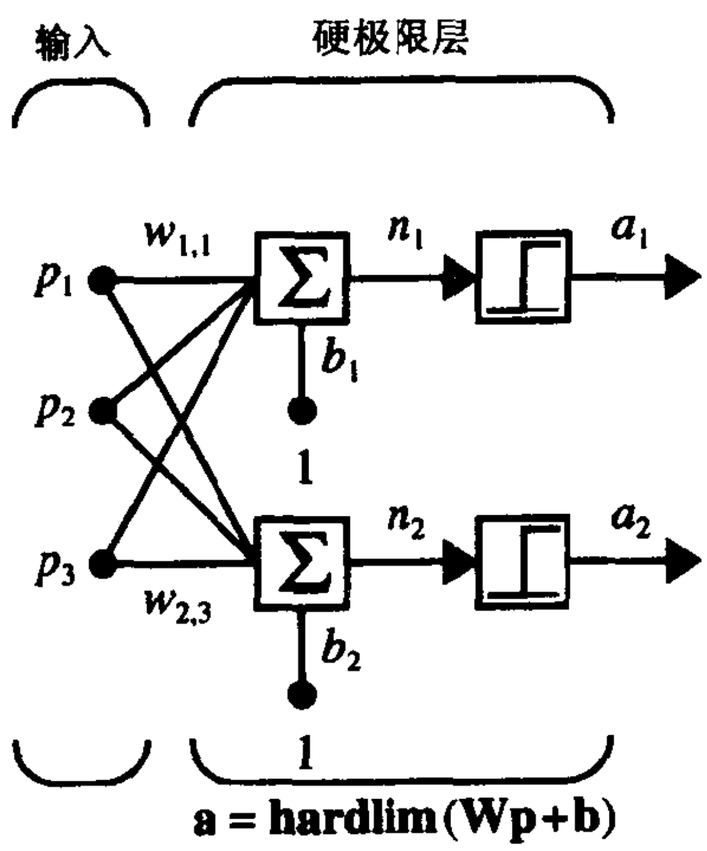



图 13-13 两向量识别网络

现在设置第一个神经元的权值$_1\mathbf{w}$为$\mathbf{p}_1$，所以当输入向量点与$\mathbf{p}_1$方向相同时其净输入达到最大值。类似地，设置$_2\mathbf{w}$为$\mathbf{p}_2$，这样第二个神经元对$\mathbf{p}_2$方向的向量最敏感。将权值向量组合成权值矩阵

$$\mathbf{W} = \begin{bmatrix} {}_1\mathbf{w}^T \\ {}_2\mathbf{w}^T \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix} = \begin{bmatrix} 5 & -5 & 5 \\ -5 & 5 & 5 \end{bmatrix}$$

(注意，我们这里使用了与 Hamming 网络第一层相同的确定权值矩阵方式。实际上，Hamming 网络的第一层就是由 instar 神经元构成的。更详细的讨论请见下一章。)

$\mathbf{p}_1$，$\mathbf{p}_2$的长度相同：

13-27

$$\|\mathbf{p}_1\| = \|\mathbf{p}_2\| = \sqrt{(5)^2 + (-5)^2 + (5)^2} = \sqrt{75}$$

为了保证输入向量和引起响应的存储向量能精确匹配，两者的偏置值依式(13.29)可设置如下：

$$b_1 = b_2 = -\|\mathbf{p}_1\|^2 = -75$$

我们用 MATLAB 来检查网络确实对$\mathbf{p}_1$作出响应。

```
W = [5  -5  5;   -5  5  5];
b = [-75;   -75];
p1 = [5;   -5;   5];
a = hardlim(W*p1+b)
a =
     1
     0
```

第一个神经元作出响应，表示输入向量是$\mathbf{p}_1$。第二个神经元没有响应，表示输入向量不是$\mathbf{p}_2$。还可以检查网络对不同于任何一个存储向量的第三个向量$\mathbf{p}_3$不会作出响应。

```
p3 = [-5; 5; -5];
a = hardlim(W*p3+b)
a =
     0
     0
```

没有一个神经元能识别这个新的向量，所以两个输出都为 0。

**P13.4** 一个用于模式识别的单 instar 神经元，它的权值和偏置值如下：

$$\mathbf{W} = {}_1\mathbf{w}^T = [1 \quad -1 \quad -1], \quad b = -2$$

一个输入向量(长度为$\sqrt{3}$)与权值向量接近到何种程度才能使神经元的输出为 1？求一个向量，它出现在能识别的向量和不能识别的向量之间的边界。

**解**

我们先写出神经元输出的表达式：

$$a = hardlim({}_1\mathbf{w}^T\mathbf{p} + b)$$

13-28 由 *hardlim* 的定义，当且仅当$_1\mathbf{w}$和$\mathbf{p}$的内积大于或等于$-b$时$a$才为 1：

$$_1\mathbf{w}^T\mathbf{p} = \|_1\mathbf{w}\| \, \|\mathbf{p}\| \cos\theta \geq -b$$

代换范数并求解可以获得满足该条件的$_1\mathbf{w}$和$\mathbf{p}$间的最大角度

$$(\sqrt{3})(\sqrt{3})\cos\theta \geq 2$$

$$\theta \leq \cos^{-1}\left(\frac{2}{3}\right) = 48.19°$$

为了得到一个模为$\sqrt{3}$的边界向量，需要求一个向量$\mathbf{p}$满足下述条件：

$$\|\mathbf{p}\| = \sqrt{p_1^2 + p_2^2 + p_3^2} = \sqrt{3}$$

$$_1\mathbf{w}^T\mathbf{p} = w_1p_1 + w_2p_2 + w_3p_3 - b = p_1 - p_2 - p_3 - 2 = 0$$

由于有三个变量，而只有两个约束条件，可以设第三个变量$p_1=0$，并求解

$$\sqrt{p_1^2 + p_2^2 + p_3^2} = \sqrt{3} \quad \Rightarrow \quad p_2^2 + p_3^2 = 3$$

$$p_1 - p_2 - p_3 - 2 = 0 \quad \Rightarrow \quad p_2 + p_3 = -2$$

$$(p_2 + p_3)^2 = p_2^2 + p_3^2 + 2p_2p_3 = (-2)^2 = 4$$

$$3 + 2p_2p_3 = 4 \quad \Rightarrow \quad p_2p_3 = 0.5$$

$$p_2(p_2 + p_3) = p_2^2 + p_2p_3 = p_2^2 + 0.5 = p_2(-2) = -2p_2$$

稍作运算求出$p_2$的两个可能解：

$$p_2^2 + 2p_2 + 0.5 = 0$$

$$p_2 = -1 \pm \sqrt{0.5}$$

为了使$p_2$满足上述值，$p_3$要取以下值：

$$p_2 + p_3 = -1 \pm \sqrt{0.5} + p_3 = -2$$

$$p_3 = -1 \mp \sqrt{0.5} \qquad \boxed{13\text{-}29}$$

因此，下述$\mathbf{p}$是与$\mathbf{w}$有合适距离的可识别向量。

$$\mathbf{p} = \begin{bmatrix} 0 \\ -1 + \sqrt{0.5} \\ -1 - \sqrt{0.5} \end{bmatrix}$$

可以将它代入网络验算：

$$a = hardlim(_1\mathbf{w}^T\mathbf{p} + b)$$

$$a = hardlim\left([1 \quad -1 \quad -1]\begin{bmatrix} 0 \\ -1 + \sqrt{0.5} \\ -1 - \sqrt{0.5} \end{bmatrix} - 2\right)$$

$$a = hardlim(0) = 1$$

向量$\mathbf{p}$使诤输入为0，因此它在instar神经元活跃区的边界上。

**P13.5** 考虑图13-4中所示的instar网络，这个网络的训练序列包括以下输入：

$$\left\{p^0(1) = 0, \mathbf{p}(1) = \begin{bmatrix} -1 \\ 1 \end{bmatrix}\right\}, \left\{p^0(2) = 1, \mathbf{p}(2) = \begin{bmatrix} -1 \\ 1 \end{bmatrix}\right\}, \cdots$$

这两个输出重复提交到网络中，直到权值矩阵 **W** 收敛。

（i）用学习速度 $\alpha = 0.5$ 执行 instar 规则的前四次迭代。假设初始权值矩阵 **W** 被设置为全零。

13-30 （ii）用图形形式显示 instar 规则的每一次迭代结果(如图 13-6)。

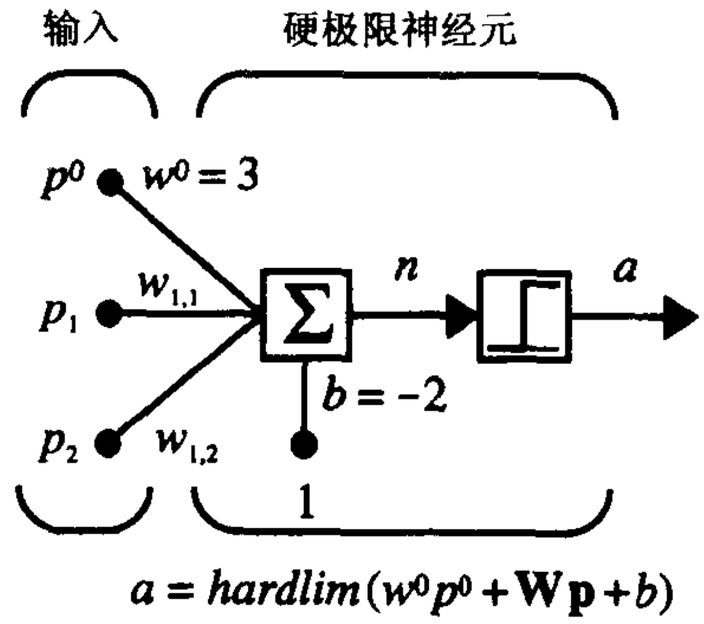



图 13-14 例题 P13.5 的 instar 网络

**解**

（i）由于 **W** 初始化为全 0，instar 神经元在第一次迭代时将不响应测量值。

$$a(1) = hardlim(w^0p^0(1) + \mathbf{W}\mathbf{p}(1) - 2)$$

$$a(1) = hardlim\left(3 \cdot 0 + [0 \quad 0]\begin{bmatrix} -1 \\ 1 \end{bmatrix} - 2\right) = 0$$

神经元不响应。根据 instar 规则，它的权值 ${}_1\mathbf{w}$ 不改变。

$${}_1\mathbf{w}(1) = {}_1\mathbf{w}(0) + 0.5a(1)(\mathbf{p}(1) - {}_1\mathbf{w}(0))$$

$$= \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 0\left(\begin{bmatrix} -1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

由于第二次迭代时出现无条件刺激，instar 神经元将响应。

$$a(2) = hardlim(w^0p^0(2) + \mathbf{W}\mathbf{p}(2) - 2)$$

$$a(2) = hardlim\left(3 \cdot 1 + [0 \quad 0]\begin{bmatrix} -1 \\ 1 \end{bmatrix} - 2\right) = 1$$

13-31 神经元产生了响应，且它的权值 ${}_1\mathbf{w}$ 根据 instar 规则更改。

$${}_1\mathbf{w}(2) = {}_1\mathbf{w}(1) + 0.5a(2)(\mathbf{p}(2) - {}_1\mathbf{w}(1))$$

$$= \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 0.5\left(\begin{bmatrix} -1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix}$$

第三次迭代时，无条件刺激没有出现，而且权值没有收敛到与输入模式足够接近，因此，instar 神经元不响应。

$$a(3) = hardlim(w^0p^0(3) + \mathbf{W}\mathbf{p}(3) - 2)$$

$$a(3) = hardlim\left(3 \times 0 + [-0.5 \; 0.5]\begin{bmatrix} -1 \\ 1 \end{bmatrix} - 2\right) = 0$$

由于神经元不响应，它的权值也不发生改变。

$$
{}_1\mathbf{w}(3) = {}_1\mathbf{w}(2) + 0.5a(3)(\mathbf{p}(3) - {}_1\mathbf{w}(2))
$$

$$
= \begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix} + 0\left(\begin{bmatrix} -1 \\ 1 \end{bmatrix} - \begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix}\right) = \begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix}
$$

由于无条件刺激在第四次迭代中再次出现，instar 神经元将响应。

$$
a(4) = hardlim(w^0 p^0(4) + \mathbf{W}\mathbf{p}(4) - 2)
$$

$$
a(4) = hardlim\left(3 \times 1 + [-0.5 \quad 0.5]\begin{bmatrix} -1 \\ 1 \end{bmatrix} - 2\right) = 1
$$

由于 instar 神经元被激活，它的权值也要更新。

$$
{}_1\mathbf{w}(4) = {}_1\mathbf{w}(3) + 0.5a(4)(\mathbf{p}(4) - {}_1\mathbf{w}(3))
$$

$$
= \begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix} + 0.5\left(\begin{bmatrix} -1 \\ 1 \end{bmatrix} - \begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix}\right) = \begin{bmatrix} -0.75 \\ 0.75 \end{bmatrix}
$$

这完成了第四次迭代。如果我们继续下去，${}_1\mathbf{w}$ 将收敛于 $\mathbf{p}$。 13-32

（ii）注意到权值仅在第二和四次迭代(instar 神经元活跃)时改变。回忆式(13.34)，当 instar 神经元活跃时，学习规则可写为

$$
{}_1\mathbf{w}(q) = {}_1\mathbf{w}(q-1) + \alpha(\mathbf{p}(q) - {}_1\mathbf{w}(q-1)) = (1-\alpha)\,{}_1\mathbf{w}(q-1) + \alpha\mathbf{p}(q)
$$

当 instar 神经元活跃时，权值向量沿着旧权值向量和输入向量之间的连线向输入向量方向移动。图 13-15 显示了本题中权值向量的移动。权值在第二和四次迭代时更新。由于 $\alpha = 0.5$，当 instar 神经元活跃时权值向量将从当前位置移动到输入向量的中心位置。

$$
{}_1\mathbf{w}(q) = (0.5)\,{}_1\mathbf{w}(q-1) + (0.5)\mathbf{p}(q)
$$

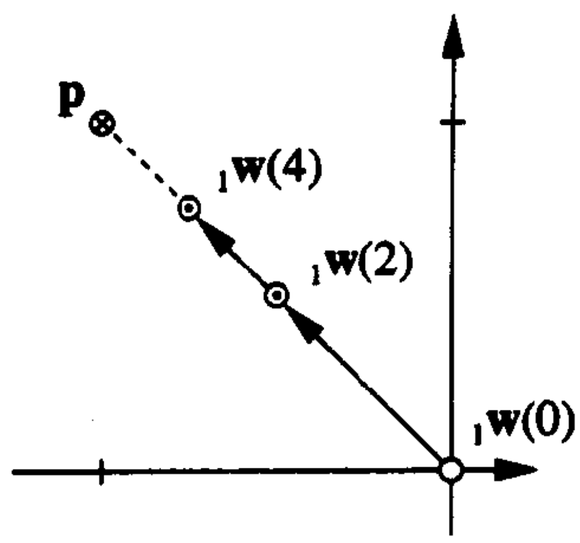



图 13-15 instar 规则的例子 13-33

## 13.5 结束语

本章中介绍了一些能产生联想的简单网络，同时还研究了使网络产生新联想的学习规则，每条规则通过增强同时发生的刺激和响应之间的联想发生作用。

简单的联想网络和学习规则本身就很有用，而它们也是构成更强网络的重要单元。本章介绍的两种网络和相关的学习规则是下面三章讨论的一些重要的网络的基础。instar 网络用于训练识别某种模式，outstar 网络则用于训练回忆模式。我们将在第 14 和 15 章中用 instar 网络层进行模式识别。这些网络非常似于第 3 章的 Hamming 网络(它的第一层事实上是由 instar 神经元构成)。第 16 章将介绍一个更复杂的网络，它将 instar 和 outstar 组合在一起产

13-34 生稳定的学习。

## 参考文献

[Ande72] J. Anderson, "A simple neural network generating an interactive memory," *Mathematical Biosciences*, vol. 14, pp. 197 - 220, 1972.

Anderson 为联想存储器提出了一种"线性联想器"模型，这个模型使用推广的 Hebb 假设进行训练，在输入和输出向量之间学习联想。

[Gros68] S. Grossberg, "Some physiological and biochemical consequences of psychological postulates," *Proceedings of the National Academy of Sciences*, vol. 60, pp. 785 - 765, 1968.

这篇文章介绍了用于联想学习的一个早期的(非线性微分方程组)数学模型，它综合了数学、心理学和生理学等各方面的思想。

[Gros82] S. Grossberg, *Studies of Mind and Brain*, Boston: D. Reidel Publishing Co., 1982.

这本书收集了 Stephen Grossberg 从 1968 年到 1980 年的文章。其中有很多重要的思想在后面的 Grossberg 网络中得到了运用，如在自适应谐振理论网络中。

[Hebb49] D. O. Hebb, *The Organization of Behavior*, New York: Wiley, 1949.

这本里程碑似的书籍的一个主要论点是行为可以用神经元的相互作用和反应来解释。其中，Hebb 提出了最早的学习规则中的一个，这个规则假定一种在细胞级的学习机制。

[Koho72] T. Kohonen, "Correlation matrix memories," *IEEE Transactions on Computers*, vol. 21, pp. 353 - 359, 1972.

Kohonen 在本书中为联想存储器提出了一种相关矩阵模型。这种模型用外积规则(也称 Hebb 规则)进行训练，学习输入向量和输出向量之间的联想。模型强调的是网络的数
13-35 学结构。Anderson 在同时发表了一篇相近的文章[Ande72]，但两人是独立进行研究的。

[Koho87] T. Kohonen, *Self-Organization and Associative Memory*, 2nd Ed., Berlin: Springer-Verlag, 1987.

本书中介绍了 Kohonen 学习规则和几个使用该规则的网络。在书中还提供了对线性联想模型的完整分析，并给出了很多扩充和例子。

[Leib90] D. Lieberman, *Learning, Behavior and Cognition*, Belmont, CA: Wadsworth, 1990.

这是一本优秀的行为心理学书籍。这个领域对于用神经网络模拟人类(或动物)学习
13-36 的任何人而言都是很有兴趣的。

## 习题

**E13.1** 图 13-16 中的网络使用带衰减的 Hebb 规则进行训练，其中学习速度 $\alpha = 0.3$，衰减速度 $\gamma = 0.1$

(i) 如果 $w$ 初始化为 0，$w^0$ 和 $b$ 保持常数(如图 13-16 所示)，需要将下面的训练集连续输入网络多少次才能使神经元响应测试集？绘制 $w$ 与迭代次数的关系图：

训练集：$\{p^0 = 1,\ p = 1\}$，　测试集：$\{p^0 = 0,\ p = 1\}$

（ii）如果 $w$ 初始化为 1，需要将下述训练集连续输入网络多少次才能使神经元响应测试集？绘制 $w$ 与迭代次级的关系图：

训练集：$\{p^0=1,\ p=0\}$，　测试集：$\{p^0=0,\ p=1\}$

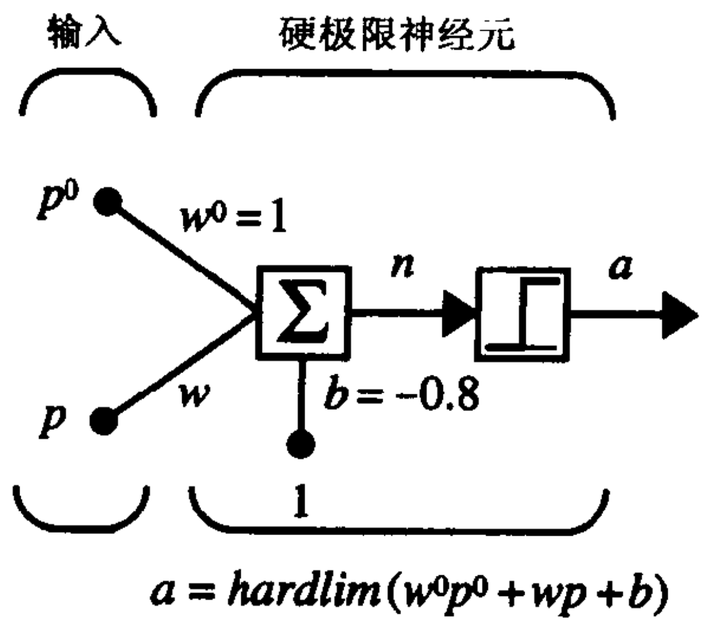



图 13-16　联想网络

**E13.2** 对练习 E13.1 的问题(i)，用式(13.19)确定 $w$ 的稳定状态值，用练习 E13.1 问题(i)的图验证这个结果。

**E13.3** 重复练习 E13.1，但此时用无衰减($\gamma=0$)的 Hebb 规则。

**E13.4** 下述规则类似于 instar 规则，但它的表现有很大不同：　13-37

$$\Delta w_{ij} = -\alpha a_i(p_j + w_{ij}^{old})$$

（i）确定 $\Delta w_{ij}$ 为非 0 的条件。

（ii）当 $\Delta w_{ij}$ 非 0 时权值逼近于多少？

（iii）考虑该规则有哪些用途？

**E13.5** 图 13-17 中的 instar 网络用以识别向量。

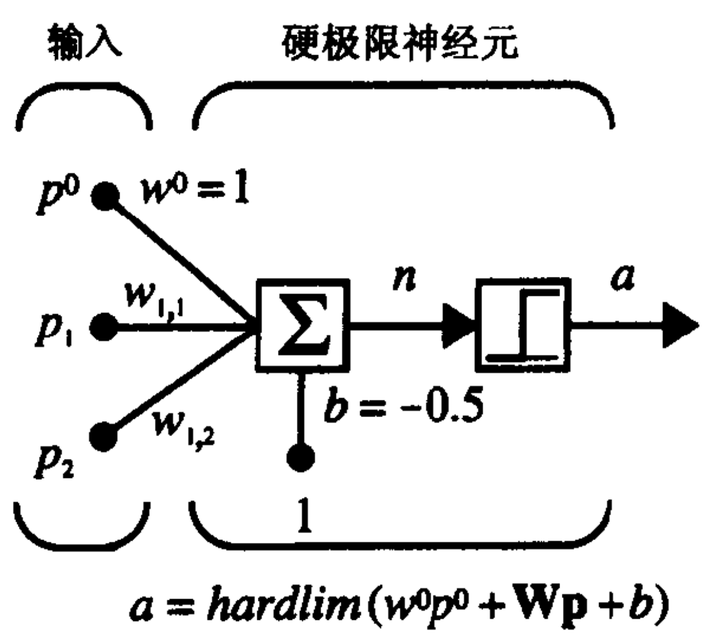



图 13-17　向量识别器

（i）用 instar 规则和下述训练序列训练网络。只将 instar 规则用于第二个输入权值(初始化为 0)，使用学习速度 0.6。其他权值和偏置值保持为图中所示的常数。(可以用 MATLAB 完成计算。)

$$\left\{p^0(1)=1,\ \mathbf{p}(1)=\begin{bmatrix}0.174\\0.985\end{bmatrix}\right\}\quad\left\{p^0(2)=0,\ \mathbf{p}(2)=\begin{bmatrix}-0.174\\0.985\end{bmatrix}\right\}$$

$$\left\{p^0(3)=1,\ \mathbf{p}(3)=\begin{bmatrix}0.174\\0.985\end{bmatrix}\right\}\quad\left\{p^0(4)=0,\ \mathbf{p}(4)=\begin{bmatrix}-0.174\\0.985\end{bmatrix}\right\}$$

$$\left\{p^0(5)=1,\ \mathbf{p}(5)=\begin{bmatrix}0.174\\0.985\end{bmatrix}\right\}\quad\left\{p^0(6)=0,\ \mathbf{p}(6)=\begin{bmatrix}-0.174\\0.985\end{bmatrix}\right\}$$

(**ii**) 最后的 **W** 值是多少?

13-38 (**iii**) 最后的值与训练序列中的向量相比如何?

(**iv**) 如果网络用相同的训练序列迭代多次，权值在训练后的数值应该是多少?

**E13.6** 考虑图 13-18 中的 instar 网络，网络的训练序列将由下列输入组成:

$$\left\{p^0(1)=0,\ \mathbf{p}(1)=\begin{bmatrix}1\\-1\end{bmatrix}\right\},\ \left\{p^0(2)=1,\ \mathbf{p}(2)=\begin{bmatrix}1\\-1\end{bmatrix}\right\},\cdots$$

这两个输入集重复输入网络直至权值矩阵 **W** 收敛。

(**i**) 执行 instar 规则的前 8 次迭代，其中学习速度 $\alpha=0.25$。假设权值矩阵 **W** 的初始值为 $\mathbf{W}=[1\quad 0]$。

(**ii**) 用图形方式显示每次迭代的结果(如图 13-6 所示)。

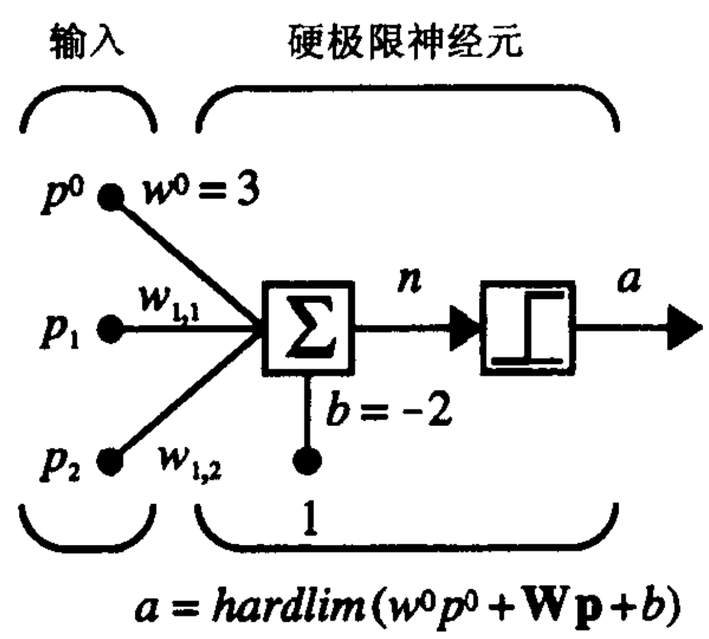



图 13-18 习题 E13.6 的 instar 网络

**E13.7** 画出当给定不同的刺激(值为 1)时能识别三种不同的四元素向量(元素值为 ±1)的网络图。

(**i**) 网络应有多少个输入和输出? 传输函数是什么?

13-39 (**ii**) 如果它能识别下述向量，请给出网络的权值:

$$\mathbf{p}_1=\begin{bmatrix}1\\-1\\1\\-1\end{bmatrix},\quad \mathbf{p}_2=\begin{bmatrix}-1\\-1\\1\\-1\end{bmatrix},\quad \mathbf{p}_3=\begin{bmatrix}1\\-1\\-1\\1\end{bmatrix}$$

(**iii**) 选择合适的偏置值，并解释原因。

(**iv**) 用上述向量之一检查网络。它的响应正确吗?

(**v**) 用向量 $\mathbf{p}_1=\begin{bmatrix}-1\\-1\\1\\1\end{bmatrix}$ 检查网络。它的响应为什么是正确的?

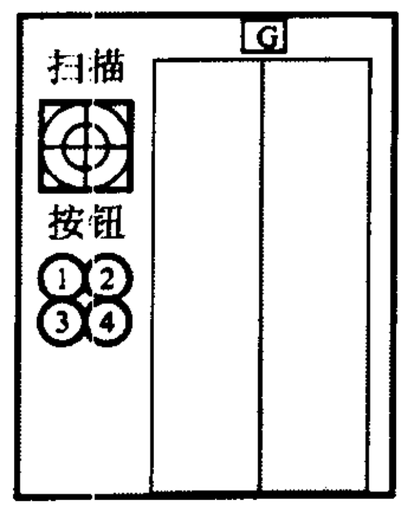
**E13.8** 本章包含一个识别网络的例子，最初用视觉系统来识别橘子。一开始需要视觉系统通知网络何时有橘子，但最后网络学习了用传感器的测量来识别橘子。

(i) 让我们用人来代替视觉系统。一开始网络依靠人告诉它是否有橘子。你认为此时的网络是有监督学习还是无监督学习？

(ii) 在何种情况下，人的输入类似于前面几章中用于有监督训练的目标？

(iii) 它在什么情况下是不同的？

**E13.9** 图 13-19 的网络安装在一个电梯中，该电梯在一个豪华和高度安全的公司大厦中由三个高级执行官使用，它有标记“1”到“4”的 4 个按钮表示底层上面的四层。当一个执行官 进入电梯底层后，电梯用视网膜扫描的方式判断是谁，并用网络决定这个人最可能去的楼层。如果猜测不对，这个人可以在任何时候按不同的按钮，否则它将省去这个重要执行官的按钮动作。




13-40

图 13-19

网络的输入/输出函数为

$$\mathbf{a} = \mathbf{hardlims}(\mathbf{W}^0\mathbf{p}^0 + \mathbf{W}\mathbf{p} + \mathbf{b})$$

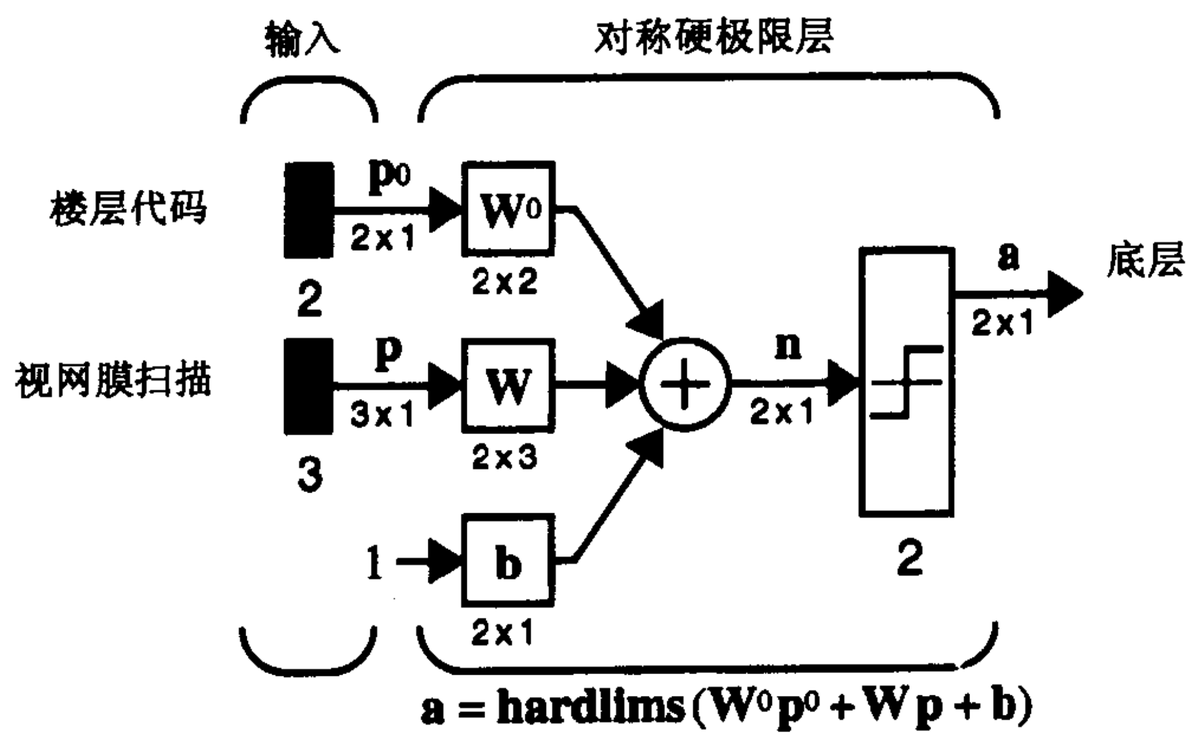



图 13-20 电梯网络

当按钮按下时第一个输入 $\mathbf{p}^0$ 提供网络一个楼层代码(图 13-21)。

$$\mathbf{p}_1^0 = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \quad (1\text{ 楼}) \qquad \mathbf{p}_2^0 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad (2\text{ 楼})$$

$$\mathbf{p}_3^0 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad (3\text{ 楼}) \qquad \mathbf{p}_4^0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad (4\text{ 楼})$$

如果没有按按钮，则无代码。

$$\mathbf{p}_0^0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (\text{没有按按钮})$$

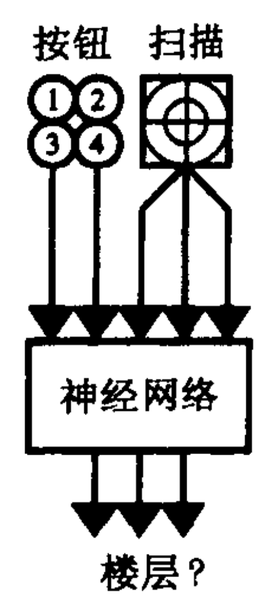



图 13-21

第一个输入由一单位矩阵加权，且偏置值设为 -0.5，这样如果按了按钮，网络将响应出楼层代码。

$$\mathbf{W}^0 = \mathbf{I}, \quad \mathbf{b} = \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}$$

13-41 第二个输入总是存在的。它包含的三个元素代表了三位执行官：

$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}(\text{总裁}), \quad \mathbf{p}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}(\text{副总裁}), \quad \mathbf{p}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}(\text{董事长})$$

网络通过用 outstar 规则(学习速度为 0.6)更新第二组权值，学习回忆三位执行官所在的楼层。这些权值最初均设置为零：

$$\mathbf{W} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

(i) 用 MATLAB 模拟下列事件的网络：

总裁按按钮‘4’，副总裁按按钮‘3’
董事长按按钮‘1’，副总裁按按钮‘3’
董事长按按钮‘2’，总裁按按钮‘4’

换句话说，用下述序列训练网络：

$$\{\mathbf{p}^0 = \mathbf{p}_4^0, \quad \mathbf{p} = \mathbf{p}_1\}, \{\mathbf{p}^0 = \mathbf{p}_3^0, \quad \mathbf{p} = \mathbf{p}_2\}, \{\mathbf{p}^0 = \mathbf{p}_1^0, \quad \mathbf{p} = \mathbf{p}_3\},$$
$$\{\mathbf{p}^0 = \mathbf{p}_3^0, \quad \mathbf{p} = \mathbf{p}_2\}, \{\mathbf{p}^0 = \mathbf{p}_2^0, \quad \mathbf{p} = \mathbf{p}_3\}, \{\mathbf{p}^0 = \mathbf{p}_4^0, \quad \mathbf{p} = \mathbf{p}_1\}$$

(ii) 最后的权值是什么？

(iii) 现在继续对下述事件模拟网络：

总裁不按按钮
副总裁不按按钮
董事长不按按钮

(iv) 网络把每位执行官送到哪一楼层？

(v) 如果三位执行官分别按下述按钮许多次，你期望得到的权值矩阵是什么形式？

总裁按按钮‘3’
副总裁按按钮‘2’
董事长按按钮‘4’

13-42

# 第14章 竞争网络

## 14.1 目的

第3章所介绍的Hamming网络，展示了一种用神经网络进行模式识别的技术，这种技术需要事先知道原型模式并且将原型模式以权值矩阵的行和网络相结合。

本章我们将讨论一些在结构以及操作上都与Hamming网络极为相似的网络。与Hamming网络不同的是，这些网络使用第13章的联想学习规则对模式分类进行自适应学习。本章介绍了三种这样的网络：竞争网络、特征图网络和学习矢量量化网络。 14-1

## 14.2 理论和实例

Hamming网络是竞争网络中最简单的例子。它的输出层神经元互相竞争以确定胜者。胜者将指出哪一种原型模式最能代表输入模式。这种竞争是通过在输出层神经元之间一组负连接(即侧向抑制)来实现的。本章我们将说明这种竞争何以能够与第13章中的联想学习规则相结合来建立强大的自组织(无监督的)网络。

早在1959年，Frank Rosenblatt就创造了一种简单的"自发"分类器，这是种基于感知机的无监督的神经网络。这种网络能够学会将输入的矢量分类成数目大致相等的两类。

在20世纪60年代后期及70年代早期，Stephen Grossberg引入了许多使用侧向抑制而产生良好效果的竞争网络。他获得的有用特性就是减少噪声、对比增加和向量规格化。第15章及第16章将讨论他的这些网络。

1973年，Christoph von der Malsburg引入了一种自组织的学习规则，这种规则用这样方法归类输入，使得相邻的神经元对相似的输入产生反应。他这种网络的拓扑结构以某些方式模仿David Hubel和Torten Wiesel过去所发现的猫的视觉皮层的结构。他的学习规则引起了人们极大的兴趣，但是这种学习规则使用一种非局部计算以保证权值是规格化的。这使得它在生物学上缺少合理性。

Grossberg重新发现instar规则(在第13章中介绍过)，从而发展了von der Malsburg的成果(instar规则首先是由Nils Nilsson 1965年在他的《学习机器》(*Learning Machines*)一书中首先介绍的)。Grossberg证明了instar规则免去了重新规格化权值的必要性，因为学习规格输入向量的权值向量能够自动使自己规格化。

Grossberg和von der Malsburg的工作着重于他们的网络在生物学上的合理性。另一位颇具影响力的研究者Teuvo Kohonen也是竞争网络的积极提议者。然而他的重点主要在网络的工程应用以及有效的数学描述。20世纪70年代中，他发展一种instar规则的简化形式，并且受von der Malsburg和Grossberg的启发，发现一种有效的将拓扑结构结合为竞争网络的方法。

在本章我们集中讨论Kohonen的竞争网络框架，他的模型体现了竞争网络的主要特征， 14-2
而且也比Grossberg的网络在数学上更易处理。这些网络为竞争学习提供了良好的说明。

我们将从简单的竞争网络开始。然后将结合网络拓扑结构讲解自组织特征图。最后我们将讨论学习向量量化。它将竞争和有监督的学习框架结合起来。

### 14.2.1 Hamming 网络

既然本章所讨论的竞争网络与 Hamming 网络(见图 14-1)紧密相关，所以先回顾一下 Hamming 网络的一些主要概念。

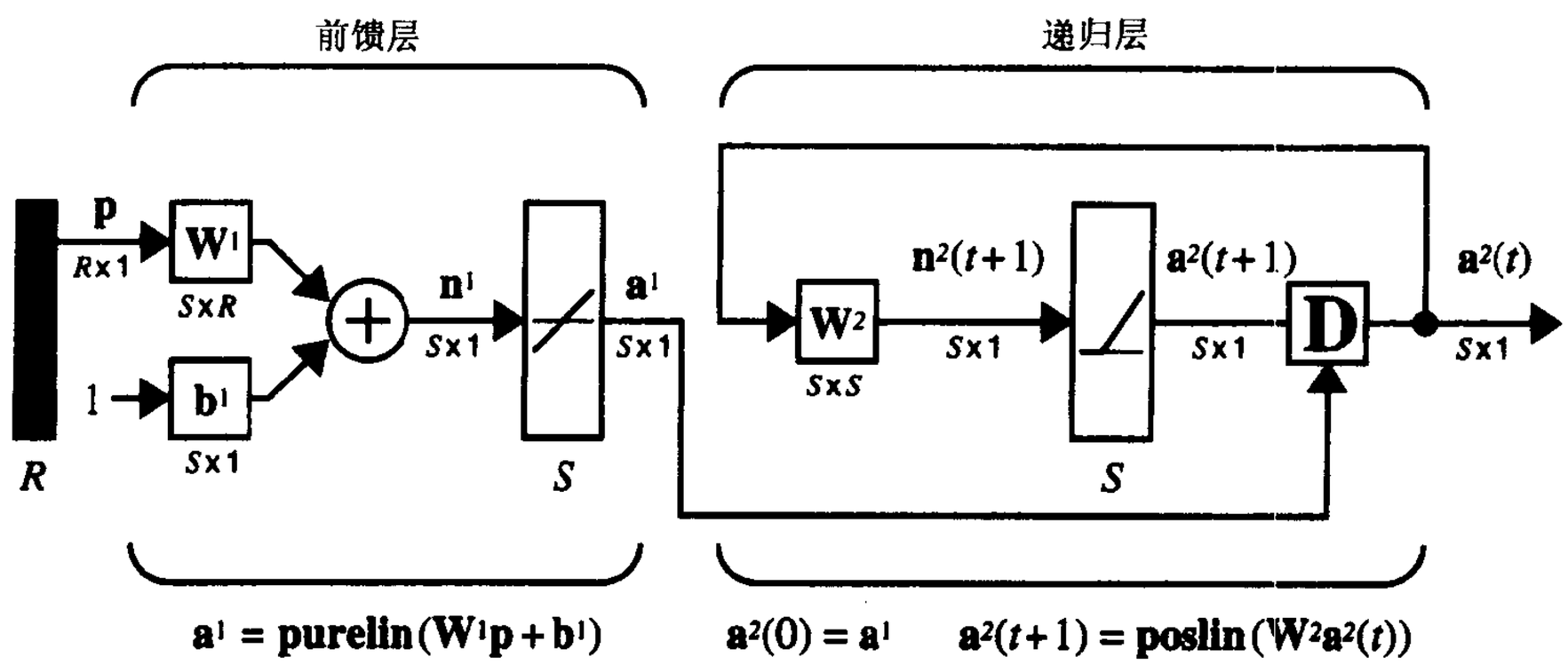



图 14-1 Hamming 网络

Hamming 网络由两层组成。第一层(有 instar 的那一层)将输入向量与原型向量联系起来。第二层采用竞争方式决定哪种原型向量最接近输入向量。

1. *第一层*

从第 3 章知道，一个 instar 只能够识别一种模式。为了能够识别多种模式，就必须有多种 instar，Hamming 网络实现了这一点。

假设要让网络识别以下原型向量：

14-3

$$\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_Q\} \tag{14.1}$$

第一层的权值矩阵为 $\mathbf{W}^1$，偏置向量 $\mathbf{b}^1$ 是

$$\mathbf{W}^1 = \begin{bmatrix} {}_1\mathbf{w}^T \\ {}_2\mathbf{w}^T \\ \vdots \\ {}_s\mathbf{w}^T \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \vdots \\ \mathbf{p}_Q^T \end{bmatrix}, \quad \mathbf{b}^1 = \begin{bmatrix} R \\ R \\ \vdots \\ R \end{bmatrix} \tag{14.2}$$

矩阵 $\mathbf{W}^1$ 的每一行都代表我们想要识别的一种原型向量，$\mathbf{b}^1$ 的每个元素都设为等于每个输入向量的元素个数 $R$(神经元个数 $S$ 等于将识别的原型向量的个数 $Q$)。

第一层的输出是

$$\mathbf{a}^1 = \mathbf{W}^1\mathbf{p} + \mathbf{b}^1 = \begin{bmatrix} \mathbf{p}_1^T\mathbf{p} + R \\ \mathbf{p}_2^T\mathbf{p} + R \\ \vdots \\ \mathbf{p}_Q^T\mathbf{p} + R \end{bmatrix} \tag{14.3}$$

注意：第一层的输出等于原型向量与输入的内积再加上 $R$。正如我们在第 3 章 3.2.3 节讨论的那样，这些内积表明原型向量与输入向量之间的接近程度。(也见 13.2.3 节对 instar 的讨论。)

2. 第二层

在第 13 章对 instar 的讨论中，使用了硬极限传输函数来确定输入向量离原型向量是否足够近。在 Hamming 网络的第二层有多个 instar，因此必须确定哪个原型向量与输入最为接近。我们将会使用一个竞争层而不是硬极限传输函数来选择最为接近的原型。

第二层是竞争层。这一层的神经元用前馈层的输出初始化，这些输出指明了原型模式与输入向量的相互关系。然后神经元相互竞争以确定胜者。竞争过后，只有一个神经元有非零输出。获胜的神经元指明输入属于哪类(每个原型向量代表一个类)。

第一层的输出 $\mathbf{a}^1$ 用来初始化第二层：

$$\mathbf{a}^2(0) = \mathbf{a}^1 \tag{14.4}$$

14-4

然后第二层的输出用如下递归关系更新：

$$\mathbf{a}^2(t+1) = \mathbf{poslin}(\mathbf{W}^2\mathbf{a}^2(t)) \tag{14.5}$$

第二层的权值矩阵 $\mathbf{W}^2$ 的对角线元素都被设为 1，不在对角线上的元素，设为某个小的负数：

$$w_{ij}^2 = \begin{cases} 1, & i = j \\ -\varepsilon, & i \neq j \end{cases} \quad \left(\text{其中 } 0 < \varepsilon < \frac{1}{S-1}\right) \tag{14.6}$$

**横向抑制** 这个矩阵产生横向抑制，即每个神经元的输出都将对所有其他的神经元产生一种抑制作用。为了说明这种效果，用 1 和 $-\varepsilon$ 的权值代入 $\mathbf{W}^2$ 中合适的元素。

$$a_i^2(t+1) = poslin\left(a_i^2(t) - \varepsilon\sum_{j\neq i} a_j^2(t)\right) \tag{14.7}$$

每次迭代，每个神经元的输出都会随着所有其他神经元输出的和按比例下降(最小的输出为 0)。初始状态最大的神经元比起其他神经元的输出降得慢一些。最终这个神经元将成为惟一一个有正值输出的神经元。这时网络已达到了稳定状态。第二层神经元中有稳定正值输出的神经元是和输入匹配得最好的原型向量的那个神经元。

**胜者全得** 因为只有一个神经元有非零输出，这就被叫作胜者全得竞争，在第 15 章我们将讨论这种竞争。

如果想试验 Hamming 网络如何解决苹果、橘子的分类问题，可以用第 3 章介绍过的 *Neural Network Design Demonstration Hamming Classification*(**nnd3hamc**)。

### 14.2.2 竞争层

**竞争** Hamming 网络的第二层的神经元激活自己而抑制所有其他神经元，这就叫做竞争。为了简化本章余下部分的讨论，我们将定义一个传输函数，来作一个递归竞争层所做的工作：

$$\mathbf{a} = \mathbf{compet}(\mathbf{n}) \tag{14.8}$$

它找到最大净输入的神经元的下标，并将其输出设置为 1(最低下标的神经元的将受到束缚)。所有其他的输出都设置为 0。

14-5

$$a_i = \begin{cases} 1, & i = i^* \\ 0, & i \neq i^* \end{cases} (\text{其中 } n_{i^*} \geqslant n_i, \forall i, \text{且 } i^* \leqslant i, \forall n_i = n_{i^*}) \tag{14.9}$$

用一个作用于第一层的竞争传输函数代替 Hamming 网络的递归层，将简化此章的讨论（我们将在第 15 章进一步讨论竞争过程的细节）。竞争层如图 14-2 所示。

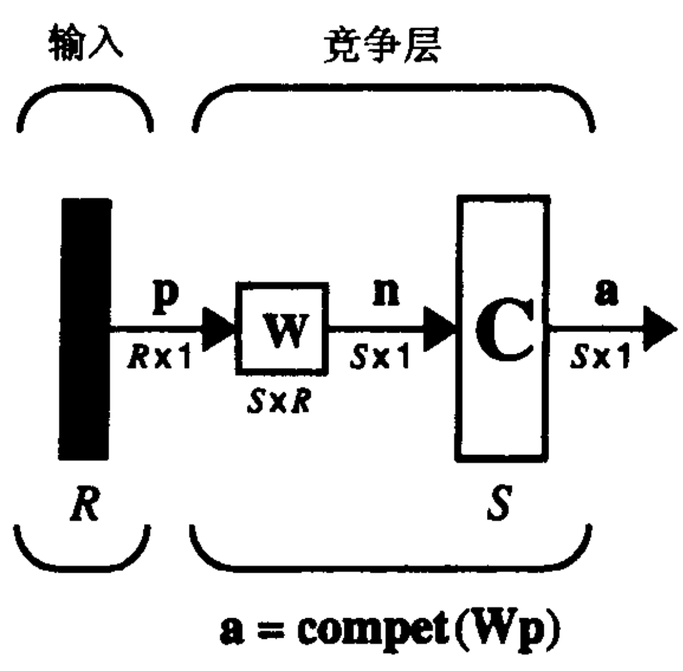



图 14-2 竞争层

正如 Hamming 网络那样，原型向量存储于 **W** 矩阵的每行之中，净输入 **n** 计算输入向量 **p** 和原每个原型$_i$**w** 之间的矩离（假设向量规格化长度为 $L$）。每个神经元 $i$ 的净输入 $n_i$ 和 **p** 与原型向量$_i$**w** 之间的夹角 $\theta_i$ 成正比：

$$\mathbf{n} = \mathbf{W}\mathbf{p} = \begin{bmatrix} {}_1\mathbf{w}^T \\ {}_2\mathbf{w}^T \\ \vdots \\ {}_S\mathbf{w}^T \end{bmatrix} \mathbf{p} = \begin{bmatrix} {}_1\mathbf{w}^T\mathbf{p} \\ {}_2\mathbf{w}^T\mathbf{p} \\ \vdots \\ {}_S\mathbf{w}^T\mathbf{p} \end{bmatrix} = \begin{bmatrix} L^2\cos\theta_1 \\ L^2\cos\theta_2 \\ \vdots \\ L^2\cos\theta_s \end{bmatrix} \tag{14.10}$$

竞争传输函数对权值向量与输入向量方向最为接近的神经元的输出指定为 1：

$$\mathbf{a} = \mathbf{compet}(\mathbf{W}\mathbf{p}) \tag{14.11}$$

14-6 要试验竞争网络和苹果、橘子分类问题，请用 *Neural Network Design Demonstration Competitive Classification* (**nnd14cc**)。

1. 竞争学习

我们现在可以通过把 **W** 矩阵的行设置为理想原型向量值而设计出一个竞争网络的分类器。然而，我们宁愿有一个学习规则，在不知道原型向量的情况下用来训练竞争网络的权值。其中一个这样的学习规则就是第 13 章中讨论的 instar 规则：

$${}_i\mathbf{w}(q) = {}_i\mathbf{w}(q-1) + \alpha a_i(q)(\mathbf{p}(q) - {}_i\mathbf{w}(q-1)) \tag{14.12}$$

对于竞争网络：**a** 只对竞争获胜神经元是非零的。因此，能够从 Kohonen 规则中得到相同的结果。

$$\begin{aligned} {}_i\mathbf{w}(q) &= {}_i\mathbf{w}(q-1) + \alpha(\mathbf{p}(q) - {}_i\mathbf{w}(q-1)) \\ &= (1-\alpha)\,{}_i\mathbf{w}(q-1) + \alpha\mathbf{p}(q) \end{aligned} \tag{14.13}$$

而

$${}_i\mathbf{w}(q) = {}_i\mathbf{w}(q-1), \quad i \neq i^* \tag{14.14}$$

因此，权值矩阵中最接近输入向量的行(或者与输入向量有最大内积的行)向输入向量移动。它沿着权值矩阵原有行与输入向量之间的连线移动，如图 14-3 所示。

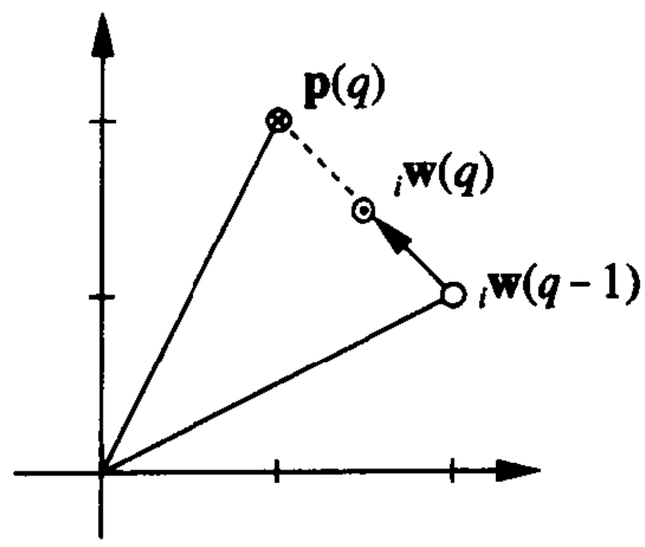



图 14-3 Kohonen 规则的图示

现在用图 14-4 的 6 个向量来演示竞争层如何学习分类向量。6 个向量为：

$$\mathbf{p}_1 = \begin{bmatrix} -0.1961 \\ 0.9806 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} 0.1961 \\ 0.9806 \end{bmatrix}, \quad \mathbf{p}_3 = \begin{bmatrix} 0.9806 \\ 0.1961 \end{bmatrix},$$
$$\mathbf{p}_4 = \begin{bmatrix} 0.9806 \\ -0.1961 \end{bmatrix}, \quad \mathbf{p}_5 = \begin{bmatrix} -0.5812 \\ -0.8137 \end{bmatrix}, \quad \mathbf{p}_6 = \begin{bmatrix} -0.8137 \\ -0.5812 \end{bmatrix} \tag{14.15}$$

14-7

我们的竞争网络有 3 个神经元，因而它可以将向量分成 3 类。下面是“随机”选择的规格化初始权值：

$${}_1\mathbf{w} = \begin{bmatrix} 0.7071 \\ -0.7071 \end{bmatrix}, \quad {}_2\mathbf{w} = \begin{bmatrix} 0.7071 \\ 0.7071 \end{bmatrix}, \quad {}_3\mathbf{w} = \begin{bmatrix} -1.0000 \\ 0.0000 \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} {}_1\mathbf{w}^T \\ {}_2\mathbf{w}^T \\ {}_3\mathbf{w}^T \end{bmatrix} \tag{14.16}$$

数值向量如图 14-5 所示，其中权值向量用箭头表示。我们将 $\mathbf{p}_2$ 提交给网络：

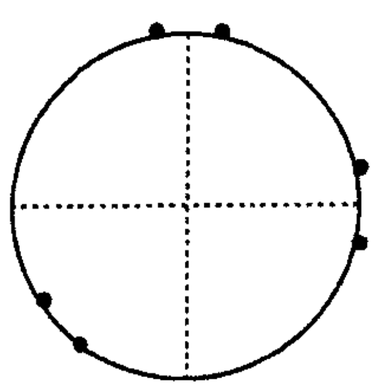


图 14-4 样本输入向量

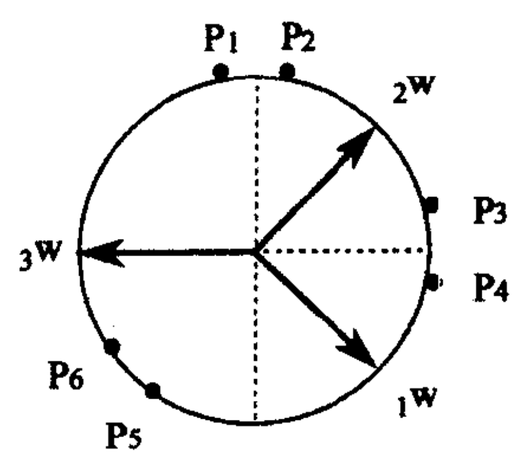



图 14-5

$$\mathbf{a} = \mathbf{compet}(\mathbf{W}\mathbf{p}_2) = \mathbf{compet}\left(\begin{bmatrix} 0.7071 & -0.7071 \\ 0.7071 & 0.7071 \\ -1.0000 & 0.0000 \end{bmatrix}\begin{bmatrix} 0.1961 \\ 0.9806 \end{bmatrix}\right)$$
$$= \mathbf{compet}\left(\begin{bmatrix} -0.5547 \\ 0.8321 \\ -0.1961 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{14.17}$$

第 2 个神经元的权值向量与 $\mathbf{p}_2$ 最接近，因而它竞争获胜($i^* = 2$)，且输出为 1。现在应

用 Kohonen 学习规则，其中学习速度 $\alpha = 0.5$。

$$
\begin{aligned}
{}_2\mathbf{w}^{new} &= {}_2\mathbf{w}^{old} + \alpha(\mathbf{p}_2 - {}_2\mathbf{w}^{old}) \\
&= \begin{bmatrix} 0.7071 \\ 0.7071 \end{bmatrix} + 0.5\left(\begin{bmatrix} 0.1961 \\ 0.9806 \end{bmatrix} - \begin{bmatrix} 0.7071 \\ 0.7071 \end{bmatrix}\right) = \begin{bmatrix} 0.4516 \\ 0.8438 \end{bmatrix}
\end{aligned} \tag{14.18}
$$

Kohonen 学习规则将 ${}_2\mathbf{w}$ 移近至 $\mathbf{p}_2$，正如图 14-6 中所示。如果不断随机选择输入向量并且将它们输入网络，那么每次迭代与输入向量最近的权值向量与将会向输入向量移动。最终每个权值向量将指向输入向量的不同簇。每个权值向量会变成不同的簇的原型向量。

14-8

这个问题是足够简单的，以至能够预计哪个权值向量将指向那个簇。最终的权值向量将会如图 14-7 中所示。

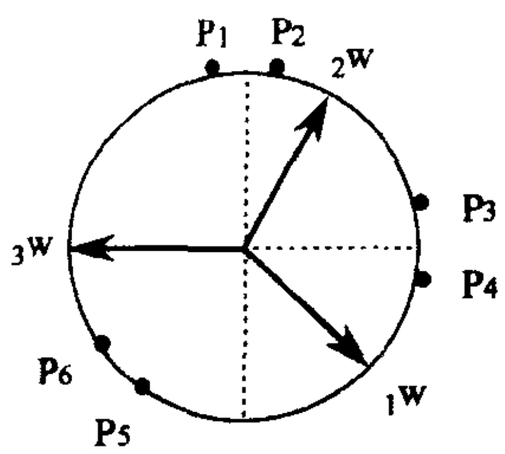



图 14-6

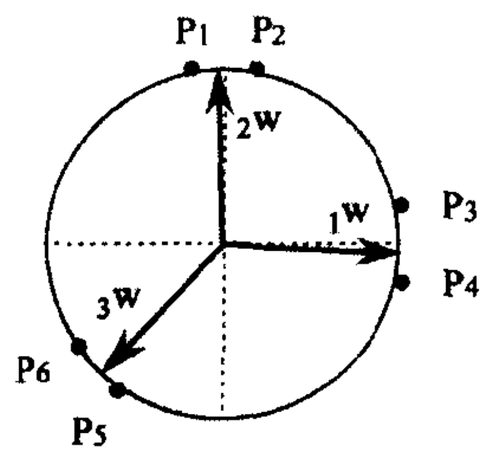



图 14-7 最终的权值

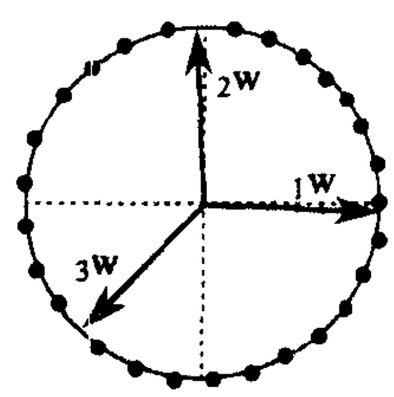



图 14-8

一旦神经网络学会了如何将输入向量分类，那么对于新向量它也将同样分类，如图14-8所示。阴影表示每个神经元将作出响应的区域。竞争层通过使权值向量最接近输入向量 $\mathbf{p}$ 的神经元的输出为 1 为每个输入向量 $\mathbf{p}$ 指定给这些类中的一个。

*试验竞争学习请使用 Neural Network Design Demonstration Competitive Learning* (**nnd14cl**)。

2. *竞争层中存在的问题*

竞争网络能够进行有效的自适应分类，但它仍存在一些问题。第一个问题就是学习速度的选择使得不得不在学习速度和最终权值向量的稳定性之间进行折衷(见图 14-9)。一个接近 0 的学习速度意味着慢速的学习。然而，一旦权值向量到达一个簇的中心，它将保持在中心附近。

相反，接近 1.0 的学习速度将导致快速学习。然而，一旦权值向量到达一个簇，它将作为它所代表的簇中的不同向量来回振荡(见图 14-10)。

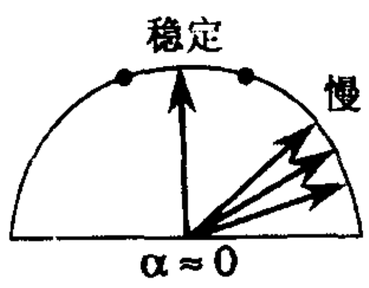



图 14-9

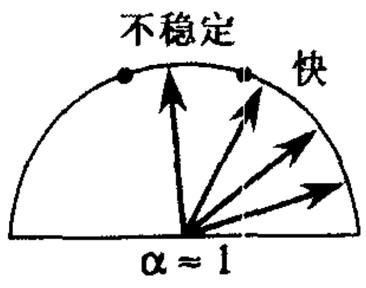



图 14-10

有时这种在快速学习和稳定性之间的折衷能够带来好处。初始训练对于快速学习可以用大的学习速度来完成。然后学习速度可以随着训练的进程而逐步减少，以达到稳定的原型向量。令人遗憾的是如果网络需不断的对输入向量的新排列作出调整的话，这种技术就不起作

用了。

当簇彼此很靠近的时候，一种更为严重的稳定性问题产生了。在特定的情况下，一个形成某簇原型的权值向量会“侵入”另一个权值向量的领地，从而破坏目前的分类状况。 14-9

图 14-11 中的 4 个图说明了这个问题。两个输入向量(图(a)中用空心圆圈表示)被提交了几次。结果是代表中间和右边簇的权值向量移向了右边。最后右边簇的一个向量被中心权值向量重新分类。进一步的提交向量使中间向量移向右边，直到它“丢失”了一些它的向量才停止，这些丢失的向量成为左边权值向量所代表的类的一部分。

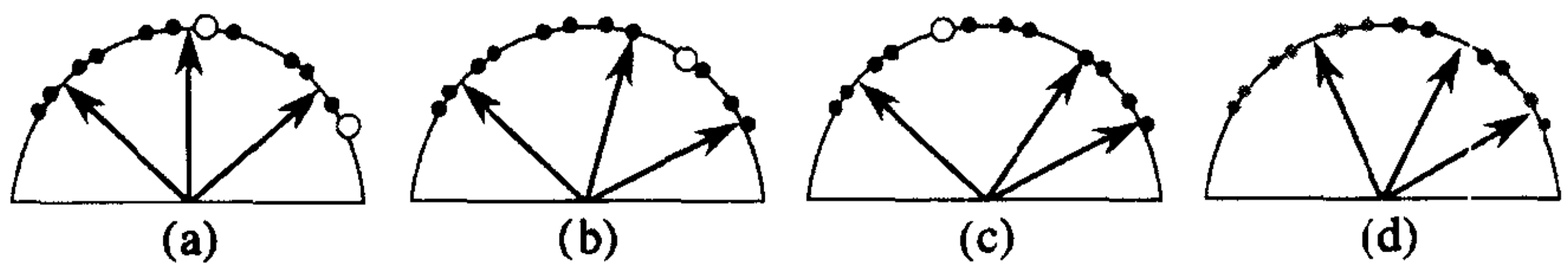



图 14-11 不稳定学习的例子

第三个问题是有时一个神经元的初始权值向量离输入向量太远以至于它从未在竞争中获胜，因此从来也得不到学习。这将产生一个毫无用处的“死”神经元。例如图 14-12，无论以什么次序把向量提交给网络，向下指的那个权值向量永远都得不到学习。这个问题的一个解决办法是给每个神经元的净输入加入一个负的偏置值。每次那个神经元竞争获胜则将偏置值减少。这将使一个经常竞争获胜的神经元获胜的机会减少。这种机制有时叫做“良心”(见习题 E14.4)。

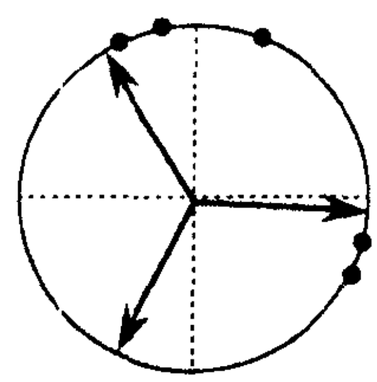


图 14-12

最终，一个竞争层有多少个神经元，就能够有多少个类。这对于某些应用将无法适用，尤其在事先并不知道簇的个数的时候。此外，对于竞争层，每个类在输入空间之中都含有一个凸区域。当在非凸区域或类是由不连接的区域所组成的时候，竞争层不能形成类。

本小节讨论的一些问题将由特征图及 LVQ 网络解决，这些将在本章的后一部分讨论，而自适应网络将在第 16 章中介绍。

### 14.2.3 生物学意义上的竞争层

在前面几章我们未曾提及神经元在一层之内是如何组织的(网络的拓扑结构)。在生物的神经网络中，神经元的典型情况是排列成二维层次，它们通过横向反馈密集地联结在一起。图 14-13 展示了以二维网格形式组织的 25 个神经元。 14-10

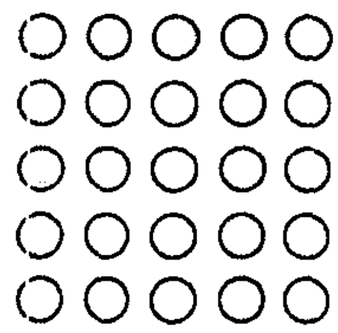


图 14-13

通常权值是联结的神经元之间的距离的函数。例如，Hamming 网络第二层的权值定义如下：

$$w_{ij} = \begin{cases} 1, & i = j \\ -\varepsilon, & i \neq j \end{cases} \tag{14.19}$$

等式(14.20)与等式(14.9)定义了同样的值，只是基于神经元之间的距离 $d_{ij}$。

$$w_{ij} = \begin{cases} 1, & d_{ij} = 0 \\ -\varepsilon, & d_{ij} > 0 \end{cases} \tag{14.20}$$

图 14-14 展示了等式(14.20)或式(14.19)所定义的权值。每个神经元 $i$ 都标以权值 $w_{ij}$，

即从它到神经元 $j$ 的权值。

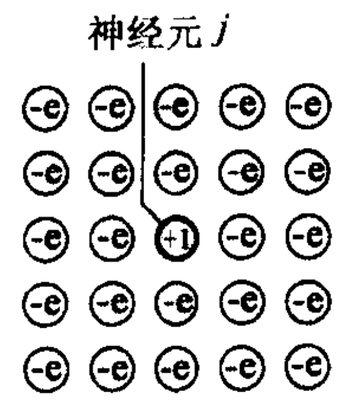
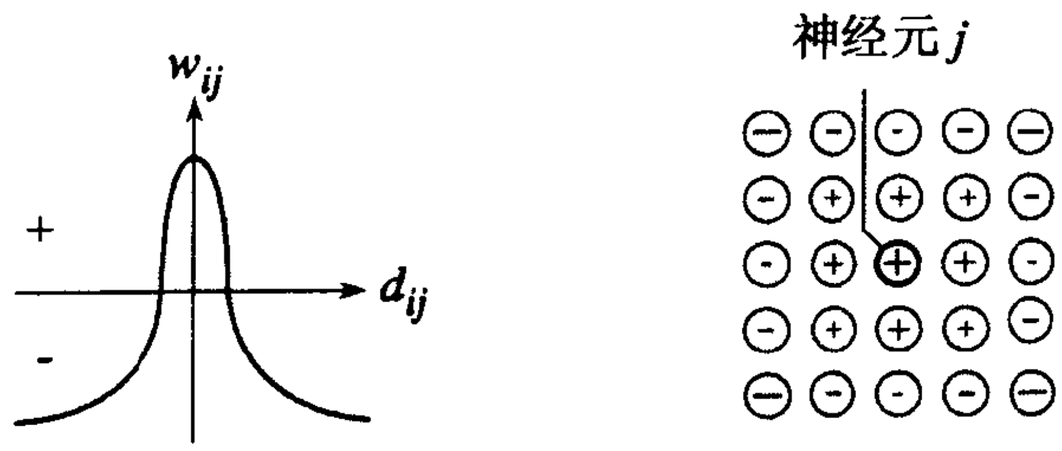



图 14-14

**加强中心/抑制周围** 加强中心/抑制周围常被用来描述如下神经元之间的一种联结方式：每个神经元加强自身(中心)，并且同时抑制周围的神经元。

这是生物学竞争层中的一种天然的近似。在生物学中，大神经元不仅加强自己，同时也加强接近它的那些神经元。一般情况下，随着神经元之间的距离增加，从加强到抑制的转变是平滑地出现的。

**墨西哥草帽函数** 这种转变见图 14-15 中的左图。这是一种将神经元之间的距离与连接他们的权值相联系起来的函数。那些相近的神经元提供互相加强的连接，并且激励的幅度随着距离的增加而降低。超过一定距离，神经元将呈现一种抑制性联结，并且抑制随着距离的增加而增加。因为这个函数的形状，它被称为墨西哥草帽函数。图 14-15 中的右图是墨西哥草帽(加强中心/抑制周围)函数的一个二维图示，每个神经元 $i$ 都被标以符号及表示它到神经元 $j$ 的权值的相对强度 $w_{ij}$。




14-11

图 14-15 生物学上的加强中心/抑制周围层

生物竞争系统，除了在加强中心/抑制周围的联结模式下，从激励区域到抑制区域的转变是渐变以外，还是一种相对于 Hamming 网络的“胜者全得”竞争较为弱的竞争形式。生物网络通常不是单个神经元(竞争获胜者)活跃，而是在最为活跃的神经元的周围有活跃区。这部分是由于加强中心/抑制周围的联结方式以及非线性的反馈联结引起的(参见第 15 章对轮廓线增强的讨论)。

### 14.2.4 自组织特征图

**SOFM 网络 邻域** 为了模仿生物学系统的活跃区，并且不必实现非线性的加强中心/抑制周围的反馈联结，Kohonen 作了如下的简化设计。他的自组织特征图(SOFM)网络首先使用竞争层所用的同一过程来决出获胜的神经元 $i^*$；然后，在获胜神经元周围一定范围内的所有神经元的权值向量用 Kohohen 规则更新，

$$\begin{aligned} {}_i\mathbf{w}(q) &= {}_i\mathbf{w}(q-1) + \alpha(\mathbf{p}(q) - {}_i\mathbf{w}(q-1)) \\ &= (1-\alpha)\,{}_i\mathbf{w}(q-1) + \alpha\mathbf{p}(q), \quad i \in N_{i^*}(d) \end{aligned} \tag{14.21}$$

其中邻域 $N_{i^*}(d)$包含所有落在以获胜神经元 $i^*$ 为中心、半径为 d 的所有神经元的下标：

$$N_{i^*}(d) = \{j, d_{ij} \leqslant d\} \tag{14.22}$$

当向量 $\mathbf{p}$ 被提交，获胜神经元的权值和其邻域内的神经元的将向 $\mathbf{p}$ 移动。结果是多次提交结束之后，邻域内的神经元将通过学习而拥有彼此相像的学习向量。

为了展示邻域的概念，请考虑图 14-16 中的两幅图。左边的图说明围绕神经元 13、半径为 1 的二维邻域；右边的图表示半径为 2 的邻域。

这两个邻域的定义如下：

$$N_{13}(1) = \{8,12,13,14,18\} \tag{14.23}$$

$$N_{13}(2) = \{3,7,8,9,11,12,13,14,15,17,18,19,23\} \tag{14.24}$$

14-12

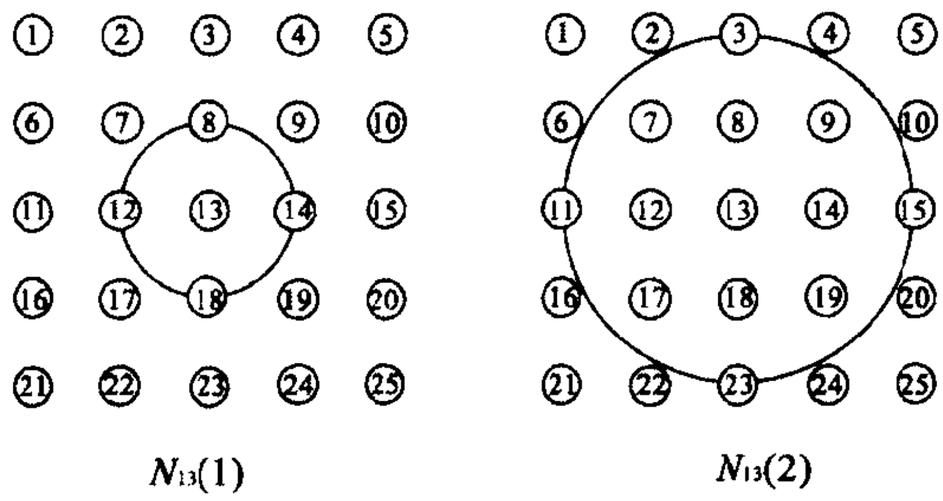



图 14-16 邻域

我们必须提及的是 SOFM 中的神经元不必排列成二维的形式，一维或者三维甚至多维的排列都是可能的。对于一维的 SOFM，神经元在半径为 1 时只有 2 个邻域(当该神经元位于线之端点时只有一个邻域)。当然也可以用不同的方法来定义距离，例如，Kohonen 为了更有效地实现曾建议使用矩形或六边形的邻域。神经网络的性能对邻域的确切形状并不敏感。

现在我们演示 SOFM 神经网络是如何工作的。图 14-17 表示一个特征图以及其神经元的二维拓扑结构。

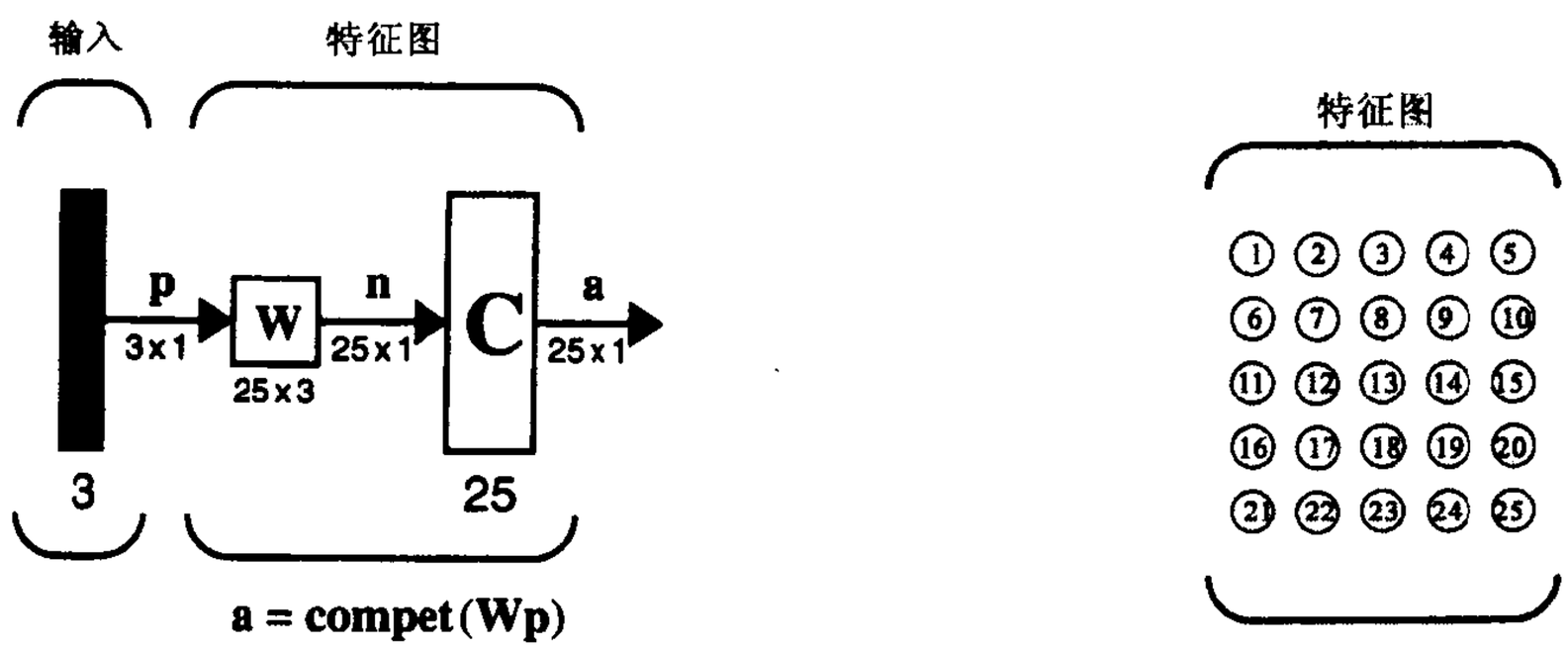
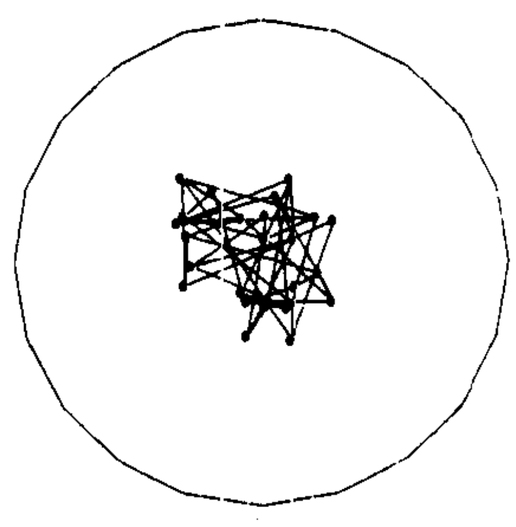



图 14-17 自组织特征图

图 14-18 展示了特征图的初始权值向量，每个三元素权值向量都用球体上的一个点表示(权值已经规格化，都能够落在球面之上)。邻域内的神经元都用线连接起来，因而可以看到网络拓扑结构在输入空间中是如何安排的。



14-13

图 14-18

图 14-19 展示了一个球面上的方形区域。我们将从这个区域中随机抽取向量，以提交给特征图。

每当一个向量被提交，具有最近权值向量的那个神经元将竞争获胜。获胜神经元及其邻域内的神经元将移动它们的

权值向量从而离输入向量更近一些(并且互相靠近)。本例中我们使用的邻域半径为 1。

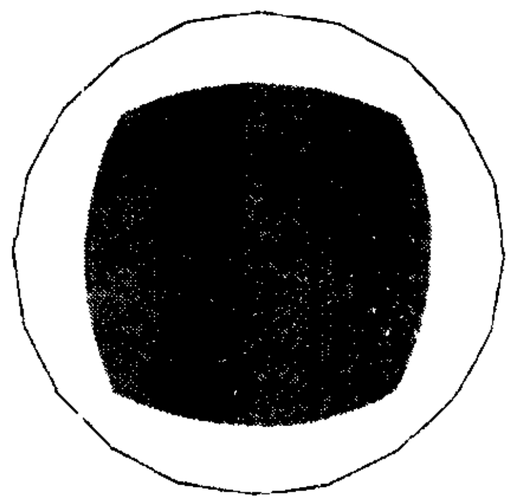


图 14-19

权值向量有两个趋势：首先，它们随着更多的输入向量被提交而分布到整个输入空间。其次，它们移向邻域内的神经元。这两个趋势共同作用使神经元在那一层重新排列，从而最终输入空间得到分类。

图 14-20 所示的一系列图展示了 25 个的神经元如何在活动的输入空间内展开，并自组织以匹配其拓扑结构。

在这个例子中，输入向量以等概率产生于输入空间的任何

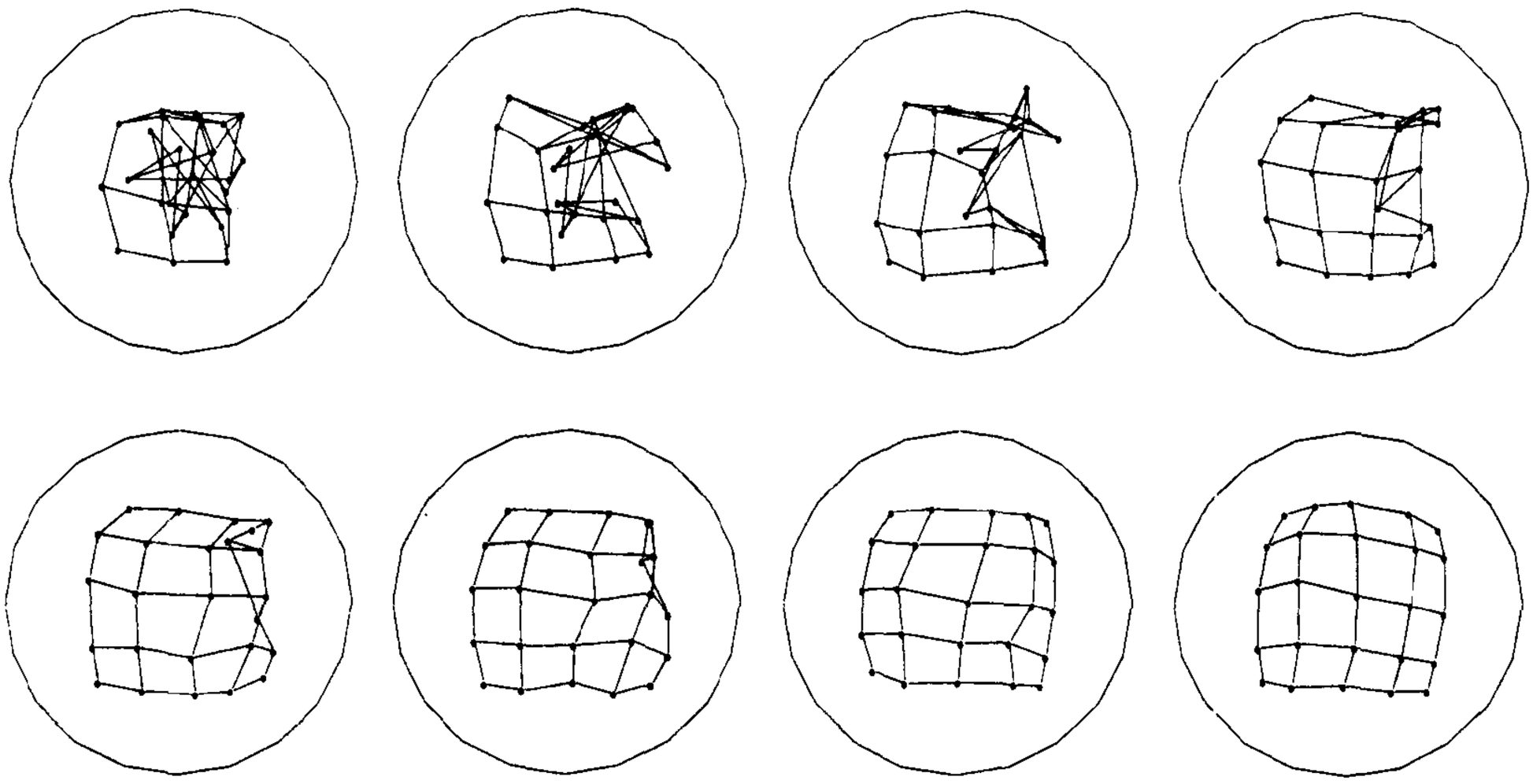


图 14-20 自组织，每张图迭代 250 次

位置。因此，神经元能够将输入空间分成大致相等的区域。

14-14 图 14-21 提供了更多的关于输入区域及自组织之后的结果特征图的例子。

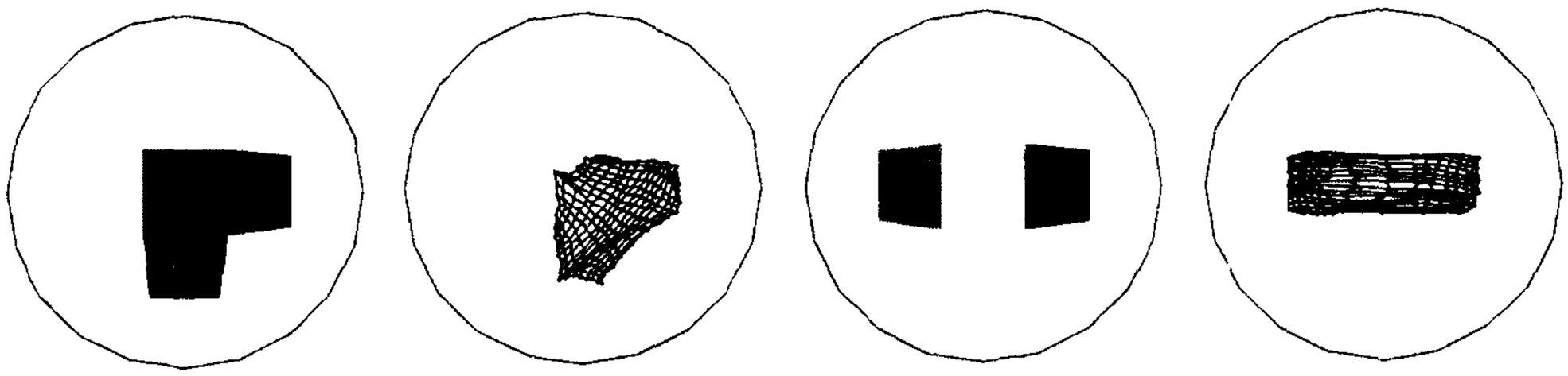


图 14-21 特征图训练的其他例子

有时特征图不能够与他们输入空间的拓扑结构相匹配。这种情况通常发生在网络的两部分与输入空间的独立部分的拓扑结构相匹配，但网络在这两部分之间却发生了扭曲，见图 14-22 中的例子。这种扭曲现象不大可能消除，因为网络的两端都已经形成对不同区域的稳定的分类。

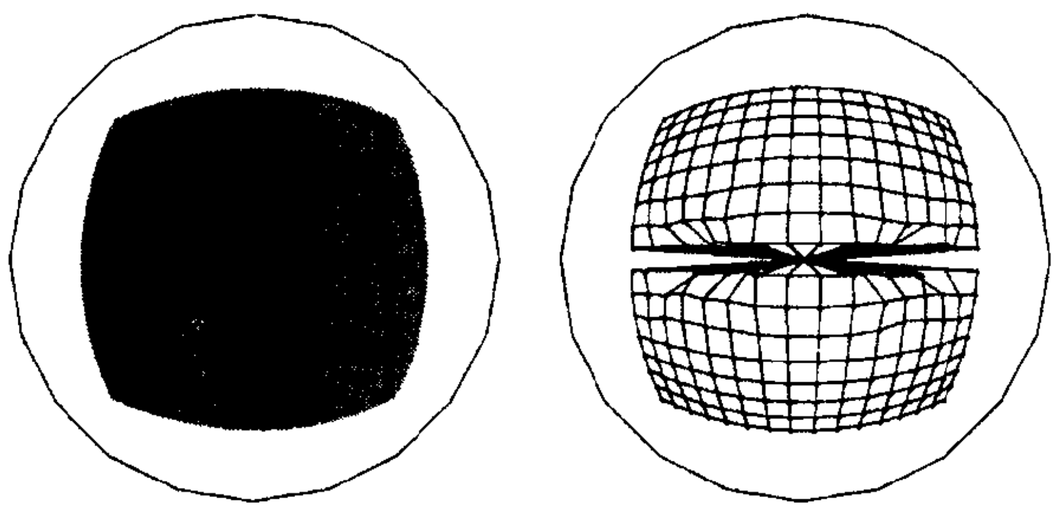


图 14-22 带扭曲的特征图

改进特征图

到目前为止，我们仅仅讨论训练特征图的最基本算法。现在考虑几种能够加速自组织过程并且使它更加可靠的技术。

一个改进自组织图的方法是在训练过程中改变邻域的大小。开始，邻域的半径 $d$ 设置得较大。随着训练的进行，$d$ 逐渐减少，直到最终只包含竞争获胜的神经元。这种方法加速自组织并且极不可能在网络中造成扭曲。

学习速度也可以随时间而改变。初始学习速度为 1 使神经元能够很快地学习提供的向量。在训练过程中，学习速度逐渐降至 0，于是学习变得稳定了。(在本章早些时候曾讨论过将这种技术用于竞争层。) 14-15

另外一种加速自组织的改进是使竞争获胜的神经元有比其邻域内的神经元更大的学习速度。

最终，竞争层和特征图通常使用另外一种表达式作为净输入。它们能够直接计算输入向量与原型向量之间的距离而不采用计算内积的方法。这种利用距离的方法，优点在于输入向量不必规格化。这种改进的净输入表达式将在下一节的 LVQ 网络中介绍。

试验特征图请使用 *Neural Network Design Demonstration* ***1-D Feature Maps*** (**nnd14fm1**)和 ***2-D Feature Maps*** (**nnd14fm2**)。

### 14.2.5 学习向量量化

这章我们讨论的最后一种神经网络是学习向量量化(LVQ)网络，见图 14-23 所示。LVQ 神经网络是一种混合网络。通过有监督及无监督的学习来形成分类。

在 LVQ 网络中，第一层的每个神经元都指定给某个类，常常几个神经元被指定给同一类。每类再被指定给第二层的一个神经元。第一层神经元的个数 $S^1$，与第二层神经元的个数 $S^2$ 至少相同，并且通常要大一些。 14-16

和竞争网络一样，LVQ 网络的第一层的每个神经元学习原型向量，它可以对输入空间的区域分类。然而，不是通过计算内积得到输入和权值向量中最接近者，我们通过直接计算距离的方法来模拟 LVQ 网络。直接计算距离的一个优点是向量不必先规格化，当向量规格化了，无论是采用计算内积的方法还是直接计算距离，网络的响应将是相同的。

LVQ 网络的第一层的净输入是

$$n_i^1 = -\|{}_i\mathbf{w}^1 - \mathbf{p}\| \tag{14.25}$$

或者，用向量形式

$$\mathbf{n}^1 = -\begin{bmatrix} \|{}_1\mathbf{w}^1 - \mathbf{p}\| \\ \|{}_2\mathbf{w}^1 - \mathbf{p}\| \\ \vdots \\ \|{}_{S^1}\mathbf{w}^1 - \mathbf{p}\| \end{bmatrix} \tag{14.26}$$

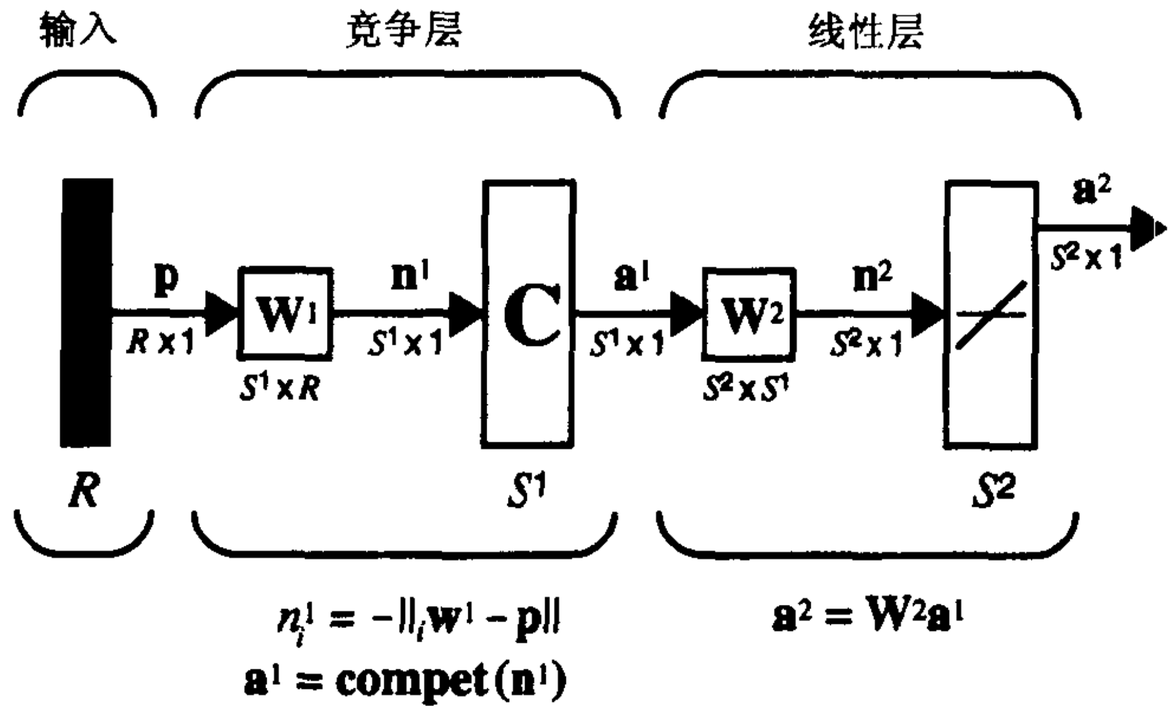



图 14-23 LVQ 网络

LVQ 网络第一层的输出是

$$\mathbf{a}^1 = \mathbf{compet}(\mathbf{n}^1) \tag{14.27}$$

因此那种权值向量与输入向量最为接近的神经元的输出将为 1，而其他神经元的输出为 0。

**子类** 迄今，LVQ 网络与竞争网络的特性几乎相同(至少对规格化向量)。然而，其解释方面有区别。对于竞争网络，有非零输出的神经元表示输入向量属于那个类。而对于 LVQ 网络，竞争获胜的神经元表示的是一个子类而非一个类。一个类可能由几个不同的神经元(子类)组成。

LVQ 网络的第二层将子类组合成一个类。这是通过 $\mathbf{W}^2$ 矩阵来实现的。$\mathbf{W}^2$ 矩阵的列代表子类，而行则代表类。$\mathbf{W}^2$ 的每列仅有一个 1，其他元素都设置为 0。1 出现的行表明这个子类属于那个类。

$$(w_{ki}^2 = 1) \Rightarrow \text{子类 } i \text{ 是类 } k \text{ 的一部分} \tag{14.28}$$

这种将子类组合成为类的过程使得 LVQ 网络产生了复杂的类边界。一个标准的竞争层存在局限，即只能够创造凸的判定区域。LVQ 网络克服了这个局限。

14-17

1. LVQ 学习

LVQ 网络的学习结合了竞争学习和有监督的学习。正如所有有监督的学习算法一样，它需要一组正确网络行为的例子：

$$\{\mathbf{p}_1, \mathbf{t}_t\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \cdots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

每个目标向量除了一个是 1 以外必须全是 0。1 出现的行表示输入向量属于那个类。例如，如果有这样一个问题，必须将一个特别的三元素向量归类入四个类中的第二类，我们可以这样表达：

$$\left\{\mathbf{p}_1 = \begin{bmatrix} \sqrt{1/2} \\ 0 \\ \sqrt{1/2} \end{bmatrix}, \quad \mathbf{t}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}\right\} \tag{14.29}$$

在学习能够进行之前，把第一层的每个神经元指定给一个输出神经元。这样就产生了矩阵 $\mathbf{W}^2$。典型情况下，相同数量的隐藏神经元联结到每个输出神经元，因而每个类都能够由相同数量的凸区域组成。$\mathbf{W}^2$ 矩阵的所有元素都设置为 0，除了如下情况：

$$\text{如果隐含神经元 } i \text{ 是指定给类 } k\text{,那么设 } w_{ki}^2 = 1 \tag{14.30}$$

一旦定义了 $\mathbf{W}^2$，它将不会再改变了。隐藏权值 $\mathbf{W}^1$ 将用 Kohonen 规则的一个变化形式训练。

LVQ 学习规则以如下方式进行。在每次迭代过程，一个输入向量 $\mathbf{p}$ 被提供给网络，并且计算每个原型向量与 $\mathbf{p}$ 的距离。隐含的神经元进行竞争，神经元 $i^*$ 竞争获胜，$\mathbf{a}^1$ 的第 $i$ 个元素被设置为 1。接着 $\mathbf{a}^1$ 与 $\mathbf{W}^2$ 相乘从而得到最终输出 $\mathbf{a}^2$，也是只有一个非零元素 $k^*$，表明 $\mathbf{p}$ 是指定给 $k^*$ 类的。

Kohonen 规则被用在两个方面以改进 LVQ 网络的隐含层。首先，如果 $\mathbf{p}$ 分类正确的话，那么获胜的隐含神经元向 $\mathbf{p}$ 移动：

$$_{i^*}\mathbf{w}^1(q) = {}_{i^*}\mathbf{w}^1(q-1) + \alpha(\mathbf{p}(q) - {}_{i^*}\mathbf{w}^1(q-1)), \text{ 如果 } a_{k^*}^2 = t_{k^*} = 1 \tag{14.31}$$

其次，如果 $\mathbf{p}$ 被不正确归类，那么我们知道错误的隐含层神经元竞争获胜，因此，移动它的权值 $_{i^*}\mathbf{w}^1$ 远离 $\mathbf{p}$：

14-18

$$_{i^*}\mathbf{w}^1(q) = {}_{i^*}\mathbf{w}^1(q-1) - \alpha(\mathbf{p}(q) - {}_{i^*}\mathbf{w}^1(q-1)), \text{ 如果 } a_{k^*}^2 = 1 \neq t_{k^*} = 0 \tag{14.32}$$

结果是每个隐含神经元移向那些落入形成子类的类中的向量，而远离那些落入其他类中的向量。

让我们看一个 LVQ 训练的例子。我们训练 LVQ 网络来求解如下分类问题：

$$\text{类 1}:\left\{\mathbf{p}_1 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right\}, \quad \text{类 2}:\left\{\mathbf{p}_3 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \mathbf{p}_4 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}\right\} \tag{14.33}$$

正如图 14-24 所示，开始为每个输入指定目标向量：

$$\left\{\mathbf{p}_1 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right\}, \quad \left\{\mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{t}_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right\} \tag{14.34}$$

$$\left\{\mathbf{p}_3 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \mathbf{t}_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right\}, \quad \left\{\mathbf{p}_4 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{t}_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right\} \tag{14.35}$$

现在必须决定选择多少子类来组成这两个类中的每一类。如果让每个类是两个子类的联合，那么隐含层中最终将有四个神经元。输出层的权值矩阵将是

$$\mathbf{W}^2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{14.36}$$

$\mathbf{W}^2$ 将隐含神经元 1 和 2 与输出神经元 1 连结起来，将隐含神经元 3 和 4 与输出神经元 2

相连。每个类都将由 2 个凸区域组成。

$\mathbf{W}^1$ 的行向量最初被设置为随机值，见图 14-25。定义类 1 的两个隐含神经元的权值用空心圆圈标记，定义类 2 的权值用实心圆圈标记。这些权值是

$$ {}_1\mathbf{w}^1 = \begin{bmatrix} -0.543 \\ 0.840 \end{bmatrix}, \ {}_2\mathbf{w}^1 = \begin{bmatrix} -0.969 \\ -0.249 \end{bmatrix}, \ {}_3\mathbf{w}^1 = \begin{bmatrix} 0.997 \\ 0.094 \end{bmatrix}, \ {}_4\mathbf{w}^1 = \begin{bmatrix} 0.456 \\ 0.954 \end{bmatrix} \tag{14.37} $$

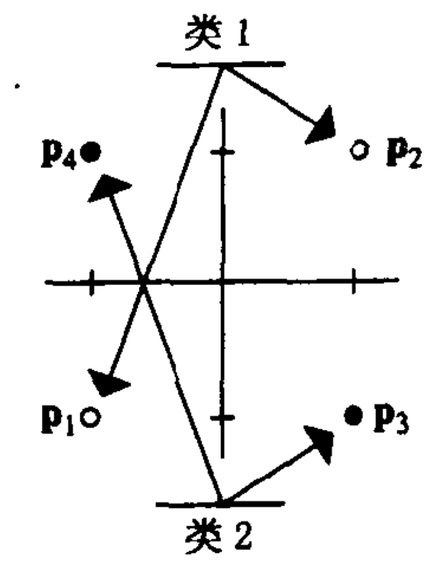



图 14-24

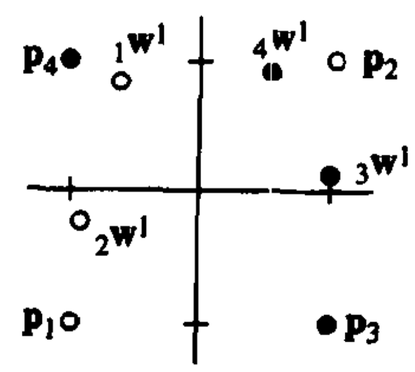



图 14-25

在训练过程中的每次迭代，我们提供一个输入向量，找出其响应，然后调整权值。在本 14-19 例中我们将从提交 $\mathbf{p}_3$ 开始。

$$ \mathbf{a}^1 = \mathbf{compet}(\mathbf{n}^1) = \mathbf{compet}\left(\begin{bmatrix} -\|{}_1\mathbf{w}^1 - \mathbf{p}_3\| \\ -\|{}_2\mathbf{w}^1 - \mathbf{p}_3\| \\ -\|{}_3\mathbf{w}^1 - \mathbf{p}_3\| \\ -\|{}_4\mathbf{w}^1 - \mathbf{p}_3\| \end{bmatrix}\right) $$

$$ = \mathbf{compet}\left(\begin{bmatrix} -\|[-0.543\ 0.840]^T - [1\ -1]^T\| \\ -\|[-0.969\ -0.249]^T - [1\ -1]^T\| \\ -\|[-0.997\ 0.094]^T - [1\ -1]^T\| \\ -\|[-0.456\ 0.954]^T - [1\ -1]^T\| \end{bmatrix}\right) = \mathbf{compet}\left(\begin{bmatrix} -2.40 \\ -2.11 \\ -1.09 \\ -2.03 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{14.38} $$

第三个隐含神经元的权值向量与 $\mathbf{p}_3$ 最近。为了确定这个神经元属于哪个类，令 $\mathbf{a}^1$ 与 $\mathbf{W}^2$ 相乘：

$$ \mathbf{a}^2 = \mathbf{W}^2\mathbf{a}^1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{14.39} $$

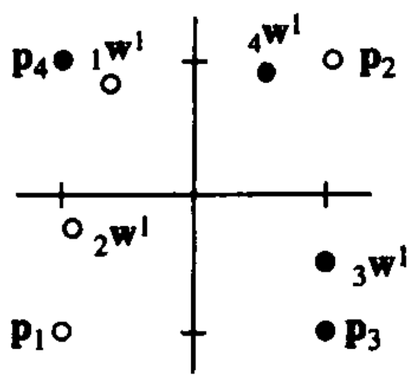
这个输出表明 $\mathbf{p}_3$ 属于类 2。这是正确的，于是 ${}_3\mathbf{w}^1$ 被移向 $\mathbf{p}_3$ 而更新。

$$ {}_3\mathbf{w}^1(1) = {}_3\mathbf{w}^1(0) + \alpha(\mathbf{p}_3 - {}_3\mathbf{w}^1(0)) $$

$$ = \begin{bmatrix} 0.997 \\ 0.094 \end{bmatrix} + 0.5\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0.997 \\ 0.094 \end{bmatrix}\right) = \begin{bmatrix} 0.998 \\ -0.453 \end{bmatrix} \tag{14.40} $$

图 14-26 中的左图表示 ${}_3\mathbf{w}^1$ 在第一次迭代之后的更新状况，右图表示算法收敛之后的权值。

图 14-26 中的右图也指明了输入空间如何被分类。那些归入类 1 的区域用浅灰色表示，归入类 2 的区域用深灰色表示。 14-20



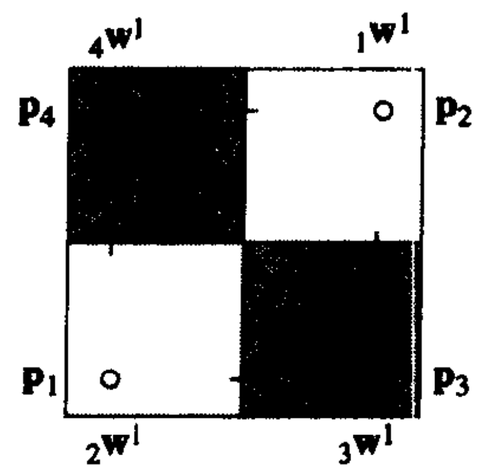




图 14-26 在第一次迭代和多次迭代之后

2. 改进的 LVQ 网络(LVQ2)

上面讨论的 LVQ 网络对许多问题都解决得较好，但却存在两种局限。首先，与竞争层的情况一样，有时一个 LVQ 网络的隐含神经元可能有使其从竞争获胜停止下来的初始权值。结果就是造成了一个无用的死神经元。这个问题用“良心”机制解决，这是前面在竞争层就已讨论过的技术，并请参见习题 14.4。

其次，由于有时初始向量的排列，在取某些初始向量的时候一个神经元的权值向量不得不经过一个它不代表的类的区域以到达它所代表的区域。由于这样的神经元的权值将被它必须经过的区域内的向量排斥，它可能无法通过，以至可能对吸引它的区域不能进行正确分类。这个问题通常通过如下改变 Kohonen 规则来解决。

如果隐含层中的获胜神经元对当前的输入不正确地归类，我们将它的权值向量从输入向量移开，正如以前所做的那样。然而，我们也调整与输入向量最接近的且归类正确神经元的权值。这种第二个神经元的权值将向输入向量移近。

**LVQ2** 当网络正确地分类一种输入向量时，只有一个神经元的权值被移向输入向量。然而如果输入向量被错误地归类，两个神经元的权值都将改变，一个权值向量被移开输入向量，另一个被移向输入向量。这种算法就是 LVQ2。

试验 LVQ2 网络请用 *Neural Network Design Demonstration LVQ1 Networks* (**nnd14lv1**)和 *LVQ2 Networks* (**nnd14lv2**)。 14-21

## 14.3 小结

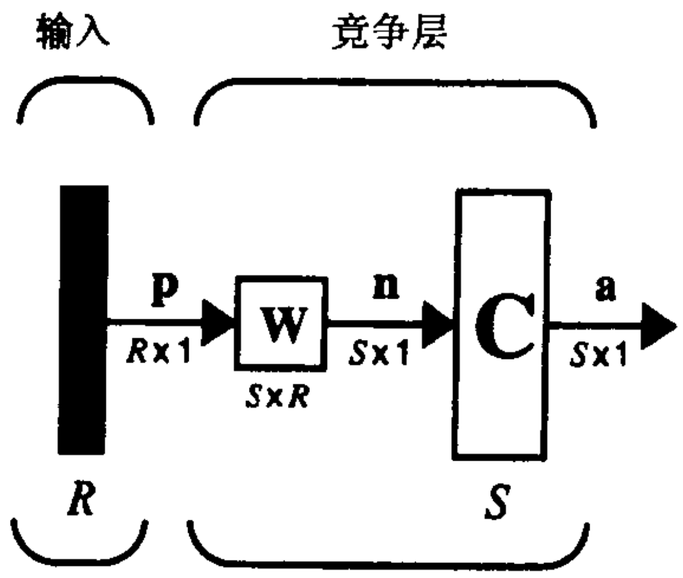
### 竞争层

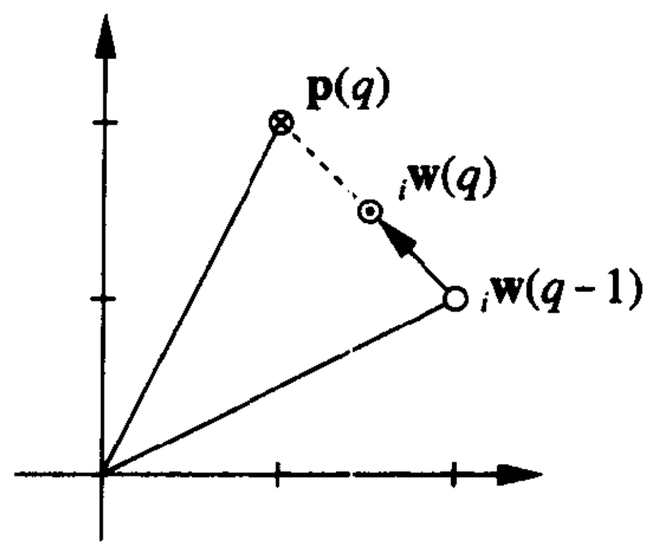
- **用 Kohonen 规则进行竞争学习**

$$
{}_{i^*}\mathbf{w}(q) = {}_{i^*}\mathbf{w}(q-1) + \alpha(\mathbf{p}(q) - {}_{i^*}\mathbf{w}(q-1)) = (1-\alpha)\,{}_{i^*}\mathbf{w}(q-1) + \alpha\mathbf{p}(q)
$$

$$
{}_{i^*}\mathbf{w}(q) = {}_{i^*}\mathbf{w}(q-1), \quad i \neq i^*
$$

其中 $i^*$ 是获胜的神经元。

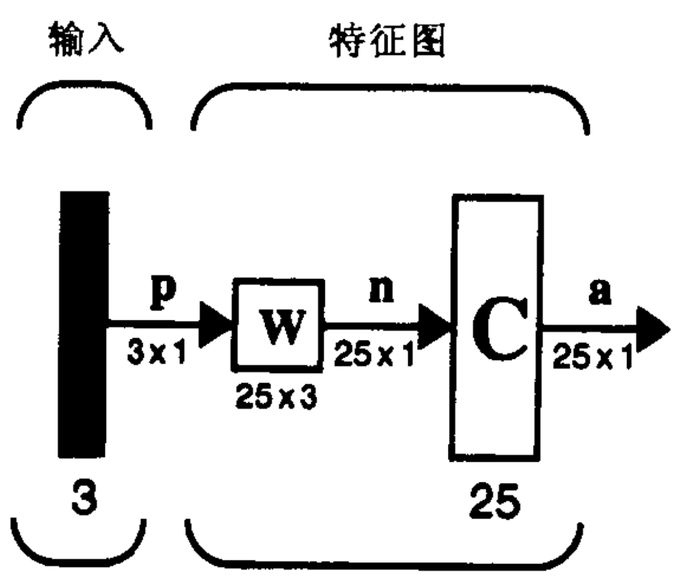


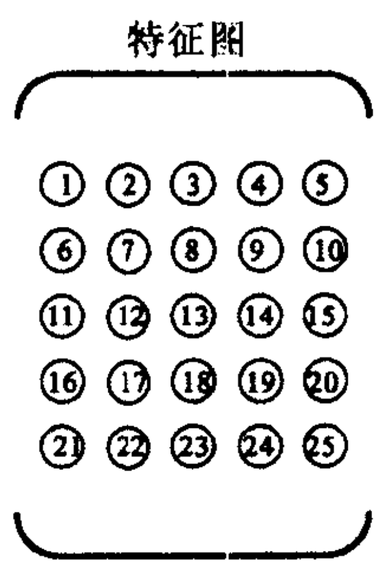




14-22

a = compet(Wp)

## 自组织特征图



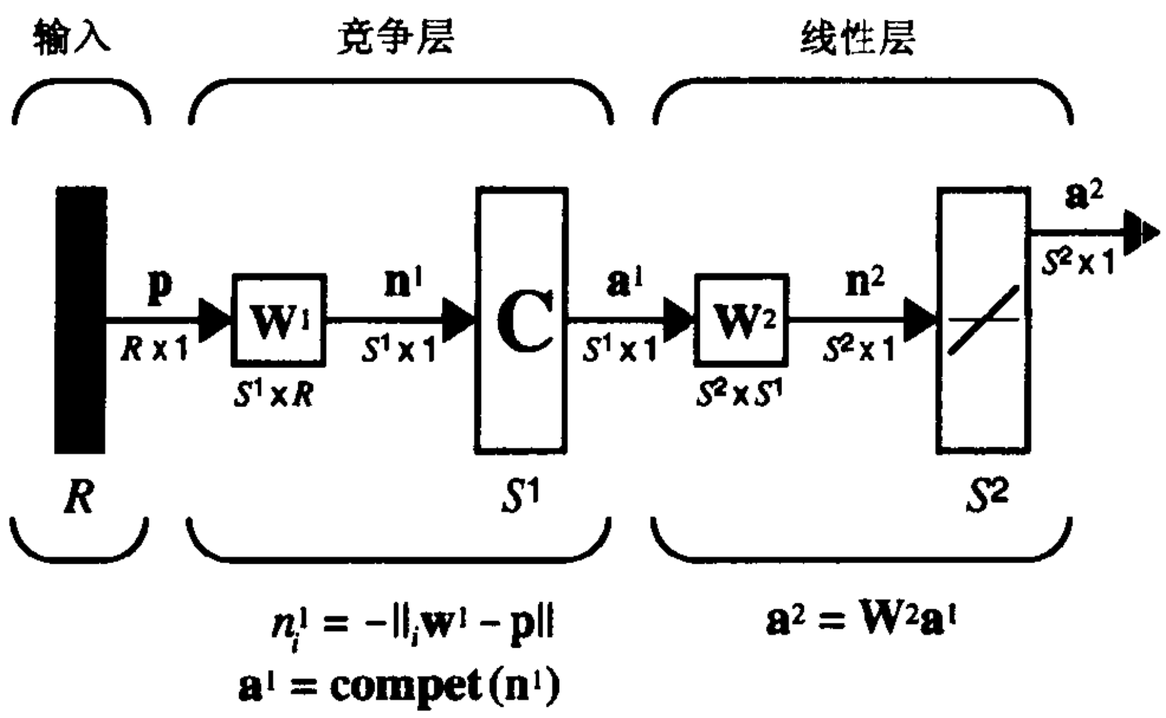

a = compet(Wp)

### 用 Kohonen 规则进行自组织

$$
\begin{aligned}
{}_i\mathbf{w}(q) &= {}_i\mathbf{w}(q-1) + \alpha(\mathbf{p}(q) - {}_i\mathbf{w}(q-1)) \\
&= (1-\alpha)\,{}_i\mathbf{w}(q-1) + \alpha\mathbf{p}(q)
\end{aligned}
\qquad (i \in N_{i^*}(d))
$$

$$N_i(d) = \{j, d_{ij} \leqslant d\}$$

## LVQ 网络




$(w_{ki}^2 = 1) \Rightarrow$ 子类 $i$ 是类 $k$ 的一部分

**用 Kohonen 规则进行 LVQ 网络学习**

$$
\begin{aligned}
{}_{i^*}\mathbf{w}^1(q) &= {}_{i^*}\mathbf{w}^1(q-1) + \alpha(\mathbf{p}(q) - {}_{i^*}\mathbf{w}^1(q-1)),\ 若\ a_{k^*}^2 = t_{k^*} = 1 \\
{}_{i^*}\mathbf{w}^1(q) &= {}_{i^*}\mathbf{w}^1(q-1) + \alpha(\mathbf{p}(q) - {}_{i^*}\mathbf{w}^1(q-1)),\ 若\ a_{k^*}^2 = 1 \neq t_{k^*} = 0
\end{aligned}
\tag{14-23}
$$

## 14.4 例题

**P14.1** 图 14-27 表示规格化向量的几个簇。设计图 14-28 中竞争网络的权值，使得它能够如图所示以最少的神经元数分类向量。

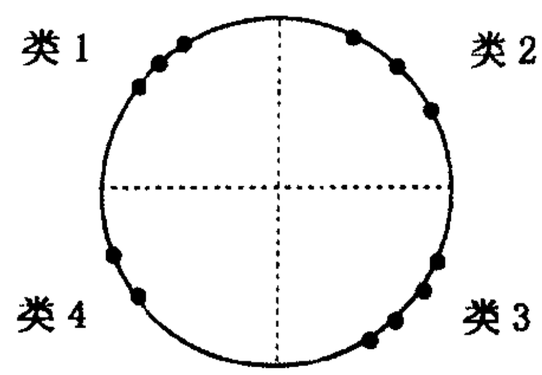



图 14-27 例题 P14.1 的输入向量簇

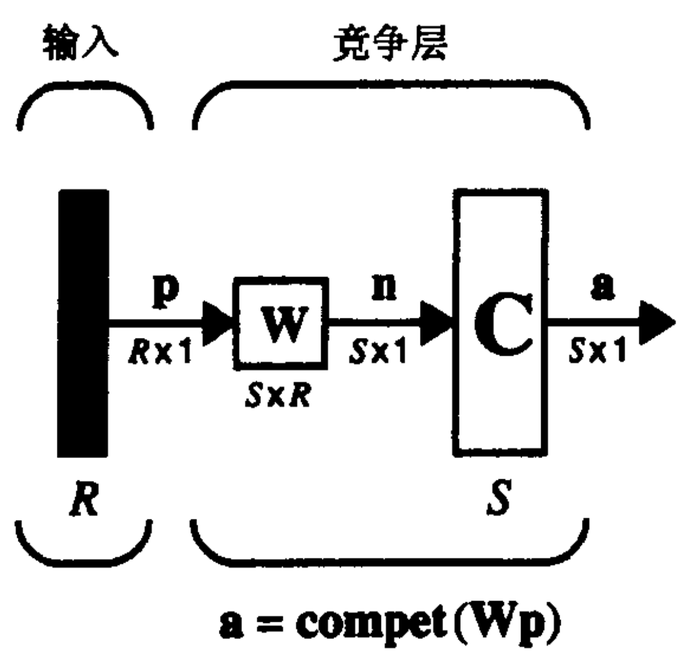



图 14-28 例题 P14.1 的竞争网络

重画图，使之表现出你所选择的权值和隔离每个类区域的判定边界。

**解**

因为有 4 个类需要定义，因而竞争层需要 4 个神经元。每个神经元的权值作为这个神经元所代表的类的原型。因此，对每个神经元我们将选择大致位于一个类中心的原型向量。

类 1，2，3 近似集中在以 45°角为倍数的位置。假定以下三个向量已经规格化(正如竞争网络所需要的那样)并且指向正确的方向。 14-24

$$
{}_1\mathbf{w} = \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix},\quad {}_2\mathbf{w} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix},\quad {}_3\mathbf{w} = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}
$$

第 4 簇的中心离纵轴的距离大致是离横轴距离的 2 倍。结果规格化的权值向量是

$$
{}_4\mathbf{w} = \begin{bmatrix} 2/\sqrt{5} \\ -1/\sqrt{2} \end{bmatrix}
$$

竞争层的权值矩阵刚好就是转置的原型向量的矩阵：

$$\mathbf{W}=\begin{bmatrix}{}_1\mathbf{w}^T\\{}_2\mathbf{w}^T\\{}_3\mathbf{w}^T\\{}_4\mathbf{w}^T\end{bmatrix}=\begin{bmatrix}-1/\sqrt{2} & 1/\sqrt{2}\\1/\sqrt{2} & 1/\sqrt{2}\\1/\sqrt{2} & -1/\sqrt{2}\\-2/\sqrt{5} & -1/\sqrt{5}\end{bmatrix}$$

我们用箭头画出这些权值向量，并且等分相邻的权值向量之间的弧以得到各个类的区域，这就是图 14-29。

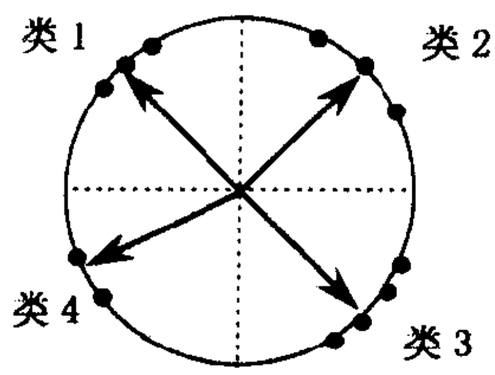



图 14-29 例题 P14.1 的最终分类结果

**P14.2** 图 14-30 表示一个由 3 个神经元组成的竞争网络层的三个输入向量及三个初始权值向量。以下是权值输入向量：

$$\mathbf{p}_1=\begin{bmatrix}-1\\0\end{bmatrix},\quad \mathbf{p}_2=\begin{bmatrix}0\\1\end{bmatrix},\quad \mathbf{p}_3=\begin{bmatrix}1/\sqrt{2}\\1/\sqrt{2}\end{bmatrix}$$

14-25 三个权值向量的初始值是

$${}_1\mathbf{W}=\begin{bmatrix}0\\-1\end{bmatrix},\quad {}_2\mathbf{W}=\begin{bmatrix}-2/\sqrt{5}\\1/\sqrt{5}\end{bmatrix},\quad {}_3\mathbf{W}=\begin{bmatrix}-1/\sqrt{5}\\2/\sqrt{5}\end{bmatrix}$$

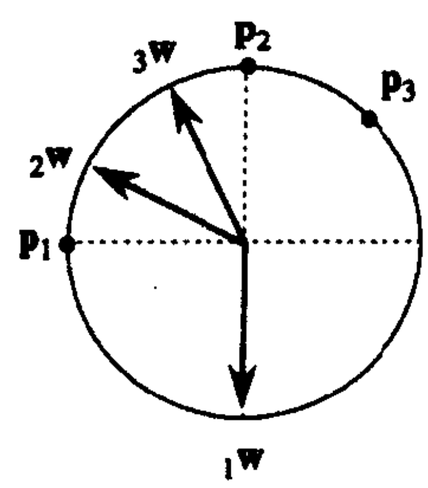



图 14-30 例题 P14.2 的输入向量及初始权值向量

计算用 Kohonen 规则训练竞争网络的结果权值，其中学习速度 $\alpha=0.5$，以下述序列作为输入：

$$\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3,\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3$$

**解**

首先我们用权值向量组成权值矩阵

$$\mathbf{W}=\begin{bmatrix}0 & -1\\-2/\sqrt{5} & 1/\sqrt{5}\\-1/\sqrt{5} & 2/\sqrt{5}\end{bmatrix}$$

然后，提交第一个向量 $\mathbf{p}_1$：

$$\mathbf{a} = \mathbf{compet}(\mathbf{W}\mathbf{p}_1) = \mathbf{compet}\left(\begin{bmatrix} 0 & -1 \\ -2/\sqrt{5} & 1/\sqrt{5} \\ -1/\sqrt{5} & 2/\sqrt{5} \end{bmatrix}\begin{bmatrix} -1 \\ 0 \end{bmatrix}\right) = \mathbf{compet}\left(\begin{bmatrix} 0 \\ 0.894 \\ 0.447 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

第二个神经元响应，因为$_2\mathbf{w}$离 $\mathbf{p}_2$ 最近，因此，我们用 Kohonen 规则更新$_2\mathbf{w}$:

$$_2\mathbf{w}^{new} = {}_2\mathbf{w}^{old} + \alpha(\mathbf{p}_1 - {}_2\mathbf{w}^{old}) = \begin{bmatrix} -2/\sqrt{5} \\ 1/\sqrt{5} \end{bmatrix} + 0.5\left(\begin{bmatrix} -1 \\ 0 \end{bmatrix} - \begin{bmatrix} -2/\sqrt{5} \\ 1/\sqrt{5} \end{bmatrix}\right) = \begin{bmatrix} -0.947 \\ 0.224 \end{bmatrix}$$

图 14-31 显示新的$_2\mathbf{w}$ 向 $\mathbf{p}_1$ 移近了。

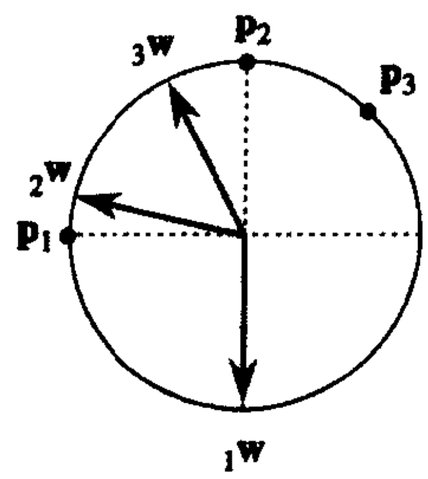



图 14-31

现在对 $\mathbf{p}_2$ 重复上述过程。 14-26

$$\mathbf{a} = \mathbf{compet}(\mathbf{W}\mathbf{p}_2) = \mathbf{compet}\left(\begin{bmatrix} 0 & -1 \\ -0.947 & 0.224 \\ -1/\sqrt{5} & 2/\sqrt{5} \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix}\right) = \mathbf{compet}\left(\begin{bmatrix} -1 \\ 0.224 \\ 0.894 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

第三个神经元获胜，因而其权值移近 $\mathbf{p}_2$:

$$_3\mathbf{w}^{new} = {}_3\mathbf{w}^{old} + \alpha(\mathbf{p}_2 - {}_3\mathbf{w}^{old}) = \begin{bmatrix} -1/\sqrt{5} \\ 2/\sqrt{5} \end{bmatrix} + 0.5\left(\begin{bmatrix} 0 \\ 1 \end{bmatrix} - \begin{bmatrix} -1/\sqrt{5} \\ 2/\sqrt{5} \end{bmatrix}\right) = \begin{bmatrix} -0.224 \\ 0.947 \end{bmatrix}$$

现在提交 $\mathbf{p}_3$:

$$\mathbf{a} = \mathbf{compet}(\mathbf{W}\mathbf{p}_3) = \mathbf{compet}\left(\begin{bmatrix} 0 & -1 \\ -0.947 & 0.224 \\ -0.224 & 0.947 \end{bmatrix}\begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}\right) = \mathbf{compet}\left(\begin{bmatrix} -0.707 \\ -0.512 \\ 0.512 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

第三个神经元再次获胜:

$$_3\mathbf{w}^{new} = {}_3\mathbf{w}^{old} + \alpha(\mathbf{p}_2 - {}_3\mathbf{w}^{old}) = \begin{bmatrix} -0.224 \\ 0.947 \end{bmatrix} + 0.5\left(\begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} - \begin{bmatrix} -0.224 \\ 0.947 \end{bmatrix}\right) = \begin{bmatrix} 0.2417 \\ 0.8272 \end{bmatrix}$$

再将 $\mathbf{p}_1$ 至 $\mathbf{p}_3$ 提交后，神经元 2 将会再获胜一次，而神经元 3 会获胜三次。最终的权值是

$$\mathbf{W} = \begin{bmatrix} 0 & -1 \\ -0.947 & 0.118 \\ 0.414 & 0.8103 \end{bmatrix}$$

最终的权值见图 14-32。

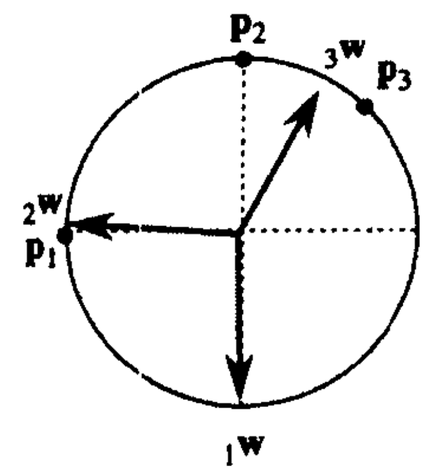



图 14-32

注意$_2\mathbf{w}$ 几乎学会了 $\mathbf{p}_1$，而$_3\mathbf{w}$指向 $\mathbf{p}_2$ 与 $\mathbf{p}_3$ 之间。另一个权值向量$_1\mathbf{w}$ 从来没有被更新。第一个神经元，因从未在竞争中获胜而成为

一个死神经元。

**P14.3** 考虑图 14-33 中所示的输入向量及初始权值。用 Kohonen 规则训练竞争网络，使
14-27 这些向量分类成簇，其中学习速度 $\alpha=0.5$。当每个输入向量都提交一次之后(按所示顺序进行)，在图上找出权值的位置。

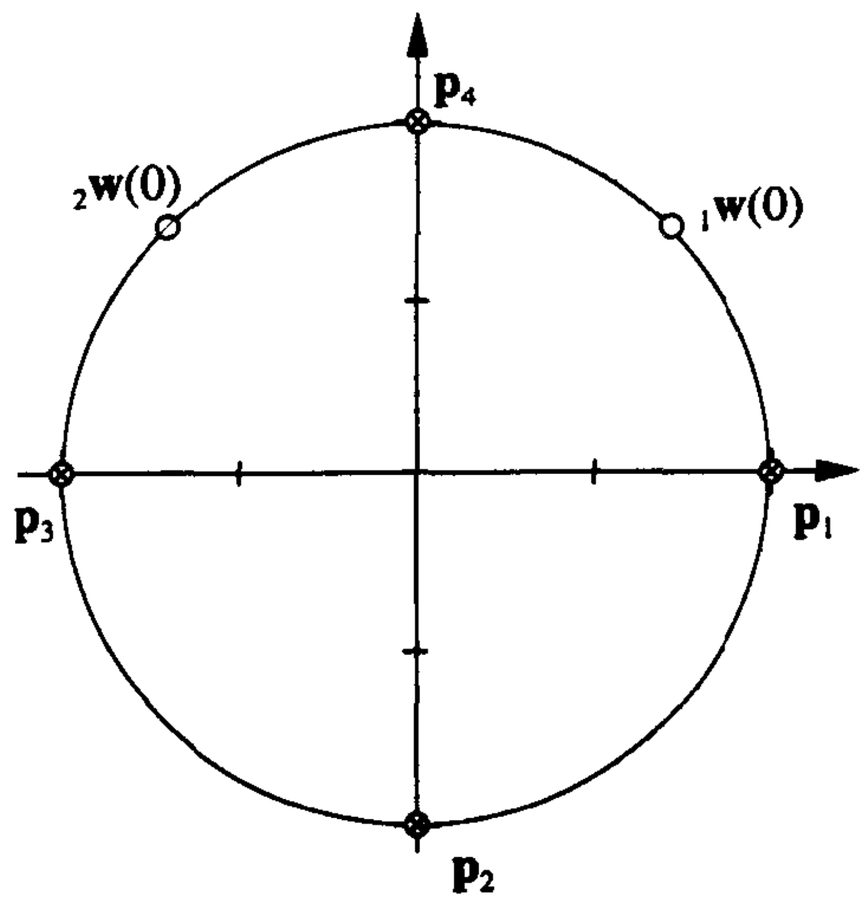



图 14-33 例题 P14.3 的输入向量和初始权值

**解**

这个问题可以不用计算而通过作图的方法解决，结果见图 14-34。

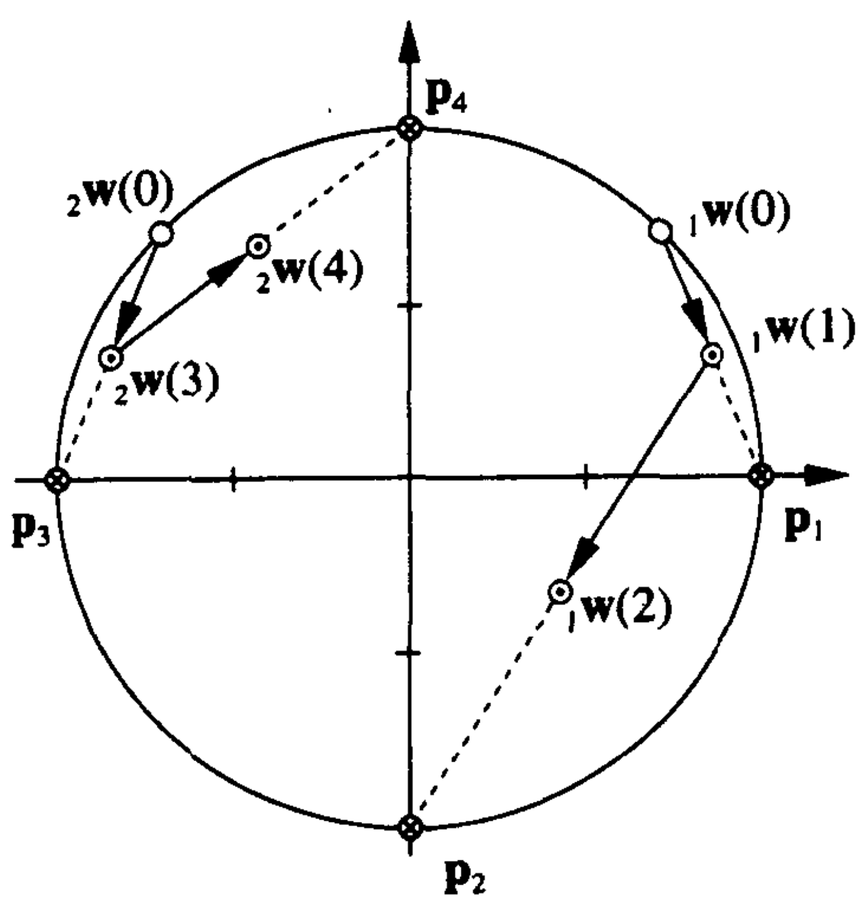



图 14-34 例题 P14.3 的解答

输入向量 $\mathbf{p}_1$ 首先被提交，权值向量 ${}_1\mathbf{w}$ 离 $\mathbf{p}_1$ 最近，因而神经元 1 竞争获胜而且 ${}_1\mathbf{w}$ 向 $\mathbf{p}_1$ 移近
14-28 一半距离(因为 $\alpha=0.5$)。然后，$\mathbf{p}_2$ 被提交，神经元 1 再次获胜，${}_1\mathbf{w}$ 再向 $\mathbf{p}_2$ 移近一半距离。在前两次迭代中，${}_2\mathbf{w}$ 没有改变。

第三次迭代 $\mathbf{p}_3$ 被提交。这次 ${}_2\mathbf{w}$ 竞争获胜并向 $\mathbf{p}_3$ 移近一半距离。第四次迭代 $\mathbf{p}_4$ 被提交，

神经元 2 再次获胜，权值向量$_2$**w** 向 $\mathbf{p}_4$ 移近一半距离。

如果我们继续训练这个网络，神经元 1 将会归类输入向量 $\mathbf{p}_1$ 及 $\mathbf{p}_2$，神经元 2 归类输入向量 $\mathbf{p}_3$ 及 $\mathbf{p}_4$。如果提交输入向量的顺序不同，最后的分类结果是否也会不同?

**P14.4** 本章所讨论的安排神经元的特征图都仅限于二维。图 14-35 所示的特征图由 9 个排成一维的神经元组成。

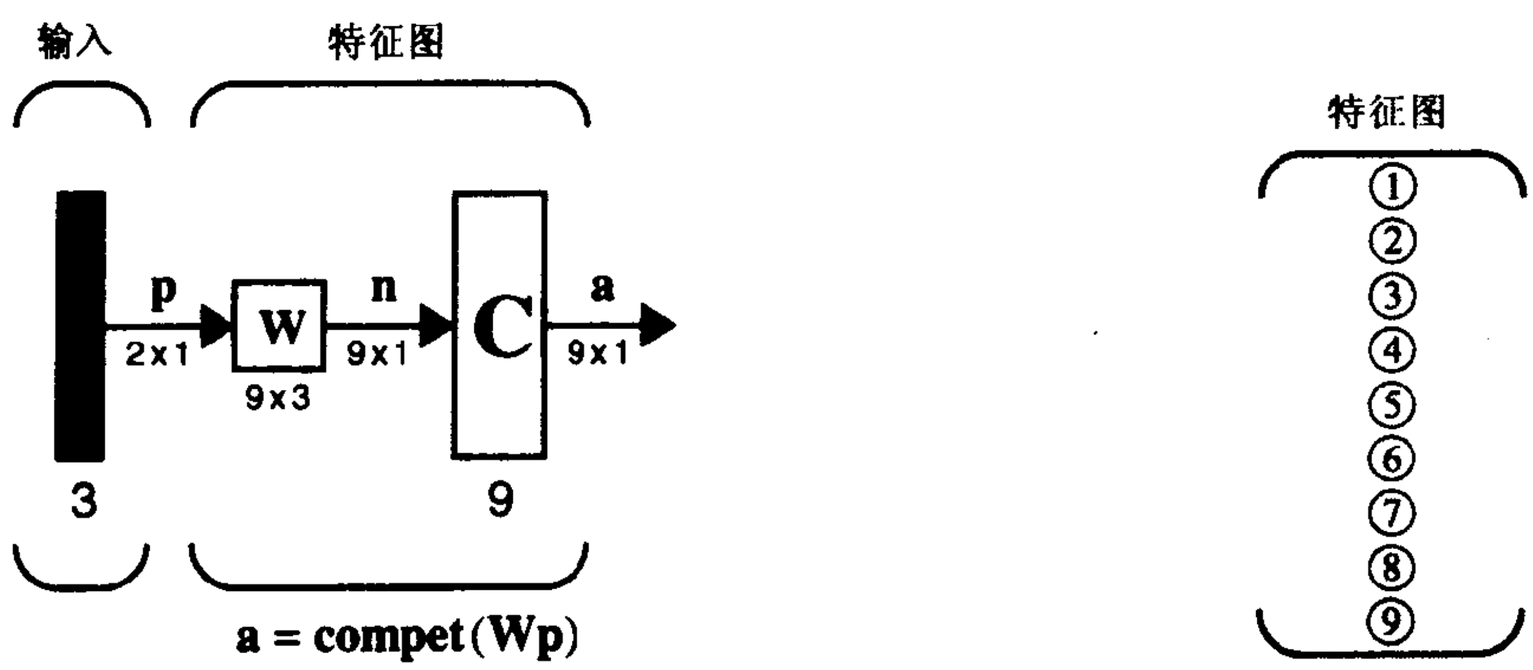



图 14-35 9 个神经元的特征图

根据如下的初始权值，画一权值向量图，并且将邻域神经元的权值用线连接起来。

$$\mathbf{W}=\begin{bmatrix}0.41 & 0.45 & 0.41 & 0 & 0 & 0 & -0.41 & -0.45 & -0.41\\ 0.41 & 0 & -0.41 & 0.45 & 0 & -0.45 & 0.41 & 0 & -0.41\\ 0.82 & 0.89 & 0.82 & 0.89 & 1 & 0.89 & 0.82 & 0.89 & 0.82\end{bmatrix}^T$$

用如下向量迭代一次来训练特征图，其中学习速度 $\alpha=0.1$，邻域半径为 1。重画对新权值矩阵的图。 14-29

$$\mathbf{p}=\begin{bmatrix}0.67\\0.07\\0.74\end{bmatrix}$$

**解**

原始权值的特征图见图 14-36。

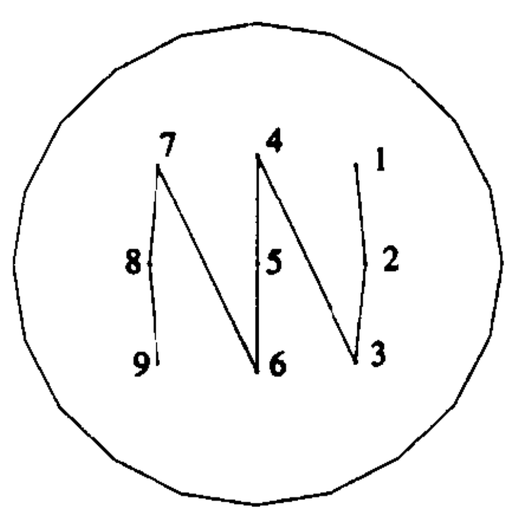



图 14-36 原始特征图

对网络提交 **p** 而开始更新网络。

$$\mathbf{a}=\mathbf{compet}(\mathbf{Wp})$$

$$= \mathbf{compet}\left(\begin{bmatrix} 0.41 & 0.45 & 0.41 & 0 & 0 & 0 & -0.41 & -0.45 & -0.41 \\ 0.41 & 0 & -0.41 & 0.45 & 0 & -0.45 & 0.41 & 0 & -0.41 \\ 0.82 & 0.89 & 0.82 & 0.89 & 1 & 0.89 & 0.82 & 0.89 & 0.82 \end{bmatrix}^T \begin{bmatrix} 0.67 \\ 0.07 \\ 0.74 \end{bmatrix}\right)$$

$$= \mathbf{compet}([0.91 \quad 0.96 \quad 0.85 \quad 0.70 \quad 0.74 \quad 0.63 \quad 0.36 \quad 0.36 \quad 0.3]^T)$$

$$= [0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$$

第二个神经元竞争获胜。从网络图示中可见，第二个神经元的邻域(半径为 1)包括神经元 1 和 3 。我们必须用 Kohonen 规则来更新这些权值。

14-30

$${}_1\mathbf{w}(1) = {}_1\mathbf{w}(0) + \alpha(\mathbf{p} - {}_1\mathbf{w}(0)) = \begin{bmatrix} 0.41 \\ 0.41 \\ 0.82 \end{bmatrix} + 0.1\left(\begin{bmatrix} 0.67 \\ 0.07 \\ 0.74 \end{bmatrix} - \begin{bmatrix} 0.41 \\ 0.41 \\ 0.82 \end{bmatrix}\right) = \begin{bmatrix} 0.43 \\ 0.37 \\ 0.81 \end{bmatrix}$$

$${}_2\mathbf{w}(1) = {}_2\mathbf{w}(0) + \alpha(\mathbf{p} - {}_2\mathbf{w}(0)) = \begin{bmatrix} 0.45 \\ 0 \\ 0.89 \end{bmatrix} + 0.1\left(\begin{bmatrix} 0.67 \\ 0.07 \\ 0.74 \end{bmatrix} - \begin{bmatrix} 0.45 \\ 0 \\ 0.89 \end{bmatrix}\right) = \begin{bmatrix} 0.47 \\ 0.01 \\ 0.88 \end{bmatrix}$$

$${}_3\mathbf{w}(1) = {}_3\mathbf{w}(0) + \alpha(\mathbf{p} - {}_3\mathbf{w}(0)) = \begin{bmatrix} 0.41 \\ -0.41 \\ 0.82 \end{bmatrix} + 0.1\left(\begin{bmatrix} 0.67 \\ 0.07 \\ 0.74 \end{bmatrix} - \begin{bmatrix} 0.41 \\ -0.41 \\ 0.82 \end{bmatrix}\right) = \begin{bmatrix} 0.43 \\ -0.36 \\ 0.81 \end{bmatrix}$$

图 14-37 展示了权值更新之后的特征图。

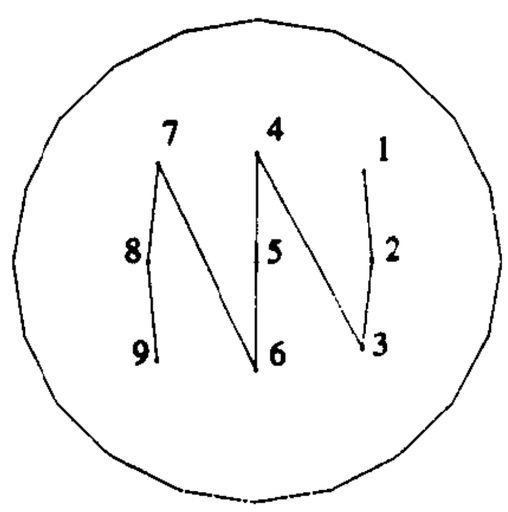



图 14-37 更新后的特征图

**P14.5** 给定图 14-38 所示的 LVQ 网络以及如下权值，画出构成每个类的输入空间的区域。

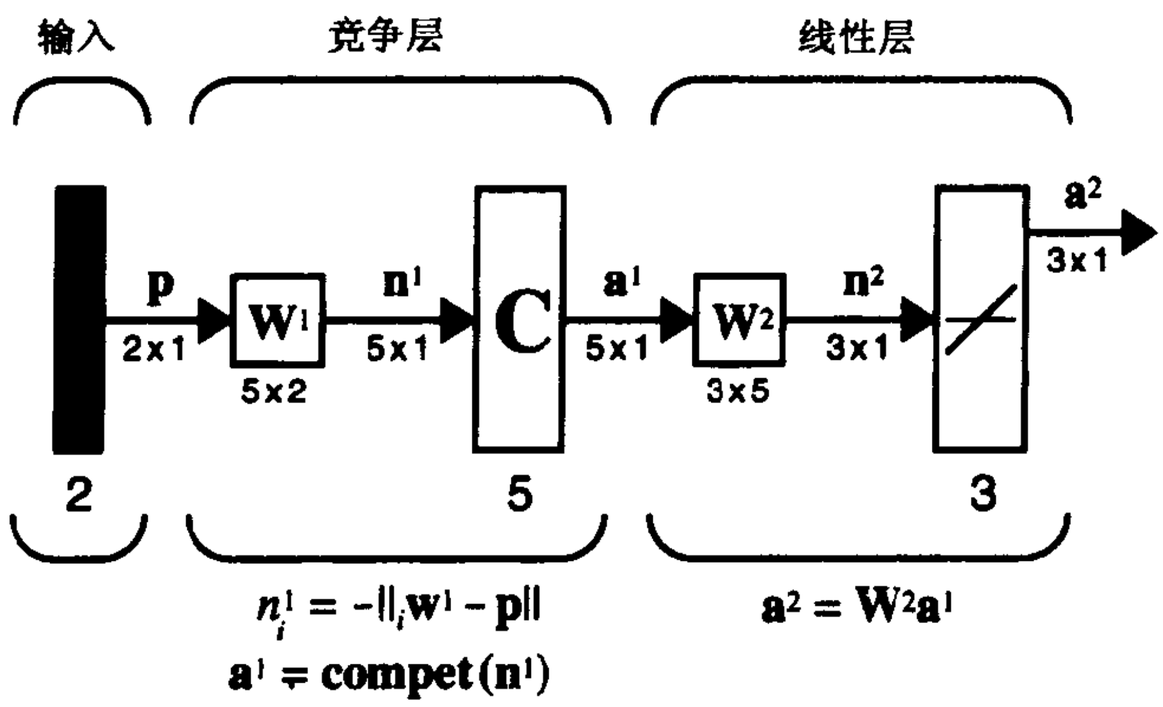



图 14-38 例题 P14.5 的 LVQ 网络

$$\mathbf{W}^1 = \begin{bmatrix} 0 & 0 \\ 1 & -1 \\ 1 & 1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix}, \quad \mathbf{W}^2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

14-31

**解**

我们根据 $\mathbf{W}^2$ 中第 $i$ 列的相应非零元素的下标 $k$ 来标记 $\mathbf{W}^1$ 中的每个向量 ${}_i\mathbf{w}$，由此作出图 14-39。

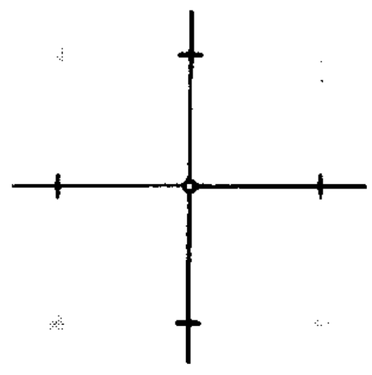
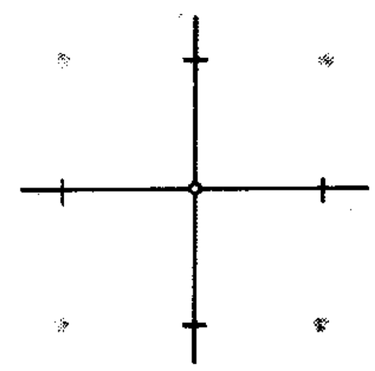


图 14-39 用类标记的原型向量

分隔每个类的判定边界，通过在每对原型向量之间画连接线而得到，这些连接线与一条假想的连接原型向量的线正交，并且与每个向量的距离相等。

在图 14-40 中，每个凸区域按其最接近的权值向量着色。



14-32

图 14-40 类区域以及判定边界

**P14.6** 设计一个 LVQ 网络求解图 14-41 中所示的分类问题。图中的向量将根据其颜色而归入三类之一。

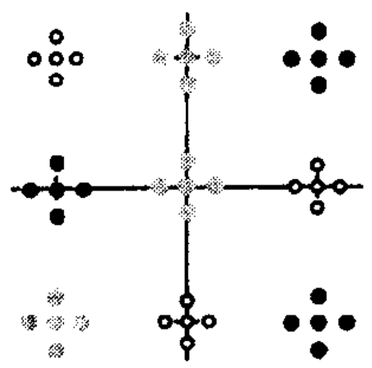


图 14-41 分类问题

当设计完成时，画图表示每个类的区域。

**解**

首先，我们注意到因为 LVQ 网络直接计算向量之间的距离，而不是采用内积，所以它能够区分未经规格化的向量，如上所示。

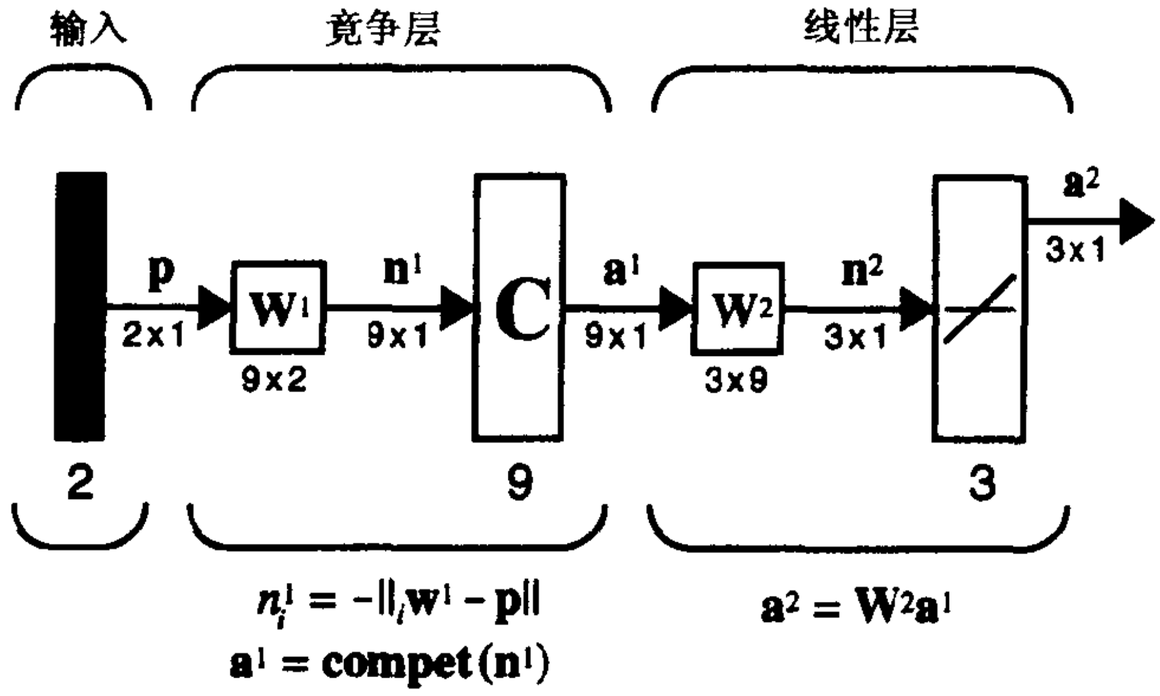
接下来为每种颜色指定一个类：

- 所有白色点属于类 1

• 所有灰色点属于类 2

• 所有黑色点属于类 3

现在选择 LVQ 网络的维数。因为有 3 个类，因而网络的输入层必有 3 个神经元；有 9 个子类(即簇)，因而隐含层将为 9 个神经元。这样就得到图 14-42 所示的网络。




14-33

图 14-42 例题 P14.6 的 LVQ 网络

我们可以通过使每行等于一个簇的转置原型向量来设计权值矩阵 $\mathbf{W}^1$。选取每个簇正中央的原型向量，得到如下结果：

$$\mathbf{W}^1 = \begin{bmatrix} -1 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & -1 & -1 & -1 \end{bmatrix}^T$$

现在第一层的每一个神经元将对不同的簇作出响应。

接下来选择 $\mathbf{W}^2$，使得每个子类都与正确的类相连。为此，使用如下规则：

$$\text{如果子类 } i \text{ 是属于类 } k, \quad \text{则令 } w_{ik}^2 = 1$$

例如，第一个子类是向量图中左上方的那个簇。这个簇的向量是白色的，因而它们属子第一类。所以我们应设置 $w_{1,1}^2$ 为 1。

一旦设置好所有的 9 类，就得到如下结果：

$$\mathbf{W}^2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

可以通过提交向量来测试网络。这里以 $\mathbf{p} = [1 \quad 0]^T$ 来计算第一层的输出：

$$\mathbf{a}^1 = \mathbf{compet}(\mathbf{n}^1) = \mathbf{compet}\left(\begin{bmatrix} -\sqrt{5} \\ -\sqrt{2} \\ -1 \\ -2 \\ -1 \\ 0 \\ -\sqrt{5} \\ -\sqrt{2} \\ -1 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

结果网络指出我们所提供的向量属于第 6 子类。再看第二层网络的结果： 14-34

$$\mathbf{a}^2 = \mathbf{W}^2\mathbf{a}^1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

第二层网络指出向量属于类 1，与事实相符。类区域和判定边界如图 14-43 所示。

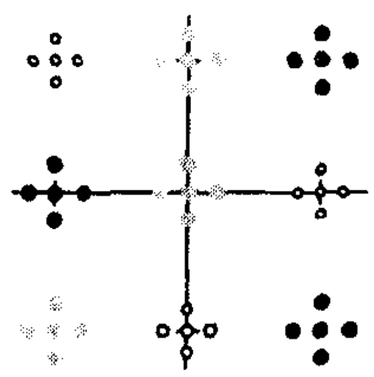


图 14-43 类区域和判定边界

**P14.7** 竞争层和特征图都要求向量是规格化的。如果所用数据是非规格化的，则结果如何？

处理这种数据的一种方法就是在将向量提交给网络之前先进行规格化。但这样做的缺点是向量大小的信息(有时是很重要的)丢失了。

另外一种解决方法是把通常用来计算净输入的内积表达式

$$\mathbf{a} = \mathbf{compet}(\mathbf{Wp})$$

改为直接计算距离

$$n_i = -\left\| {}_i\mathbf{w} - \mathbf{p} \right\| \text{和 } \mathbf{a} = \mathbf{compet}(\mathbf{n})$$

正如 LVQ 网络所做的那样。这种方法有效且保留了向量大小的信息。

然而，还有第三种解决方法，就是在规格化之前给每个输入向量附加一个常量 1，那么在附加元素 1 后改变向量将保留大小信息。 14-35

用第三种方法规格化以下向量：

$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{p}_3 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

**解**

首先我们给每个向量增加一个元素 1：

$$\mathbf{p}_1' = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad \mathbf{p}_2' = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \quad \mathbf{p}_3' = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

然后规格化每个向量：

$$\mathbf{p}_1'' = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} / \left\| \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\| = \begin{bmatrix} 1/\sqrt{3} \\ 1/\sqrt{3} \\ 1/\sqrt{3} \end{bmatrix}$$

$$\mathbf{p}_2'' = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} / \left\| \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \right\| = \begin{bmatrix} 0 \\ 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

$$\mathbf{p}_3'' = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} / \left\| \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\| = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

14-36 现在每个向量的第 3 个元素包含了大小信息，因为它等于原来向量大小的倒数。

## 14.5 结束语

本章我们讲解了第 13 章介绍的联想 instar 学习规则如何与竞争网络相结合，与第 3 章中的 Hamming 网络相同，产生了强有力的自组织网络。由于竞争和 instar 规则的结合，使得任何由神经网络学习的原型向量成为某个特定输入向量类的代表。这样竞争网络通过学习将输入空间分成不同的类。每个类都由一个原型向量(权值矩阵的行)所代表。

本章讨论了三种由 Tuevo Kohonen 提出的神经网络。第一种是标准的竞争层网络，它简单的操作使得它成为解决许多问题的有效网络。

自组织特征图与竞争层网络非常相似，但更接近于生物学中的加强中心/抑制周围的网络，结果使得网络不仅能够学习分类输入向量，还能学习输入空间的拓扑结构。

第三种网络 LVQ 网络，使用了有监督及无监督的学习来识别簇。它通过第二层将多个凸区域组合成可以有任何形状的类。LVQ 网络能够通过训练来识别由多个不联结的区域构成的类。

第 15 章与第 16 章将以本章所讲述的神经网络为基础。例如，第 15 章将更为详细地讨论横向抑制、加强中心/抑制周围网络以及这些网络的生物学基础。第 16 章将讨论标准竞争网络的一种改进(称为自适应谐振理论)，这个理论解决了本章所讨论的权值稳定性问题。 14-37

## 参考文献

[FrSk91] J. Freeman and D. Skapura, *Neural Networks: Algorithms , Applications, and Programming Techniques*, Reading, MA: Addison-Wesley, 1991 .

这本书含有一些网络算法的代码段，揭示了网络的细节部分。

[KoHo87] T. Kohonen, *Self-Organization and Associative Memory*, 2nd Ed., Berlin: Springer-Verlag, 1987.

这本书介绍了 Kohonen 学习规则和几个使用该规则的网络。书中还提供了对线性联想模型的完整分析，并给出了很多扩展和例子。

[Hech90] R. Hecht-Nielsen, *Neurocomputing*, Reading, MA: Addison-Wesley, 1990.

这本书包含对竞争学习的历史和数学方面的介绍。

[RuMc86] D. Rumelhart, J. McClelland et al., *Parallel Dsitributed Processing*, vol. 1, Cam-

bridge, MA: MIT Press, 1986.

这套两卷集著作是神经网络的经典文献。第一卷中有一章描述竞争层以及它们如何进行特征检测的过程。 14-38

## 习题

**E14.1** 假设 Hamming 网络第二层的权值矩阵如下：

$$\mathbf{W}^2 = \begin{bmatrix} 1 & -\frac{3}{4} & -\frac{3}{4} \\ -\frac{3}{4} & 1 & -\frac{3}{4} \\ -\frac{3}{4} & -\frac{3}{4} & 1 \end{bmatrix}$$

这个矩阵不符合式(14.6)的条件，因为

$$\varepsilon = \frac{3}{4} > \frac{1}{S-1} = \frac{1}{2}$$

请给出第一层的一个输出，使得第二层不能够正常操作。

**E14.2** 考虑图 14-44 中所示的输入向量及初始权值

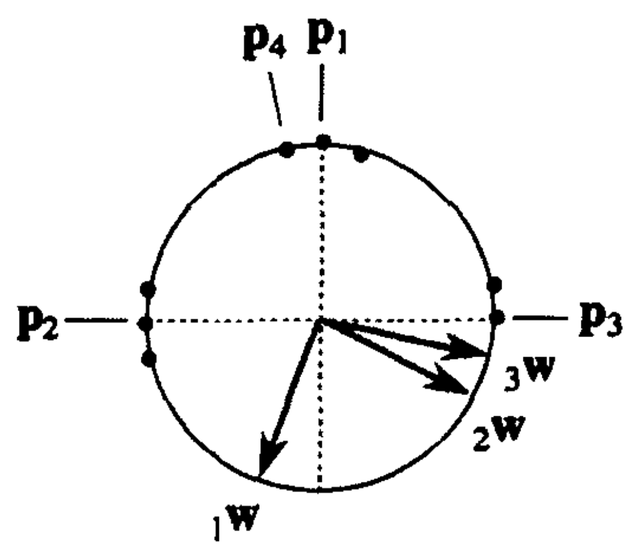



图 14-44 簇数据向量

(i) 画出一个竞争网络图，该网络能够将上图所示数据分类，从而使三簇向量都有自己的类。

(ii) 使用所提供的初始权值以图形方法训练网络，带标号的向量以 $\mathbf{p}_1$，$\mathbf{p}_2$，$\mathbf{p}_3$，$\mathbf{p}_4$ 的顺序提交。回想如果多个神经元有相同的输出，则竞争传输函数选择有最小下标的那个神经元。图 14-3 以图形方法介绍了 Kohonen 规则。 14-39

(iii) 重画图 14-1 中的图形，在其中显示你得到的最后权值向量，以及代表一个类的每一区域之间的判定边界。

**E14.3** 利用下述输入模式训练竞争网络：

$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{p}_3 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

(i) 使用 Kohonen 学习规则，其中 $\alpha = 0.5$，将输入模式训练一遍(即每个输入按给定顺序提交一次)，图示结果。假设初使权值矩阵为

$$\mathbf{W} = \begin{bmatrix} \sqrt{2} & 0 \\ 0 & \sqrt{2} \end{bmatrix}$$

(ii) 训练一遍输入模式之后，模式如何聚集？(即哪些输入模式被归入同一类中？)如果输入模式以不同顺序提交，结果会改变吗？解释其原因。

(iii) 用 $\alpha = 0.25$ 重复(i)。这种改变对训练有何影响？

**E14.4** 在本章前面我们曾用"良心"来指一种技术，用以避免困扰竞争层网络和 LVQ 网络的死神经元问题。

离输入向量太远以致无法竞争获胜的神经元，能够通过调整偏置值使神经元每竞争获胜一次就增加负偏置值，从而得到获胜的机会。其结果是常获胜的神经元开始出现"负疚"感，直到其他神经元得到获胜的机会。

图 14-45 展示了一个具有偏置值的竞争网络。一个典型的对神经元 $i$ 的偏置值 $b_i$ 的学习规则是

14-40

$$b_i^{new} = \begin{cases} 0.9 b_i^{old}, & i \neq i^* \\ b_i^{old} - 0.2, & i = i^* \end{cases}$$

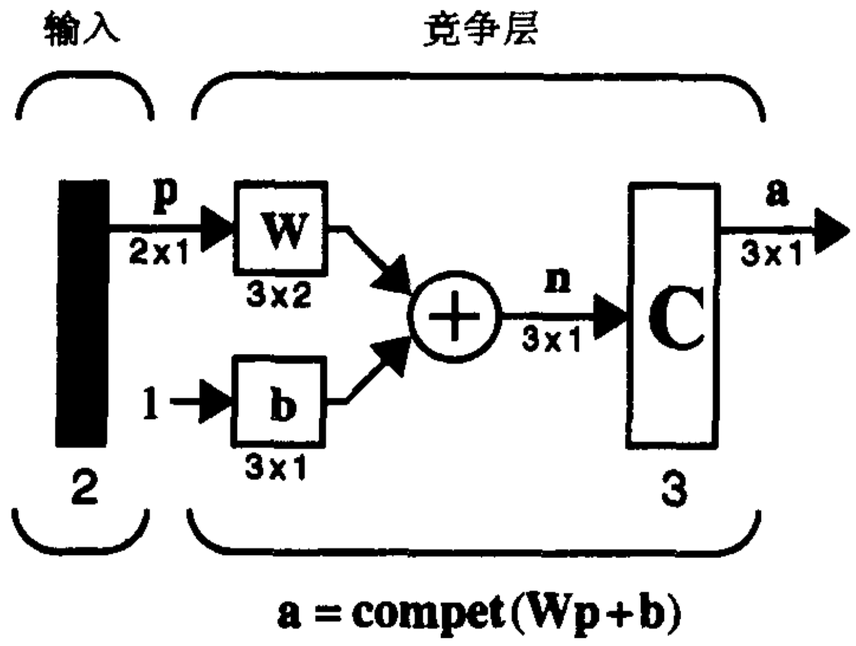



图 14-45 有偏置值的竞争层网络

(i) 检验图 14-46 中的向量，是否存在一种提交向量的次序使得 ${}_1\mathbf{w}$ 能够竞争获胜并且向其中一个向量移近？(注意：假设不使用自适应偏置值。)

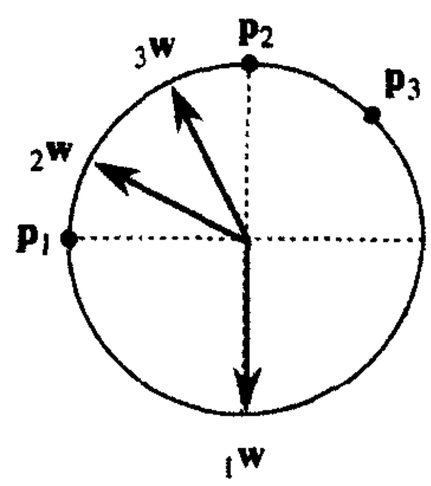



图 14-46 输入向量和死神经元

(ii) 给出如下的输入向量、初始权值以及偏置值，计算权值(用 Kohonen 规则)及偏置值(用上述偏置值规则)。重复如下所示序列，直至神经元 1 竞争获胜：

$$\mathbf{p}_1 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{p}_3 = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

$${}_1\mathbf{w} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \quad {}_2\mathbf{w} = \begin{bmatrix} -2/\sqrt{5} \\ -1/\sqrt{5} \end{bmatrix}, \quad {}_3\mathbf{w} = \begin{bmatrix} -1/\sqrt{5} \\ -2/\sqrt{5} \end{bmatrix}$$

输入向量的顺序：$\mathbf{p}_1$，$\mathbf{p}_2$，$\mathbf{p}_3$，$\mathbf{p}_1$，$\mathbf{p}_2$，$\mathbf{p}_3$，… 14-41

（**iii**）在 ${}_1\mathbf{w}$ 竞争获胜之前共提交多少次？

**E14.5** LVQ 网络的净输入表达式是直接计算输入向量与每个权值向量之间的距离，而不是使用内积。因而 LVQ 网络不需要规格化的输入向量。这种技术也可以用于使竞争层网络分类非规格化的向量。这样的网络请见图 14-47。

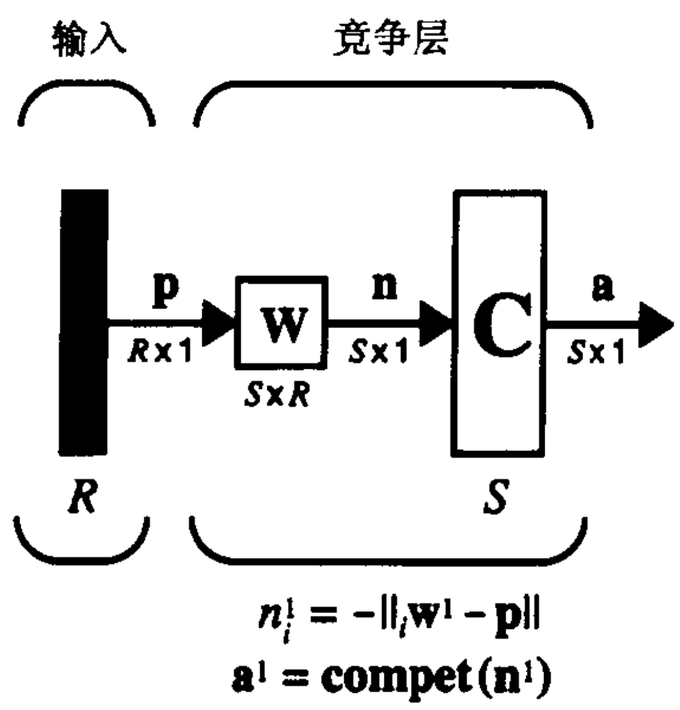



图 14-47 替换为净输入表达式的竞争层网络

使用这种技术对如下非规格化的向量训练一个 2 神经元竞争层网络，其中学习速度 $\alpha = 0.5$。

$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, \quad \mathbf{p}_3 = \begin{bmatrix} -2 \\ -2 \end{bmatrix}$$

以如下顺序提交向量：

$$\mathbf{p}_1, \quad \mathbf{p}_2, \quad \mathbf{p}_3, \quad \mathbf{p}_2, \quad \mathbf{p}_3, \quad \mathbf{p}_1$$

网络的初始权值为

$${}_1\mathbf{w} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad {}_2\mathbf{w} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

**E14.6** 证明图 14-47 所示改进的竞争网络(直接计算距离)与标准的竞争网络(使用内积且输入向量是规格化的)产生同样的结果。 14-42

**E14.7** 我们希望得到一个分类器，能够将如下定义的方形区域分成 16 个面积大致相等的类：

$$0 \leqslant p_1 \leqslant 1, \quad 2 \leqslant p_2 \leqslant 3$$

（**i**）使用 MATLAB 在上述区域内随机产生 200 个向量。

（**ii**）写一个 MATLAB 的 M-文件，用 Kohonen 学习来实现一个竞争层网络。用直接计算输入向量与权值向量之间的距离来计算净输入，正如 LVQ 网络所做的那样，因此向量不必规格化。用 M-文件训练竞争层网络以分类 200 个向量。试用不同的学习速度并比较性能。

(iii) 写一个 MATLAB 的 M-文件以实现 4 神经元 × 4 神经元(二维)的特征图。使用特征图来分类相同的向量。使用不同的学习速度和邻域大小，并比较性能。

**E14.8** 我们想要一个可以将下述定义的输入空间的区间分成 5 个类的分类器：

$$0 \leqslant p_1 \leqslant 1$$

(i) 用 MATLAB 随机产生在上述区间均匀分布的 100 个随机值。

(ii) 平方每个值使分布变成不均匀的。

(iii) 写一个 MATLAB 的 M-文件实现一个竞争层网络，对于平方后的值，用 M-文件训练一个 5 个神经元的竞争层网络，直到权值完全稳定。

(iv) 竞争层的权值是如何分布的？是否与权值如何分布和输入值的平方如何分布有关？

**E14.9** LVQ 网络有如下权值：

$$\mathbf{W}^1 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix}, \quad \mathbf{W}^2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

14-43

(i) LVQ 网络有多少个类和多少个子类？

(ii) 画图展示第一层权值向量以及将输入空间分成子类的判定边界。

(iii) 在每个子类区域上标明它所属的类。

**E14.10** 我们希望得到能够将下述向量按所示的类分类的 LVQ 网络：

$$\text{类 1:}\left\{\begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}\right\} \quad \text{类 2:}\left\{\begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}\right\} \quad \text{类 3:}\left\{\begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}\right\}$$

(i) LVQ 网络的每一层各需要多少个神经元？

(ii) 确定第一层的权值。

(iii) 确定第二层的权值。

(iv) 至少用每个类的一个向量测试你的网络。

**E14.11** 我们希望得到能够将下述向量按所示的类分类的 LVQ 网络：

$$\text{类 1:}\left\{\mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{p}_2 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}\right\} \quad \text{类 2:}\left\{\mathbf{p}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{p}_4 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right\}$$

(i) 这种分类问题是否能够通过感知机解决？解释你的答案。

(ii) 在这种能够分类上述数据的 LVQ 网络之中，每一层需要有多少神经元？假设每个类都由 2 个凸形子类组成。

(iii) 确定这个网络的第二层的权值。

(iv) 将网络第一层的权值全部初始化为零，并对下列向量计算用 Kohonen 规则学习(学习速度 $\alpha = 0.5$)时权值的变化：

$$\mathbf{p}_4, \quad \mathbf{p}_2, \quad \mathbf{p}_3, \quad \mathbf{p}_1, \quad \mathbf{p}_2$$

14-44

(v) 画图表示输入向量、最终权值向量和两个类之间的判定边界。

# 第15章 Grossberg网络

## 15.1 目的

本章我们将继续讨论第13章和第14章中的联想学习算法和竞争学习算法。本章介绍的Grossberg网络是一种自组织连续的竞争网络。这将是我们第一次讨论连续递归网络，并且将引入一些概念，这些概念在第17章和第18章要作进一步的讨论。Grossberg网络也是在第16章将要讨论的自适应谐振理论(ART)网络的基础。

我们将从讨论Grossberg网络的生物学启发(即人的视觉系统)开始。尽管我们不能全面深入地讨论这个问题，但Grossberg网络受生物学影响如此之深，如果不把它放在生物学的背景下将很难进行讨论。注意生物学为人工神经网络提供了最初的启示是很重要的，并且应该继续从中寻求启示，因为科学家对脑的功能不断有新的发现。 15-1

## 15.2 理论和实例

在20世纪60年代晚期和70年代，研究神经网络的人数急剧地减少。但是仍有一批研究人员继续在这个领域工作，其中特别包括Tuevo Kohonen，James Anderson，Kunihiko Fukushima和Shun-ichi Amari。最富于创造性的一人就是Stephen Grossberg。

Grossberg从60年代早期起就一直活跃在神经网络研究领域，并取得丰硕成果。他的工作的特点是使用非线性数学来模拟思维和脑的特定功能，并且他所取得的大量成果与对脑任务的了解程度是一致的。他的论文题目，涉及从神经网络如何在视觉中提供对比增强之类的特殊领域，到人类记忆的普遍理论这样一般性的主题。

部分由于他的成就的高度使他的工作享有“难”的名声。每一篇新的论文都是建立在过去30年研究的基础之上，因而很难去衡量其价值。此外，他用的术语是自成体系的，与其他研究人员使用的不同。他的工作也以高难度的数学以及神经生理学的复杂性为特点。他受Helmholtz，Maxwell和Mach等人对脑的机能的交叉学科研究的启发，并将他们的观点引入到自己的工作中。他的研究处于数学、生理学和神经生理学的交汇处。缺乏这些领域的背景知识对初读其作品会带来困难。

本章我们将对Grossberg一种独创性的网络作初步的了解。为了尽可能地理解他的观点，将首先简要介绍他的网络的生物学启发：视觉系统。然后给出用于许多Grossberg网络的数学构造模块：并联模型。在对这个简单模型的功能有个了解之后，我们将演示如何为自适应模式识别建立神经网络。这个网络将是第16章讨论的自适应谐振理论网络的基础。通过逐步地建立越来越复杂的网络，我们希望能够使它们更容易被理解。

我们应从本章的讨论中吸取一个非常重要的教训。尽管人工神经网络的最初启发来源于生物学，但时常我们忘记回到生物学去寻找新的思想。将会出现生物学、数学、心理学和其他学科的结合，这将极大地促进我们对神经网络的理解。 15-2

### 15.2.1 生物学的启发：视觉

本章所描述的神经网络受到人类视觉系统的生理学研究的启发。这一小节我们要对视觉作概略的介绍，从而使网络的功能更易于理解。

图 15-1 是视觉系统的第一阶段的一个示意图。光通过角膜(眼前部的透明体)和水晶体，水晶体使光线折射从而将物体聚焦在视网膜上(眼外壁的内层)。正是在光线落到视网膜后，将这种大量信息翻译为可理解的影像的过程才开始。正如将在本章后面看到的那样，我们所“看”到的许多东西并不是实际投射在视网膜上的影像。

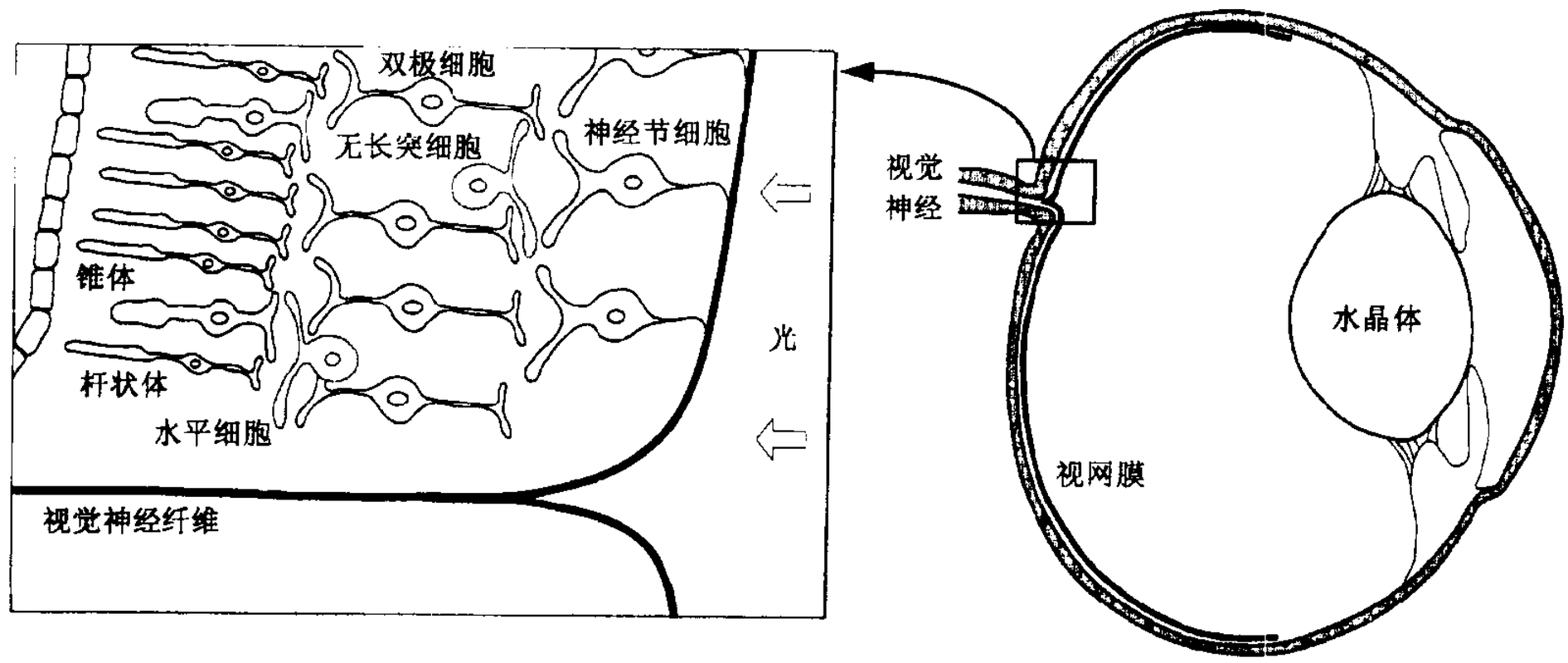



图 15-1 眼球和视网膜

**视网膜 杆状体 锥体** 视网膜实际是大脑的一部分，它在胎儿发育过程中与脑分离，但保留了视神经与脑的相连。视网膜有三层神经细胞。外层由光感受器(杆状体和锥体)组成，用来将光转化成电信号。杆状体细胞使得我们能在昏暗的情况下看见东西，而锥体使我们看到精细的细节以及颜色。由于现在还不知道的原因，光必须通过视网膜的另外两层来刺激杆状体与锥体。正如在下面将会看到的，这种障碍必须在神经的处理过程中得到补偿，以便重建可识别的图像。

**双极细胞 水平细胞 无长突细胞** 视网膜的中间层由三种细胞组成：双极细胞、水平细胞和无长突细胞。双极细胞从接受器接受输入并且传递给视网膜的第三层。水平细胞联结接受器和双极细胞，而无长突细胞联结双极细胞与神经节细胞。

15-3

**神经节细胞** 视网膜的最后一层由神经节细胞组成。神经节细胞的轴突通过视网膜的表面而集成一束形成视觉神经。很有趣的是每只眼都有大约 1.25 亿个感受器，但只有 100 万个的神经节细胞。显然在视网膜那里做了大量的处理以减少数据。

**视觉皮层** 神经节细胞的轴突部分，成为一束视觉神经，与大脑的一个叫做“横向膝状核”的区域相连，如图 15-2 所示。从这里视觉神经纤维扇出到大脑后部的主视觉皮层。神经节细胞的轴突和横向膝状核细胞构成突触，而横向膝状核细胞和视觉皮层中的细胞构成突触。视觉皮层是指大脑的一个实现视觉功能并且由许多层细胞组成的区域。

沿着视觉路径的联结绝非随意的。每一层到下一层的映射都有高度的组织。视网膜特定部分的神经节细胞的轴突伸到横向膝状核的特定部分，然后再伸到视觉皮层的特定部分(这

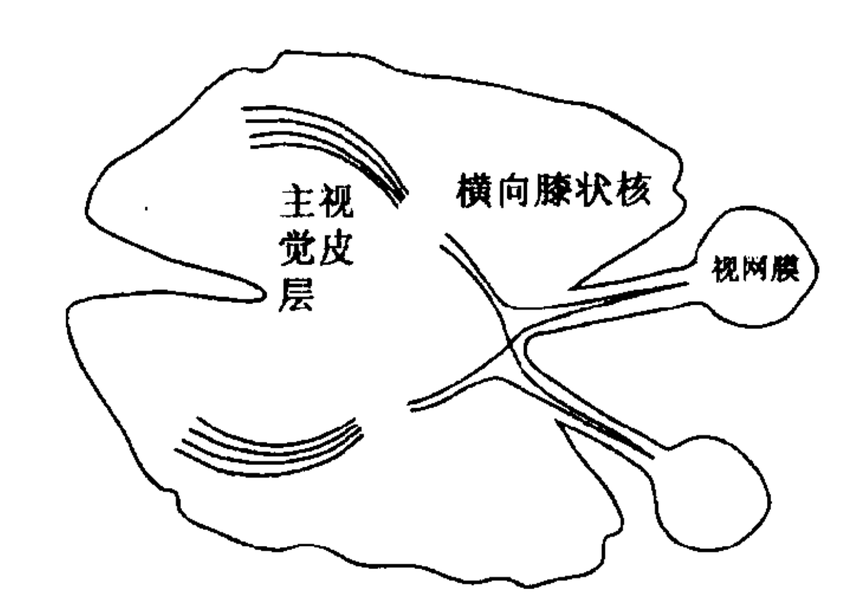



图 15-2 视觉路径

种拓扑映射是受第 14 章所描述的自组织特征图的启发)。此外，正如我们在图 15-2 中所看到的那样，脑的每个半球都接收来自两只眼的输入，因为视觉神经纤维的一半交叉而另一半保持不交叉。结果是每个视觉区的左半部分在脑的左半部分结束，而每个视觉区的右半部分在脑的左半部分结束。

1. 幻觉

我们对视觉路径的大致结构有了某些了解，但是它是如何起作用的呢？视网膜的三层各有什么作用？横向膝状核神经完成什么任务？这些问题将能够从对视觉幻觉的研究中受到一些启示。 15-4

为什么有这么多视觉幻觉？克服视网膜的不完善的吸收过程的机制产生了幻觉。Grossberg 和其他人使用了大量已知的幻觉来探测自适应感知机制[GrMi89]。如果我们能做出与生物学系统产生同样幻觉的数学模型，那么我们就会有一个机制，可以描述脑的这一部分是如何工作的。为了帮助理解幻觉存在原因，我们将首先考虑一下视网膜吸收过程的一些不完善的地方。

**视神经乳头** 图 15-3 是一幅眼科医生透过角膜所看到的视网膜的图像。图中大的浅色圆圈是*视神经乳头*，那里视神经离开视网膜而进入横向膝状核神经。这里也是动脉进入视网膜以及静脉移开的地方。这个视神经乳头导致了视觉上盲点的存在。稍后将会讨论这一点。

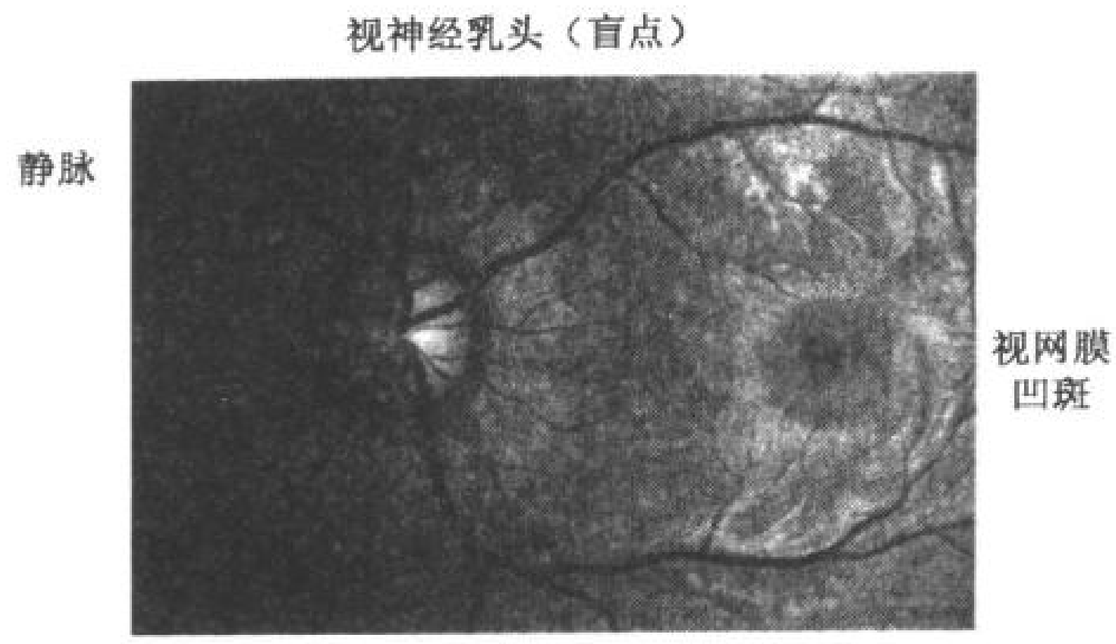



图 15-3 眼底

**视网膜凹斑** 视神经乳头右边的深色圆圈是*视网膜凹斑*，这是我们视场的中心。这是视网膜的一部分，直径大约有半毫米，其中只有锥体细胞。尽管锥体细胞在视网膜的各处都有分布，但它们大部分都集中于视网膜凹斑处。此外，在视网膜的这个区域，其他层被放置边

上，因而锥体细胞排列在前面。密集的光感受器以及没有阻碍，使得我们在视网膜凹斑处有最为精细的视觉，从而使我们能够精确地聚焦晶体。

从图 15-3 能够看到在视网膜的吸收过程中存在着一些不完善之处。首先，在视神经乳头处既无杆状细胞也无锥体细胞，从而给我们的视场造成了一个盲点。因为视觉路径所做的处理使得我们通常不能觉察到盲点的存在，但可以通过一个简单的试验来验证。请看图 15-
15-5 4 中左边的黑色圆圈，并且遮住你的左眼。你将头向纸面移近，然后再远离，那么你将注意到一个点(大约距纸面 9 英寸远)，在那里右边的圆圈将从你的视场中消失(你仍然注视着左边的圆圈)。如果以前你没有试过这么做，那么可能有点难做。有趣的是我们看到自己的盲点并不是以一个黑洞出现。有时我们的大脑设法填补了那个看不到的区域。

图 15-4 盲点测试

另外一个视网膜吸收过程不完善的地方是动脉和静脉在视网膜后部的光感受器的前面交叉。这阻碍了杆状细胞及锥体细胞接收视场中所有的光线。而且，因为光感受器在视网膜的后部，光线必须通过其他两层才能到达那里。

图 15-5 展示了这种不完善之处的结果。从图中我们看展现在视网膜上的一条边。右边的说明最初由光感受器接收到的图像。被盲点和静脉覆盖的区域没有被杆状细胞及锥体细胞观察到。(我们看不到动脉和静脉等的原因是视觉路径并不对稳定的图像产生响应。眼球的不断的震动，也称作跳跃运动，因而即使是视场中固定的物体也在相对于眼球运动。静脉对于眼球是静止的，因而在视场中是暗淡的。)

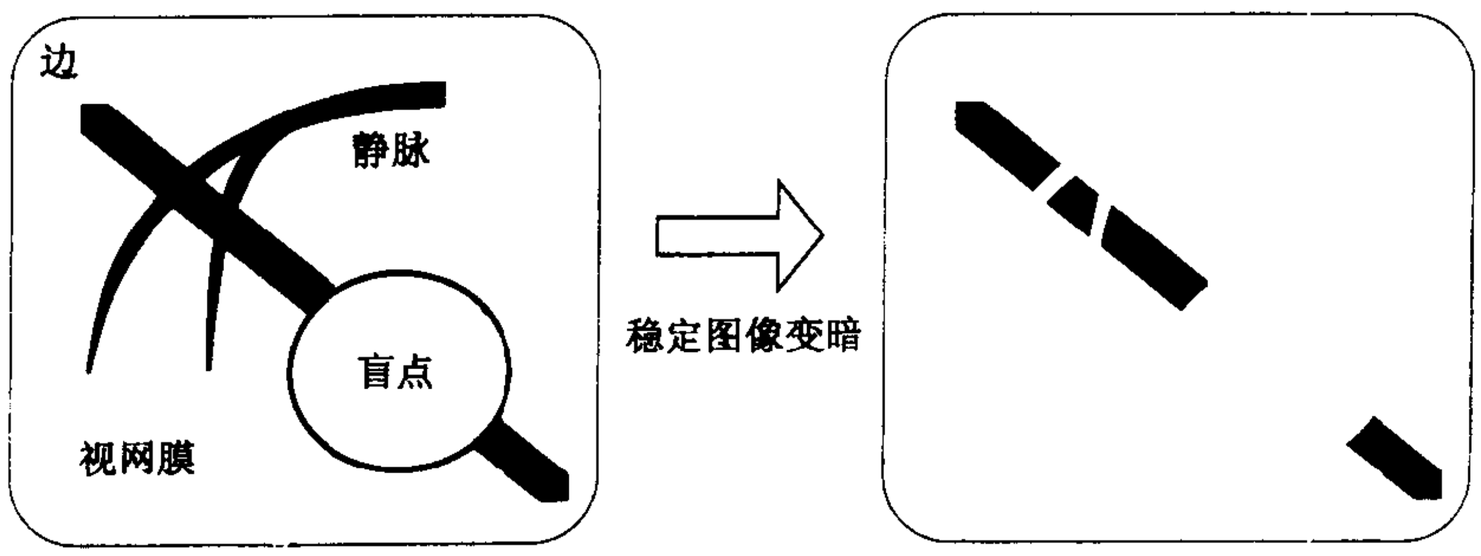



图 15-5 视网膜上一条边的感知(见[Gros90])

**应急切断 特征填充** 因为我们看到的并不是图 15-5 右边所示的边，视觉路径上的神经系统一定做了某些处理，从而弥补了那些失真，补全了图像。Grossberg 提到两种主要的补偿处理。第一种，他称之为*应急切断*(emergent segmentation)，补偿了丢失的边界。第二种，他叫做*特征填充*(featural filling - in)，在产生的边界内填充颜色和亮度。这两种过程在
15-6 图 15-6 中说明。在上面的图中我们看到一条由杆状核细胞和锥体细胞察觉到的原始边，包含丢失的片段。在下面的图中看到在应急切断和特征填充之后的完整边。

如果沿着视觉路径的处理重新建立了我们所看到图像的丢失部分，那么就一定有弄错的时候，因为它不能够确切地知道那些它没有接受到光线的场景部分。这种错误能够由视觉幻

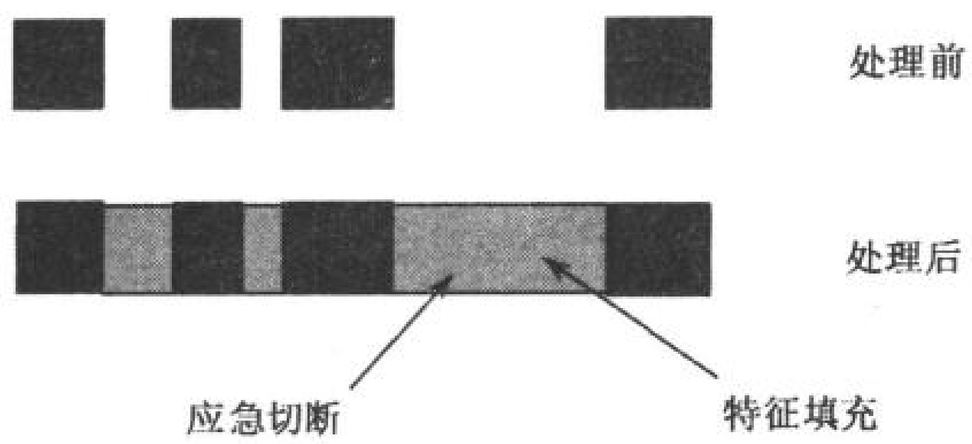



图 15-6 补偿处理(见[Cros90])

觉予以说明。例如，在图 15-7 的左图中你能够看到一个亮白色的三角形置于几个其他黑色物体的上面。事实上，图中并不存在这样的三角，这纯粹是视觉系统的应急切断及特征填充处理过程的一种创造。这种情况同样也适用于右图中那个看起来像置于那些线上的亮白色圆圈。

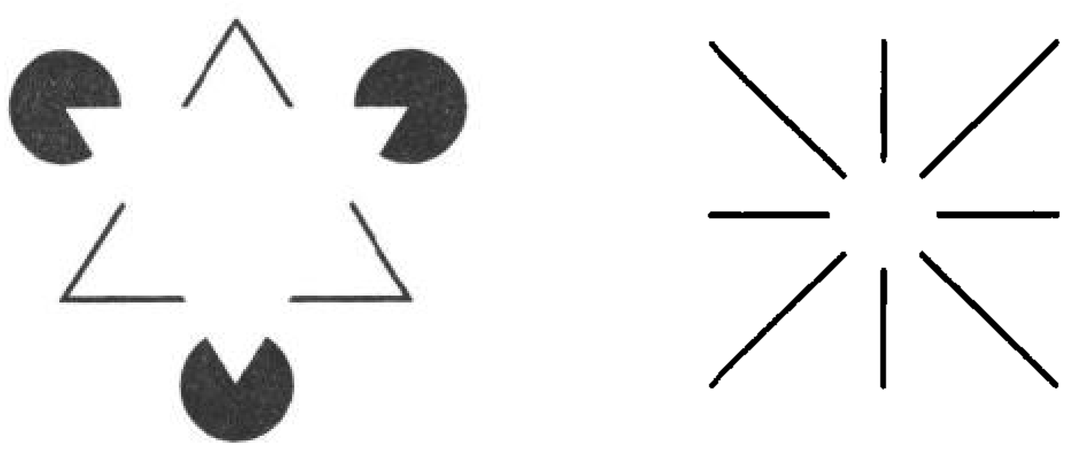


图 15-7

图 15-8 展示了特征填充的过程。这种幻觉叫做霓虹色扩展。在右边的图中你也许能够在图中看到淡蓝色的钻石，甚至宽的淡蓝色十字形线条。在左边的图中能看到淡蓝色的环。填充在钻石中的蓝色及环并不是在印刷过程中涂抹的颜色，也不是由于光的散射。这种效果并没有在视网膜上出现。除了在你的脑中它根本就不存在。(霓虹色扩展的感知现象因人而异，并且感知的强度取决于使用的颜色。如果你在图 15-8 中看不到这种效果，请看任何一期《神经网络》(*Neural Networks*, Pergamon Press)杂志的封面。)

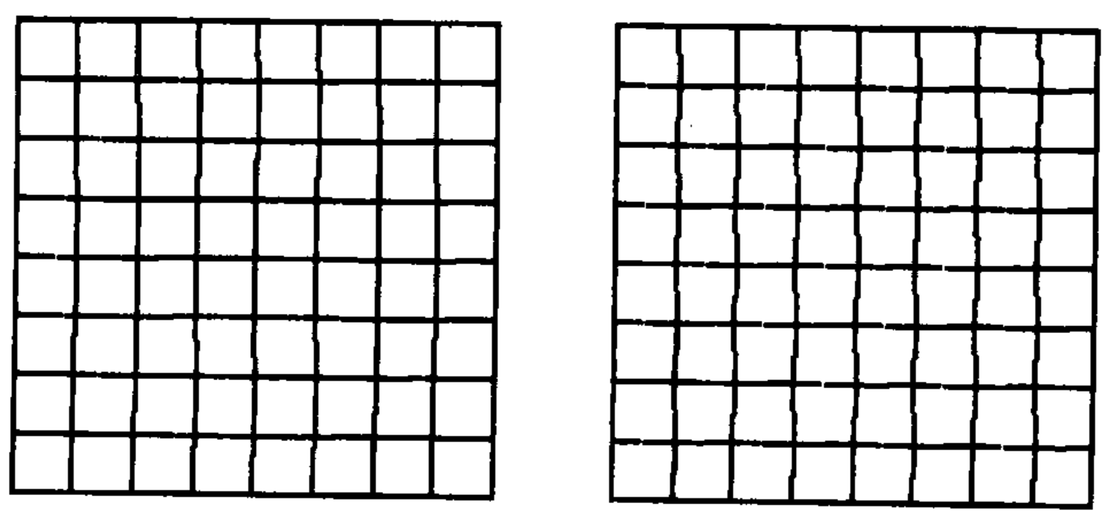
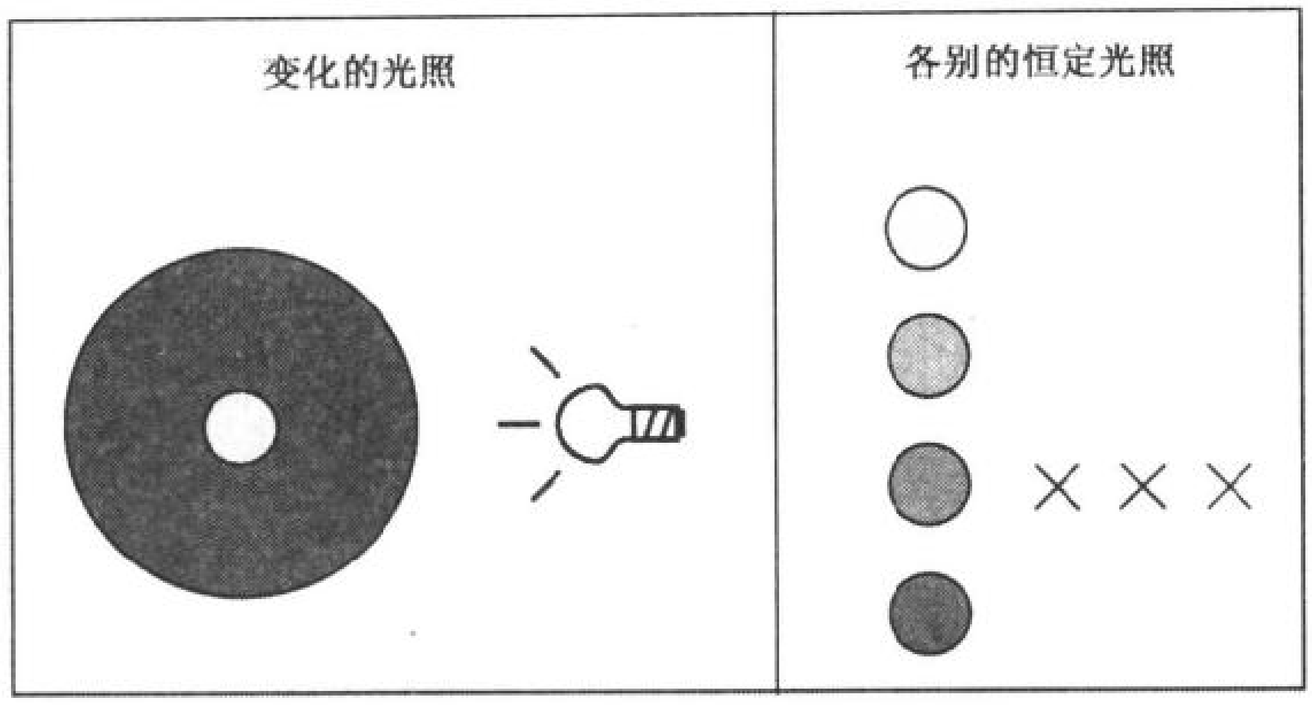


图 15-8 霓虹色扩展(特征填充)

在本章后面我们将讨论一些能够帮助解释应急切断以及其他视觉现象处理过程的神经网络模型。

2. 视觉规格化

**亮度一致　亮度对比**　除了应急切断和特征填充，还有其他两种早期的视觉系统中的现象，对于完成什么动作给予我们一种提示：亮度一致和亮度对比。亮度一致的效果能够通过图 15-9 所示的测试证实。在这个测试中被试者看到一个小灰圆盘在一个大暗灰的圆环中，用一定强度的白光照射。要求被测试者从一系列分别照射的灰色圆盘中指出中间圆盘的亮度，并且选出具有相同亮度的圆盘。然后，对照射灰色圆盘及深色圆环的灯光增加亮度，再次要求被测试者选择出具有相同亮度的圆盘。这样的过程在不同水平的照明度下进行几次。结果，每次试验中被测试者都会选择与原来中间那个圆盘一样的圆盘。即使进入被测试者眼中的总光强度是 10 到 100 倍，只有相对亮度起到了作用。




15-8

图 15-9　亮度一致性的测试(见[Gros90])

视觉系统与亮度一致关系密切的另一个现象，是亮度对比。这种效果可以从图 15-10 中得到证明。在两幅图中心各有一个相同灰度的小圆盘。左面图中的小圆盘被一个深色圆环包围着，右面图中的小圆盘被浅色圆环包围着。尽管两个圆盘有同样的灰度，那个在深色圆环中的显得更亮些。这是因为我们的视觉系统对相对亮度是敏感的。看起来好像跨越整个图像的整体亮度是连续的。

图 15-10

亮度一致和亮度对比的特性对我们的视觉系统很重要。既然我们能够看到许多种不同照明的情况，如果我们不能够补偿一个场景的绝对强度，那将永远不能学会识别物体。Grossberg 称这种规格化过程为“不完全相信光源”(discounting the illuminant)。

在本章的余下部分我们将提出一种与这小节讨论的物理现象相一致的网络结构：基本非线性模型。

### 15.2.2 基本非线性模型

**漏积分器 时间常数** 在介绍 Grossberg 网络之前，我们将先看一些构成网络的组成模块。第一种组成模块是“漏”积分器(“leaky”integrator)，如图 15-11 所示。这个系统的基本方程是

$$\varepsilon \frac{\mathrm{d}n(t)}{\mathrm{d}t} = -n(t) + p(t) \tag{15.1}$$

其中的 $\varepsilon$ 是系统的时间常量。

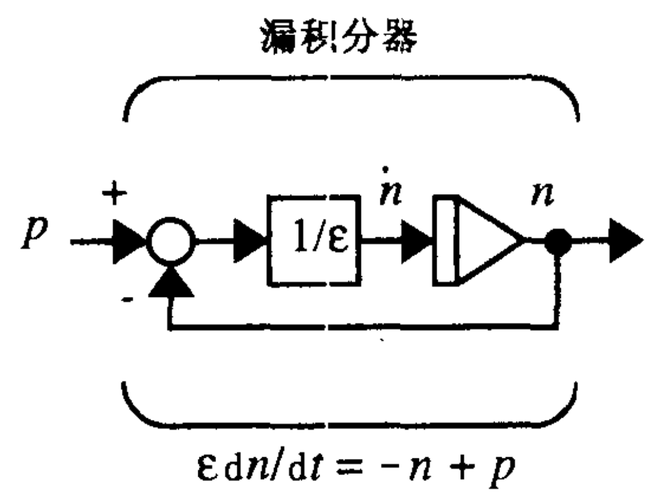



图 15-11 漏积分器

对一个任意的输入 $p(t)$，漏积分器的响应是

$$n(t) = e^{-t/\varepsilon}n(0) + \frac{1}{\varepsilon}\int_0^t e^{-(t-\tau)/\varepsilon}p(t-\tau)\mathrm{d}\tau \tag{15.2}$$ 15-9

例如，如果输入 $p(t)$是常量且初始条件 $n(0)$是 0，等式(15.2)将成为

$$n(t) = p(1 - e^{-t/\varepsilon}) \tag{15.3}$$

图 15-12 给出了这个响应的一个图示，其中 $p = 1$，$\varepsilon = 1$。响应曲线以指数形式趋于稳定的状态值 1。

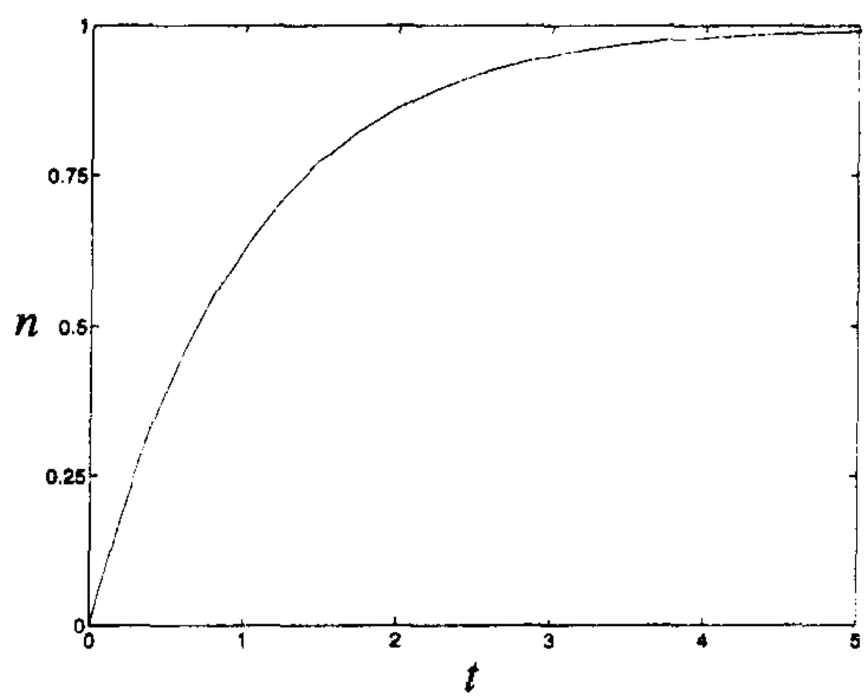



图 15-12 漏积分器的响应曲线

我们需要注意漏积分器的两种重要属性。其一，因为方程(15.1)是线性的，如果输入 $p$ 按比例变化，则响应 $n(t)$将会以同样的大小按比例变化。例如，如果输入加倍，响应也会加倍，但形状不变。这在式(15.3)中是明显的。其二，漏积分器的响应速度由时间常数 $\varepsilon$ 决定。当 $\varepsilon$ 减少时响应速度变快，当 $\varepsilon$ 增加时响应速度变慢(见例题 P15.1)。

试验漏积分器请用 *Neural Network Design Demonstration Leaky Integrator* (**nnd15li**)。

**并联模型 激励 抑制** 漏积分器组成了 Grossberg 的基本神经模型的核心：并联模型(见图 15-13)。这种网络的操作方程是

$$\varepsilon \frac{\mathrm{d}n(t)}{\mathrm{d}t} = -n(t) + (b^+ - n(t))p^+ - (n(t) + b^-)p^- \tag{15.4}$$

其中 $p^+$ 是一个非负数值，代表对网络的激励输入(使响应增加的输入)，$p^-$ 是一个非负数值，代表抑制输入(使响应减少的输入)。偏置值 $b^+$ 和 $b^-$ 是决定神经元响应的上限和下限的非负常量，下面将有详细解释。

15-10

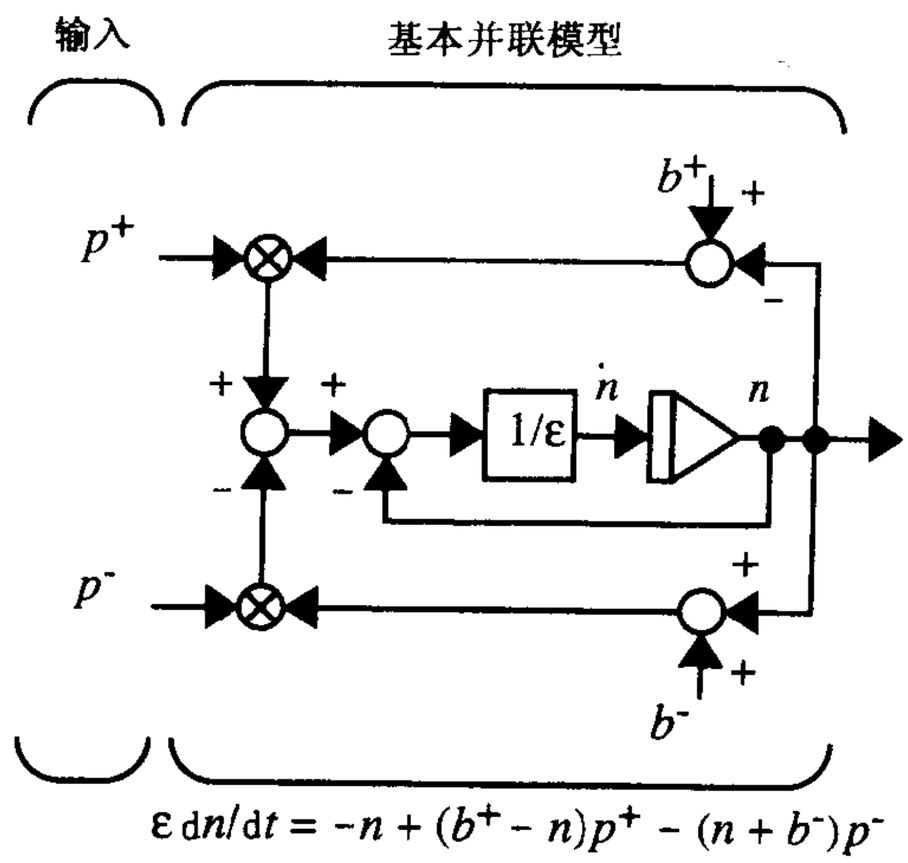



图 15-13 并联模型

在方程(15.4)的右边有三个部分，当这三个部分的最后符号是正时，$n(t)$将会增加。当最后符号为负的时候，$n(t)$将会下降。为了理解网络的性能让我们来研究这三个部分。

第一部分 $-n(t)$是一个线性衰减项，它在漏积分器中也可以见到。当 $n(t)$为正时这一项为负，$n(t)$为负时这一项为正。第二部分$(b^+ - n(t))p^+$，提供非线性的增益控制。当 $n(t)$比 $b^+$小时，这部分为负，但是当 $n(t) = b^+$时变为零。这样实际就给$n(t)$设置了上限 $b^+$。第三部分 $-(n(t) + b^-)p^-$也提供非线性的增益控制。它给$n(t)$设置一个下限 $-b^-$。

图 15-14 展示了并联模型当 $b^+ = 1$，$b^- = 0$ 和 $\varepsilon = 1$ 时的性能。在左图中可以看出当激励输入 $p^+ = 1$ 和抑制输入 $p^- = 0$ 时网络的响应。在右图 $p^+ = 5$，$p^- = 0$。注意到即使激励输入增加了 5 倍，稳定状态的网络响应只增加了 2 倍。如果继续增加激励输入，我们能够发现稳定状态的网络响应将会增加，但总是小于 $b^+ = 1$。

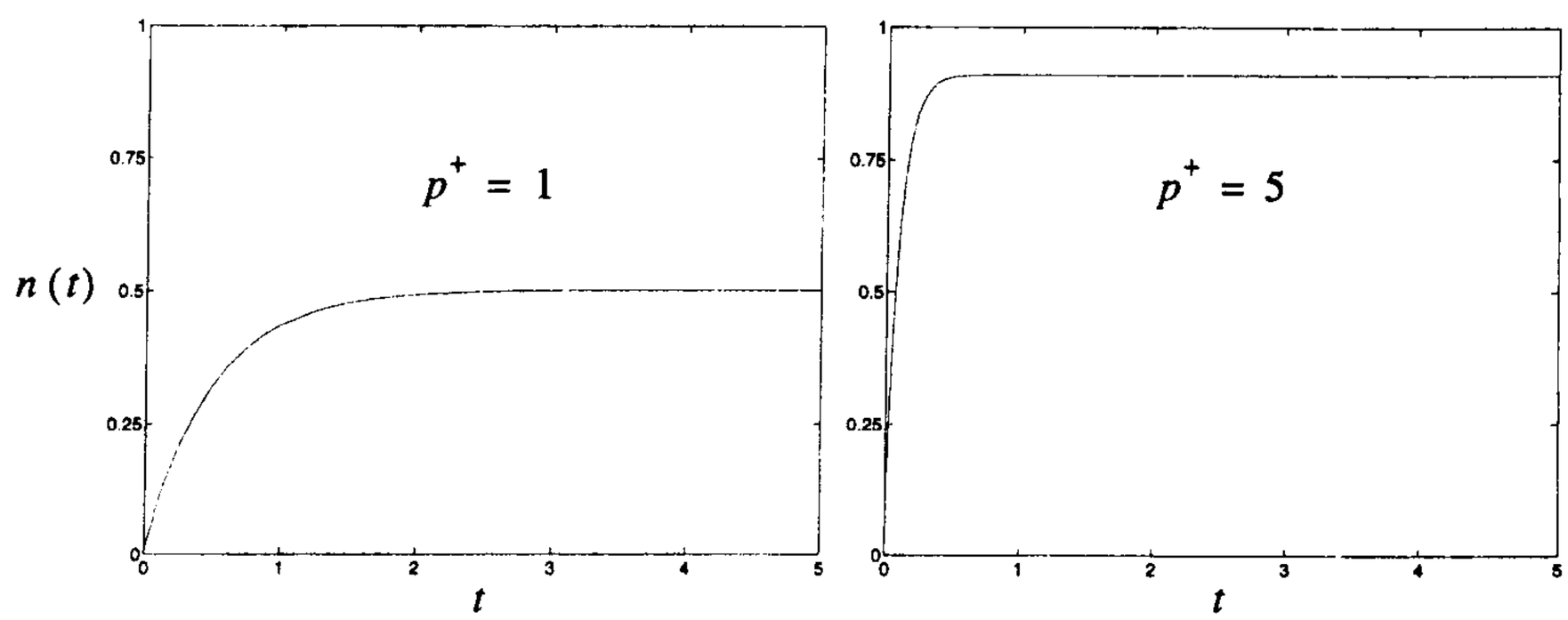



图 15-14 并联网络的响应

如果给并联网络提供一个抑制输入，则稳定状态的网络响应将会降低，但依然将比 $-b^-$ 大。总结一下并联模型的操作，如果 $n(0)$ 在 $b^+$ 和 $-b^-$ 之间，那么 $n(t)$ 将保持在这个限制中，如图 15-15 所示。

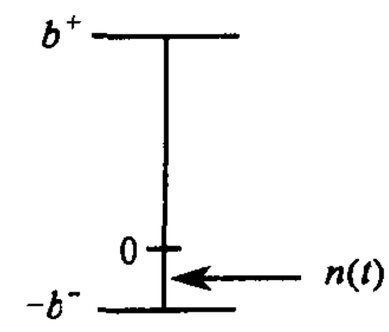


图 15-15

15-11

并联模型是 Grossberg 竞争网络的基础。我们将在下一节讨论这种网络，用非线性的增益控制来规格化输入模式并且在大范围的总体强度中保持相对强度。

试验并联模型请用 *Neural Network Design Demonstration* **Shunting Network** (**nnd15sn**)。

### 15.2.3 两层竞争网络

我们现在已经作好提出 Grossberg 竞争网络准备。这种网络是受哺乳动物的视觉系统启发的，这在本章 15.1 节已作过讨论。Grossberg 受 Chistoph von der Malsburg 的工作[vond73]的影响，而后者又受诺贝尔奖得主 David Hubel 和 Torsten Wiesel 的实验工作[HuWi62]的影响。图 15-16 展示了这样一个网络简图。

**短期记忆 长期记忆** Grossberg 网络由三部分组成：第一层，第二层和自适应权值。第一层是视网膜操作的一个粗略模型，而第二层则代表视觉皮层。这个模型并不完全说明人类视觉系统的复杂性，但它能够说明视觉系统的一些特点。网络包括短期记忆(STM)和长期记忆(LTM)的机制，并且能够进行自适应调整、过滤、标准化和对比度增强。在下面几小节中我们将讨论网络的每一组成部分的操作。

15-12

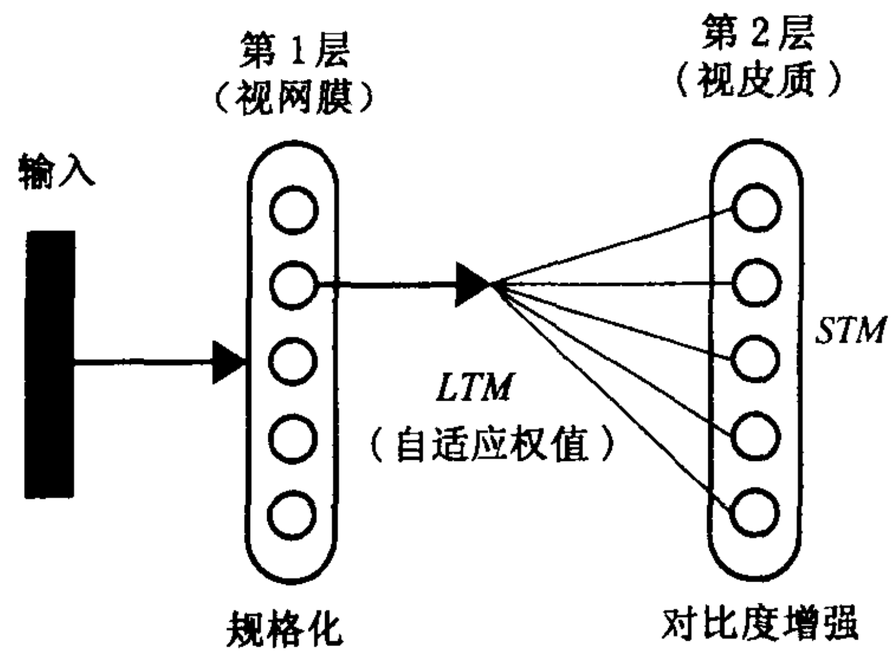


图 15-16 Grossberg 竞争网络

1. *第一层*

Grossberg 网络的第一层接收外部输入并且规格化输入模式的强度。(回忆第 14 章中 Kohonen 网络当输入模式被规格化的时候表现最好。对于 Grossberg 网络这种规格化被网络的第一层实现。)图 15-17 给出了一个这个层次的简图。注意它使用了并联模型，以输入向量 **p** 计算出来激励输入和抑制输入。

15-13

第一层的运算方程是

$$\varepsilon \frac{d\mathbf{n}^1(t)}{dt} = -\mathbf{n}^1(t) + ({}^+\mathbf{b}^1 - \mathbf{n}^1(t))[{}^+\mathbf{W}^1]\mathbf{p} - (\mathbf{n}^1(t) + {}^-\mathbf{b}^1)[{}^-\mathbf{W}^1]\mathbf{p} \qquad (15.5)$$

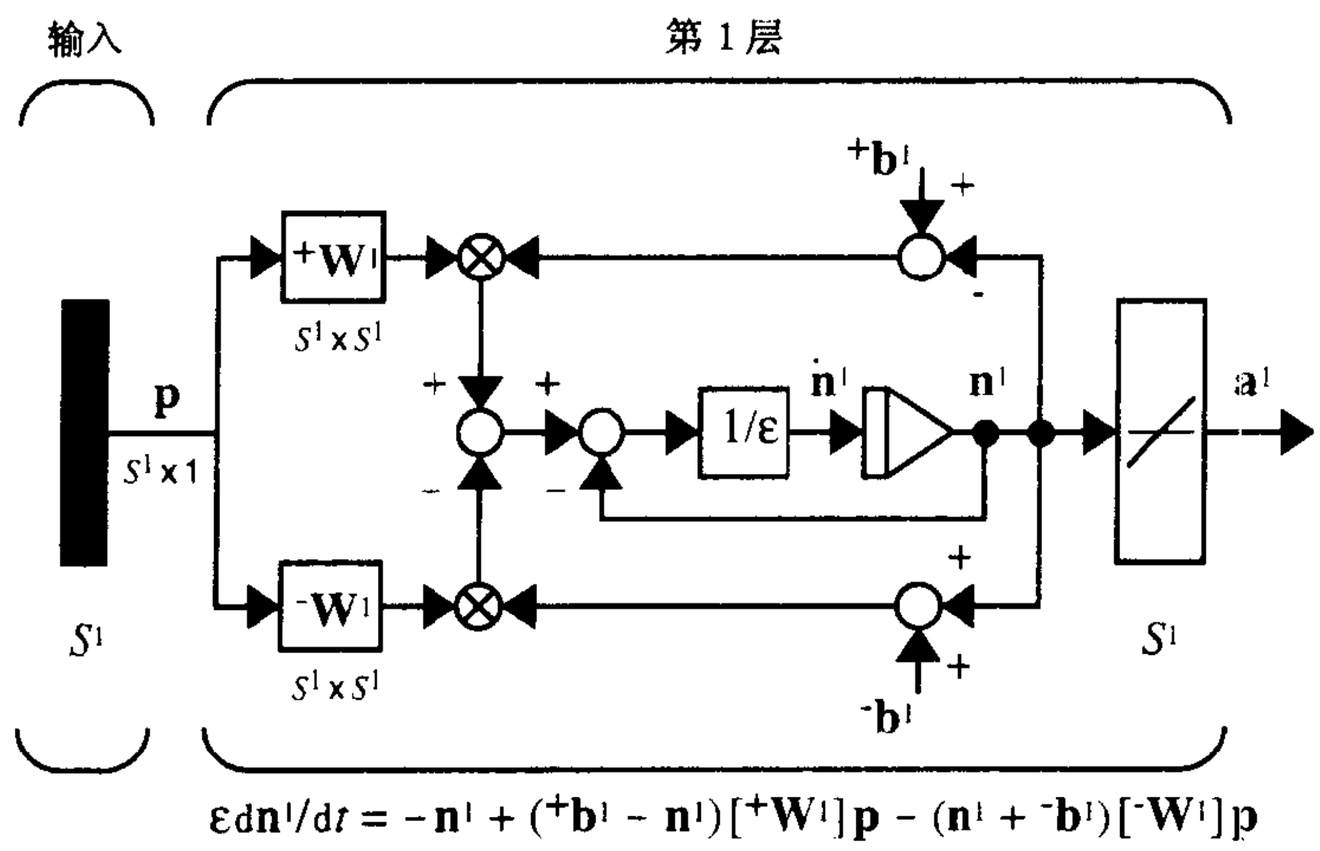



图 15-17 Grossberg 网络的第一层

正如我们早先提到的一样，参数 ε 决定了响应的速度。选择 ε 使得神经元响应的比自适应权值的变化要快得多，我们将在后面小节讨论这个问题。

式(15.5)是一个有激励输入$[{}^{+}\mathbf{W}^{1}]\mathbf{p}$的并联模型，其中

$$
{}^{+}\mathbf{W}^{1} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{15.6}
$$

因此对神经元 $i$ 的激励输入是输入向量的第 $i$ 个元素。

第一层的抑制输入是$[{}^{-}\mathbf{W}^{1}]\mathbf{p}$，其中

$$
{}^{-}\mathbf{W}^{1} = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & 0 & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ 1 & 1 & \cdots & 0 \end{bmatrix} \tag{15.7}
$$

因此对神经元 $i$ 的抑制输入是除了输入向量第 $i$ 个元素的所有元素之和。

**加强中心/抑制周围** 由矩阵${}^{+}\mathbf{W}^{1}$和${}^{-}\mathbf{W}^{1}$定义的连接模式叫做*加强中心/抑制周围模式*。这是因为对神经元 $i$ 的激励输入(使神经元打开)来自集中在同一位置的输入向量的元素(元素 $i$)，同时抑制输入(使神经元关闭)则来自周围各个位置。这种类型的连接模式创造了一种规格化输入模式，正如下面将要讨论的那样。

为了简便，我们将抑制偏置值${}^{-}\mathbf{b}^{1}$设为 0，从而使并联模型的下限为 0，并且将激励偏置值${}^{+}\mathbf{b}^{1}$的所有元素设为相同的数值，即

$$
{}^{+}b_i^1 = {}^{+}b^1, \quad i = 1,2,\cdots,S^1 \tag{15.8}
$$

15-14 因而所有神经元的上限将是相同的。

为了研究第一层规格化的效果，考虑神经元 $i$ 的响应：

$$
\varepsilon \frac{\mathrm{d}n_i^1(t)}{\mathrm{d}t} = -n_i^1(t) + ({}^{+}b^1 - n_i^1(t))p_i - n_i^1(t)\sum_{j \neq i} p_j \tag{15.9}
$$

在稳定状态($\mathrm{d}n_i^1(t)/\mathrm{d}t=0$)我们有

$$0=-n_i^1+({}^+b^1-n_i^1)p_i-n_i^1\sum_{j\neq i}p_j \tag{15.10}$$

如果解出稳定状态神经元输出 $n_i^1$ 可得

$$n_i^1=\frac{{}^+b^1p_i}{1+\sum_{j=1}^{s^1}p_j} \tag{15.11}$$

这时我们定义输入 $i$ 的相对强度为

$$\bar{p}_i=\frac{p_i}{P},\quad 其中\ P=\sum_{j=1}^{s^1}p_j \tag{15.12}$$

于是稳定状态神经元的活跃度可以写成

$$n_i^1=\left(\frac{{}^+b^1P}{1+P}\right)\bar{p}_i \tag{15.13}$$

因此 $n_i^1$ 将与相对强度 $\bar{p}_i$ 成正比，无论总输入 $P$ 的幅度如何。此外，总的神经元活跃度是有界的

$$\sum_{j=1}^{s^1}n_j^1=\sum_{j=1}^{s^1}\left(\frac{{}^+b^1P}{1+P}\right)\bar{p}_j=\left(\frac{{}^+b^1P}{1+P}\right)\leqslant{}^+b^1 \tag{15.14}$$

输入向量已经规格化，从而总的活跃度小于 ${}^+b^1$，同时输入向量的单个元素的相对强度得到了保留。因此，第一层的输出 $n_i^1$，代表相对输入强度 $\bar{p}_i$，而并非总的输入活跃度的同时振动强度 $P$。这种结果是因为采用了加强中心/抑制周围的输入连接模式与并联模型的非线性增益控制。

注意到 Grossberg 网络的第一层解释了人类视觉系统的亮度一致性和亮度对比特征，也就是我们在 15.2.1 节的“视觉规格化”中所讨论的。这种网络对于一个图像的相对强度而非绝对强度是敏感的。而且，试验证明这种加强中心/抑制周围的连接模式是视网膜神经节细胞接收区域的一个有特色的特征[Hube88]。(接收区域是视网膜上的一个区域，那里光感受器馈送信息到特定细胞。图 15-18 表示一个典型的视网膜神经节细胞的加强中心/抑制周围的接收区域。“+”号表示激励区域，“-”号代表抑制区域。这是一个二维的模式，与等式(15.6)及(15.7)所描述的一维连接不同。) 15-15

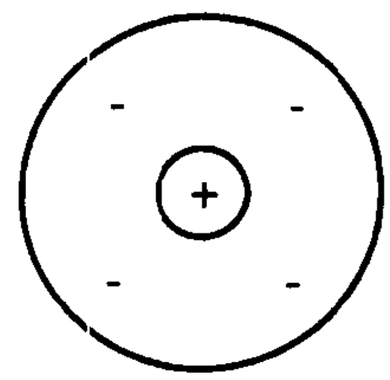


图 15-18

为了示例第一层的性能，考虑有两个神经元的情况，其中 ${}^+b^1$，$\varepsilon=0.1$：

$$(0.1)\frac{\mathrm{d}n_1^1(t)}{\mathrm{d}t}=-n_1^1(t)+(1-n_1^1(t)p_1)-n_1^1(t)p_2 \tag{15.15}$$

$$(0.1)\frac{\mathrm{d}n_2^1(t)}{\mathrm{d}t}=-n_2^1(t)+(1-n_2^1(t)p_2)-n_2^1(t)p_1 \tag{15.16}$$

这个网络对两个不同的输入向量的响应请见图 15-19。对这两个输入向量，第二个元素是第一个元素的 4 倍，尽管第 2 个输入向量的总强度是第一个输入向量的 5 倍(即 50 比 10)。从图 15-19 中可以看到网络的响应保持了输入的相对强度，而同时限制了总的响应。总响应($n_1^1(t)+n_2^1(t)$)将恒小于 1。

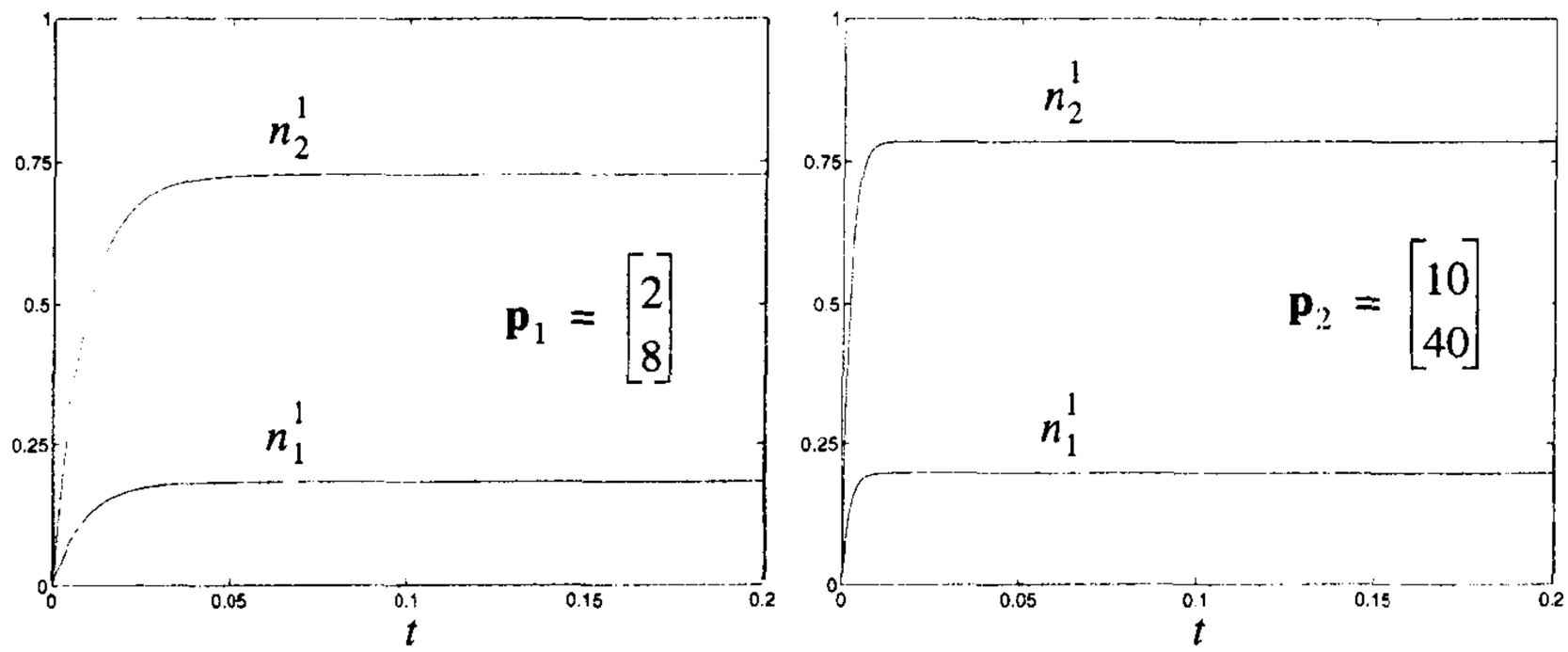


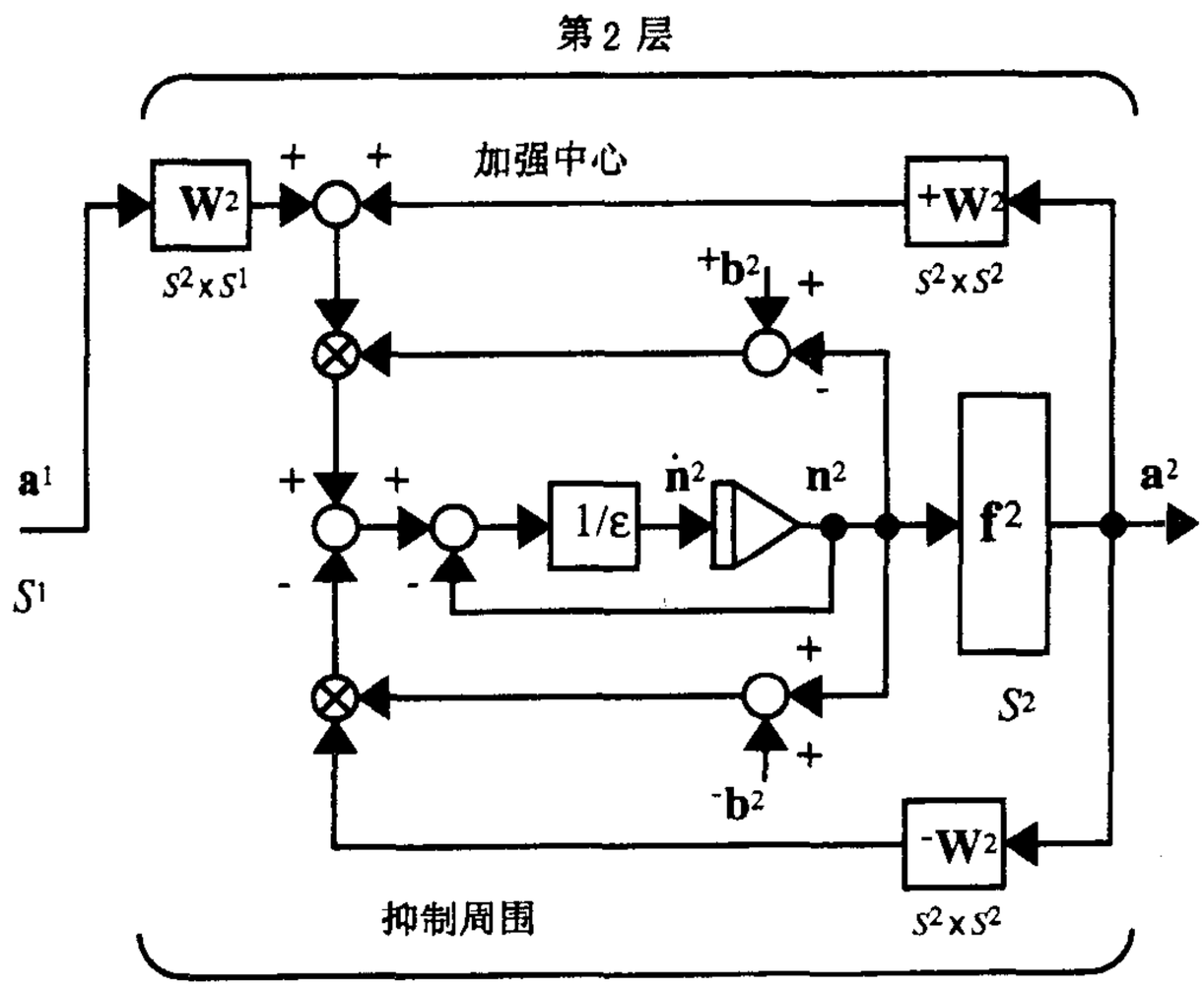

图 15-19 第一层的响应

15-16 试验 Grossberg 网络的第一层请用 *Neural Network Design Demonstration Grossberg Layer 1* (**nnd15gl1**)。

2. 第二层

**短期记忆** Grossberg 网络的第二层，是一个连续的 instar 层，实现几种功能。第一，像第一层那样，规格化这一层的总活跃度。第二，它对模式产生对比度增强，从而获得最大输入的神经元将支配响应。(这种与 Hamming 网络和 Kohonen 网络的"胜者全得"竞争密切相关。)最后，它像短期记忆(STM)那样通过存储对比度增强模式操作。

图 15-20 是第二层的图示。和第一层一样，并联模型是第二层的基础。第二层和第一层的主要区别在于第二层使用反馈式连接。反馈使得网络能够存储模式，即使是在输入撤消之后。反馈也进行竞争，从而产生模式的对比度增强。我们将在下面的讨论中讲解这些特性。




$$\varepsilon d\mathbf{n}^2/dt = -\mathbf{n}^2 + (^+\mathbf{b}^2 - \mathbf{n}^2)\{[^+\mathbf{W}^2]\mathbf{f}^2(\mathbf{n}^2) + \mathbf{W}^2\mathbf{a}^1\} - (\mathbf{n}^2 + {}^-\mathbf{b}^2)[^-\mathbf{W}^2]\mathbf{f}^2(\mathbf{n}^2)$$

图 15-20 Grossberg 网络的第二层

第二层的运算方程是

$$\varepsilon \frac{\mathrm{d}\mathbf{n}^2(t)}{\mathrm{d}t} = -\mathbf{n}^2(t) + ({}^+\mathbf{b}^2 - \mathbf{n}^2(t))\{[{}^+\mathbf{W}^2]\mathbf{f}^2(\mathbf{n}^2(t)) + \mathbf{W}^2\mathbf{a}^1\} - (\mathbf{n}^2(t) + {}^-\mathbf{b}^2)[{}^-\mathbf{W}^2]\mathbf{f}^2(\mathbf{n}^2(t)) \tag{15.17}$$

15-17

这是一个有激励输入$\{[{}^+\mathbf{W}^2]\mathbf{f}^2(\mathbf{n}^2(t)) + \mathbf{W}^2\mathbf{a}^1\}$的并联模型，其中${}^+\mathbf{W}^2 = {}^+\mathbf{W}^1$提供了加强中心的反馈连接，与 Kohonen 网络的权值相似，$\mathbf{W}^2$ 由自适应权值组成。$\mathbf{W}^2$ 的行在训练之后将会代表原型模式。并联模型的抑制输入是$[{}^-\mathbf{W}^2]\mathbf{f}^2(\mathbf{n}^2(t))$，其中${}^-\mathbf{W}^2 = {}^-\mathbf{W}^1$提供了抑制周围的反馈连接。

为了说明第二层的性能，考虑一个由两个神经元组成的层

$$\varepsilon = 0.1, {}^+\mathbf{b}^2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, {}^-\mathbf{b}^2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mathbf{W}^2 = \begin{bmatrix} ({}_1\mathbf{w}^2)^T \\ ({}_2\mathbf{w}^2)^T \end{bmatrix} = \begin{bmatrix} 0.9 & 0.45 \\ 0.45 & 0.9 \end{bmatrix} \tag{15.18}$$

和

$$f^2(n) = \frac{10(n)^2}{1 + (n)^2} \tag{15.19}$$

这一层的运算方程是

$$(0.1)\frac{\mathrm{d}n_1^2(t)}{\mathrm{d}t} = -n_1^2(t) + (1 - n_1^2(t))\{f^2(n_1^2(t)) + ({}_1\mathbf{w}^2)^T\mathbf{a}^1\} - n_1^2(t)f^2(n_2^2(t)) \tag{15.20}$$

$$(0.1)\frac{\mathrm{d}n_2^2(t)}{\mathrm{d}t} = -n_2^2(t) + (1 - n_2^2(t))\{f^2(n_2^2(t)) + ({}_2\mathbf{w}^2)^T\mathbf{a}^1\} - n_2^2(t)f^2(n_1^2(t)) \tag{15.21}$$

**对比度增强** 注意这些等式和 Hamming 网络及 Kohonen 网络的关系。第二层的输入是原型模式(矩阵 $\mathbf{W}^2$ 的行)和第一层的输出(规格化后的输入向量)的内积。最大的内积与输入模式最相近的原型对应。第二层在神经元之中实行竞争，将易于产生输出模式时对比度增强——保持大的输出并使小的输出减弱。这种对比度增强比起 Hamming 网络及 Kohonen 网络来通常要缓和一些。在 Hamming 网络和 Kohonen 网络中，竞争使除了一个以外的所有神经元输出归 0。那个除外的神经元是有最大输入的神经元。在 Grossberg 网络中，竞争保持大的值而削弱小的值，但并无必要使所有的小值归 0。对比度增强的大小是由传输函数 $f^2$ 决定的，下一节我们将看到这点。 15-18

图 15-21 显示了当输入向量 $\mathbf{a}^1 = [0.2 \quad 0.8]^T$ 时第二层的响应(稳定状态的结果从第一层的例子得到)，输入向量加入了 0.25 秒，然后撤离。

这种响应有两个重要的特点。第一，甚至在输入撤消之前，某些对比度增强已经开始产生。第二层的输入是

$$({}_1\mathbf{w}^2)^T\mathbf{a}^1 = [0.9 \quad 0.45]\begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix} = 0.54 \tag{15.22}$$

$$({}_2\mathbf{w}^2)^T\mathbf{a}^1 = [0.45 \quad 0.9]\begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix} = 0.81 \tag{15.23}$$

因此第二个神经元是第一个神经元输入的 1.5 倍。然而在 0.25 秒之后，第二个神经元的输出是第一个神经元输出的 6.34 倍。高与低的对比度急剧地增加了。

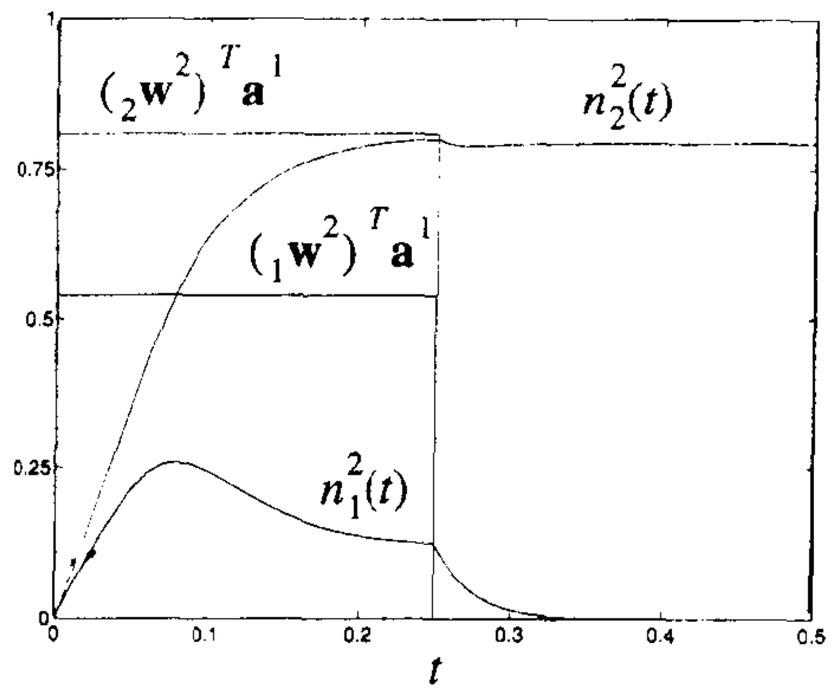



图 15-21 第二层响应

响应的第二个特点是当输入被置 0 的时候，网络进一步增强对比度，并且存储模式。从图 15-21 中可以看出，当输入撤消(0.25 秒)后第一个神经元的输出衰减至 0，而同时第二个神经元的输出达到一个稳定状态值 0.79。即使在输入撤消之后，这个输出仍然得以保存。(Grossberg 将这种行为称作回荡。)正是非线性反馈使得网络存储模式，而且出现引起对比度增强的加强中心/抑制周围的连接模式(由 ${}^+\mathbf{W}^2$ 和 ${}^-\mathbf{W}^2$ 决定)。

15-19

**定向接收区域** 说一点离题的话，注意到我们在 Grossberg 网络的两层都使用了加强中心/抑制周围的结构。对不同的应用可以采用其他的连接模式。例如回想本章早些的时候讨论过的应急切断问题。一种被提议用来实现这种机制的结构是定向接收区域[GrMi89]，如图 15-22 中所示。对于这种结构，“开”(激励)连接来自区域的一边(用蓝色区域表示)，“闭”(抑制)联接来自区域的另一边(用白色区域表示)。

图 15-22

定向接收域的作用过程见图 15-23 所示。当区域与一个边对齐时相应的神经元被激活(大的响应)。如果区域没有与一边对齐，则神经元是不活跃的(小的响应)。这就解释了我们为什么能够感受到一个根本就不存在的边。正如图 15-23 中最右边的接收域所示。

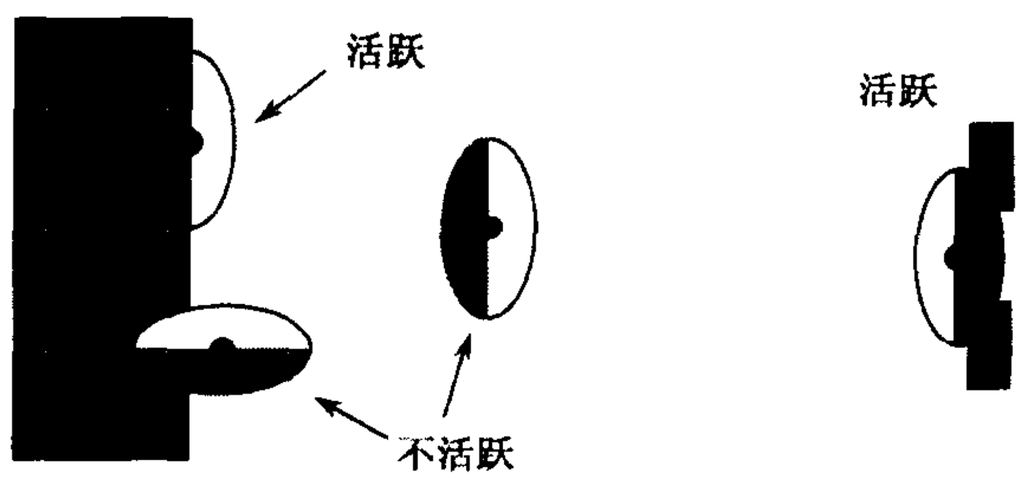



图 15-23 定向接收区域的操作

对定向接收区域及如何将它们加进一个神经网络结构以便提前观察的讨论，请参考[GrMi89]。这篇论文也讨论了特征填充的机制。

3. 传输函数的选择

Grossberg 第二层的行为在很大程度上依赖于传输函数 $f^2(n)$。例如，假定一个输入已经被使用了一段时间，因而输出已经稳定在图15-24中

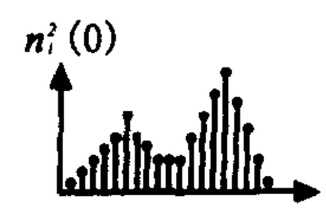



图 15-24

所示的模式。(每个点代表单个神经元的输出。)如果输入被撤消，图 15-25 展示 $f^2(n)$的选择将会怎样影响网络的稳态响应(见[Gross82])。 15-20

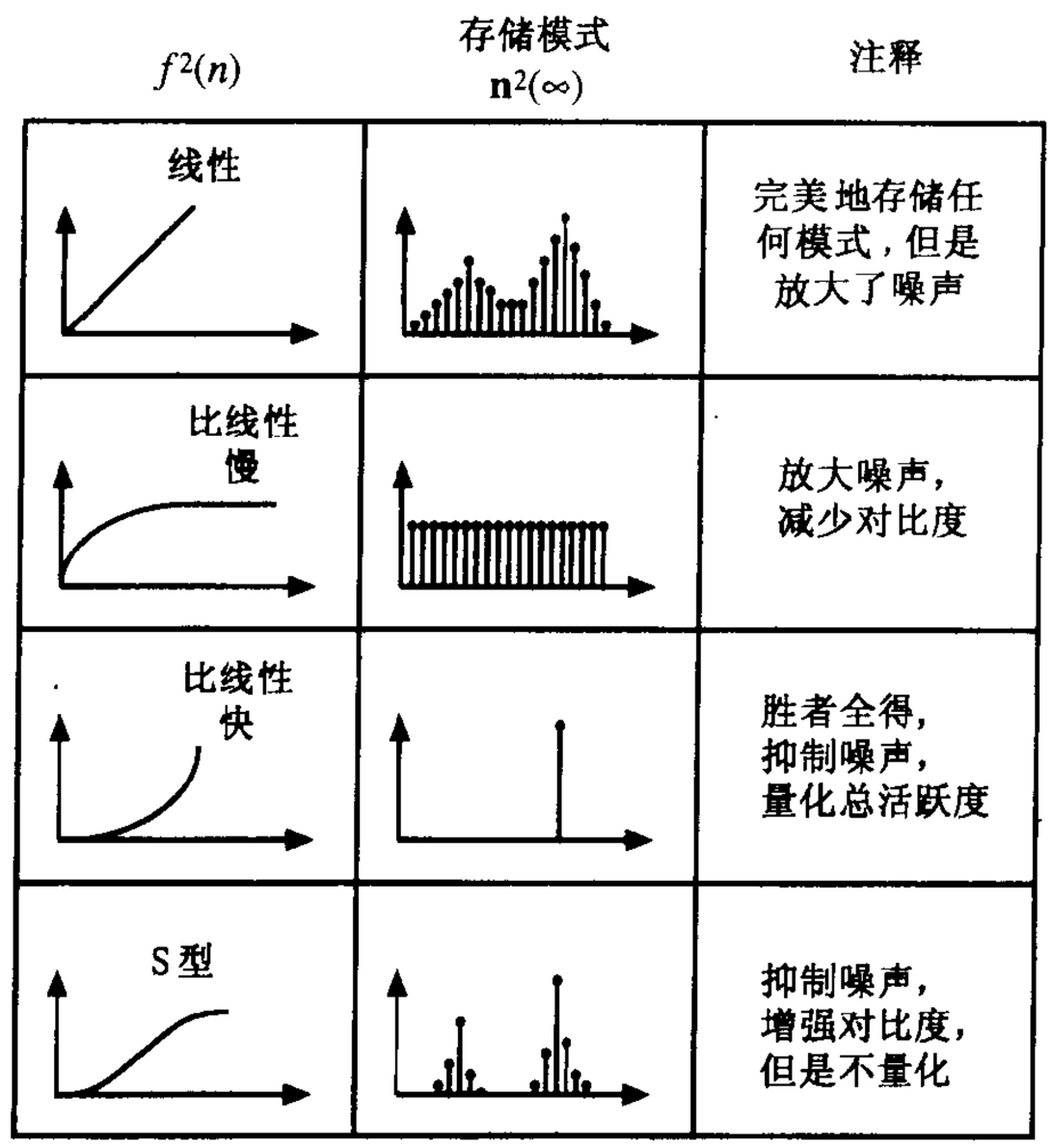



图 15-25 传输函数 $f^2(n)$的作用(摘自[Gross82])

如果传输函数是线性的，则模式被完美的存储。遗憾的是模式中的噪声被放大了，并且和有效输入一样容易被存储了(见例题 P15.6)。如果传输函数是比线性的慢(例如$f^2(n)=1-e^{-n}$)，则稳态响应与初始条件无关，所有以非零值开始的神经元将在稳定状态达到同一水平。所有的对比度都被消除而噪声被放大。

比线性快的传递函数(例如 $f^2(n)=(n)^2$)产生胜者全得的竞争。只有那些有最大初始值的神经元得到存储；所有其他神经元都被设置为 0。这就使噪声的影响达到最小，但使响应量化成有或无信号值(和 Hamming 网络和 Kohonen 网络一样)。

一个 $S$ 型函数对于小信号是比线性快的，对于中等信号是近似线性的，对于大信号是比线性慢的。当一个 $S$ 型传输函数在第二层被使用，模式对比度增强；较大的值被放大，较小的值被缩小。所有小于一定水平(被 Grossberg 称为熄灭阈值[Gross76]的初始神经元输出将衰减到 0。这就将比线性快的传输函数的噪声抑制与线性传输函数所产生的完美存储结合在一起。 15-21

*试验 Grossberg 网络的第二层请用 Neural Network Design Demonstration Grossberg Layer 2* (**nnd15g12**)。

4. 学习规则

**长期记忆** Grossberg 网络的第三个组成部分是自适应权值 $\mathbf{W}^2$ 的学习法则。Grossberg 称这些自适应权值为长期记忆(LTM)。这是因为 $\mathbf{W}^2$ 的行将代表已被存储的而且能够被网络

识别的模式。就像在 Hamming 网络和 Kohonen 网络一样，与输入模式最接近的存储的模式将在第二层产生最大的输出。在下一小节我们将更加详细讨论 Grossberg 网络与 Kohonen 网络的关系。$\mathbf{W}^2$ 的一个学习规则由下式给出：

$$\frac{\mathrm{d}w_{i,j}^2(t)}{\mathrm{d}t} = \alpha\{-w_{i,j}^2(t) + n_i^2(t)n_j^1(t)\} \tag{15.24}$$

方程(15.24)的左边的括号中的第一项是一个被动的衰减项，我们在第一层和第二层的方程中都曾经见过，而第二项实现 Hebb 型学习。这些项一起实现在第 13 章讨论过的带衰减的 Hebb 规则。

回忆在第 13 章中当 $n_i^2(t)$不活跃时关闭学习(并遗忘)常常是很有用的。这可以通过以下学习规则来实现：

$$\frac{\mathrm{d}w_{i,j}^2(t)}{\mathrm{d}t} = \alpha n_i^2(t)\{-w_{i,j}^2(t) + n_j^1(t)\} \tag{15.25}$$

或者用向量形式

$$\frac{\mathrm{d}[{}_i\mathbf{w}^2(t)]}{\mathrm{d}t} = \alpha n_i^2(t)\{-[{}_i\mathbf{w}^2(t)] + \mathbf{n}^1(t)\} \tag{15.26}$$

这里${}_i\mathbf{w}^2(t)$是由 $\mathbf{W}^2$ 的第 $i$ 行的元素所组成的向量(见等式 4.4)。

方程(15.25)右边的项用 $n_i^2(t)$乘，使得学习(并遗忘)只有当 $n_2^2(t)$为非零的时候才会发生。这是第 13 章等式(13.32)所介绍的 instar 学习规则的连续实现。在下面的小节中我们将证明方程(15.25)与式(13.32)等价。为了说明 Grossberg 学习规则的性能，考虑一个每层有 2 个神经元的网络，权值修改方程如下：

15-22

$$\frac{\mathrm{d}w_{1,1}^2(t)}{\mathrm{d}t} = n_1^2(t)\{-w_{1,1}^2(t) + n_1^1(t)\} \tag{15.27}$$

$$\frac{\mathrm{d}w_{1,2}^2(t)}{\mathrm{d}t} = n_1^2(t)\{-w_{1,2}^2(t) + n_2^1(t)\} \tag{15.28}$$

$$\frac{\mathrm{d}w_{2,1}^2(t)}{\mathrm{d}t} = n_2^2(t)\{-w_{2,1}^2(i) + n_1^1(t)\} \tag{15.29}$$

$$\frac{\mathrm{d}w_{2,2}^2(t)}{\mathrm{d}t} = n_2^2(t)\{-w_{2,2}^2(t) + n_2^1(t)\} \tag{15.30}$$

其中学习速度系数 $\alpha$ 被置为 1。为了简化我们的例子，假设两种不同的输入模式以 0.2 秒为周期交替提交给网络。我们还假设，与权值的收敛相比第一层和第二层是非常快的，因此神经元输出在 0.2 秒时已经足够稳定了。第一层和第二层对于不同的输入模式的输出将是

$$\text{对模式 1：}\mathbf{n}^1 = \begin{bmatrix} 0.9 \\ 0.45 \end{bmatrix},\quad \mathbf{n}^2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{15.31}$$

$$\text{对模式 2：}\mathbf{n}^1 = \begin{bmatrix} 0.45 \\ 0.9 \end{bmatrix},\quad \mathbf{n}^2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{15.32}$$

模式 1 用第二层的第 1 个神经元编码，模式 2 用第二层的第二个神经元所编码。

图 15-26 说明了自适应权值的响应，开始时所有的权值都置为 0。注意到权值矩阵的第一行($w_{1,1}^2(t)$和 $w_{1,2}^2(t)$)只在 $n_1^2(t)$是非零的时间里得到调整并且收敛到相应的$\mathbf{n}^1$模式($n_1^1(t)=0.9$和 $n_2^1(t)=0.45$)。(权值矩阵第一行的元素在图15-26中用粗线表示。)而且，权

15-23

值矩阵的第二行($w_{2,1}^2(t)$和 $w_{2,2}^2(t)$)只在 $n_2^2(t)$是非零的时间里得到调整，并且收敛到相应的 $\mathbf{n}^1$ 模式($n_1^1(t)=0.49$ 和 $n_2^1(t)=0.9$)(权值矩阵的第二行在图 15-26 中用细线表示)。

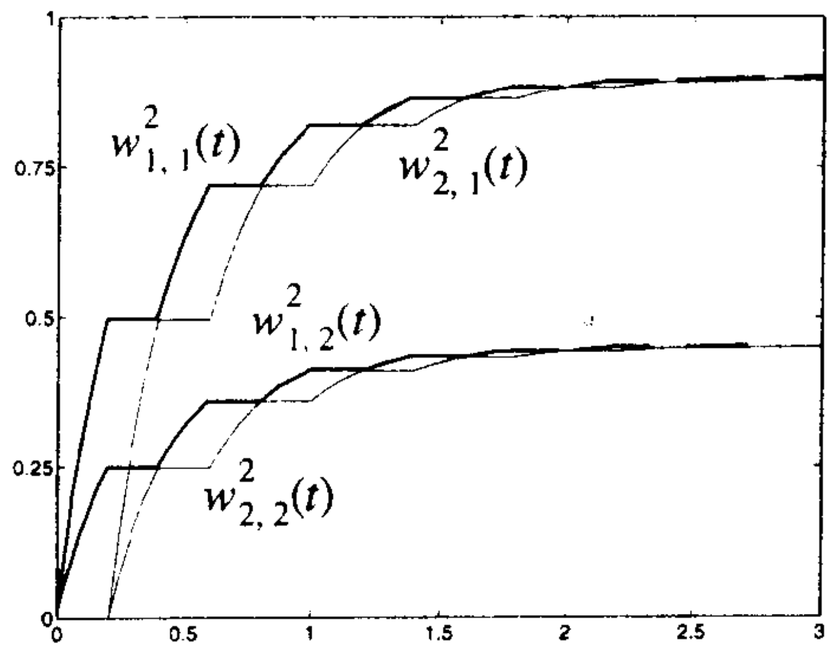



图 15-26 自适应权值的响应

试验自适应权值请用 *Neural Network Design Demonstration Adaptive Weights* (**nnd15aw**)。

### 15.2.4 与 Kohonen 规则的关系

在上一节我们指出 Grossberg 学习规则是第 13 章中介绍的 instar 学习规则的连续实现形式。现在我们来证明这个事实。我们也要证明 Grossberg 网络最简单的形式是第 14 章中介绍的 Kohonen 竞争网络的一个连续实现形式。

首先再写出方程(15.25)的 Grossberg 学习规则：

$$\frac{\mathrm{d}[{}_i\mathbf{w}^2(t)]}{\mathrm{d}t} = \alpha n_i^2(t)\{-[{}_i\mathbf{w}^2(t)] + \mathbf{n}^1(t)\} \tag{15.33}$$

如果用

$$\frac{\mathrm{d}[{}_i\mathbf{w}^2(t)]}{\mathrm{d}t} \approx \frac{{}_i\mathbf{w}^2(t+\Delta t) - {}_i\mathbf{w}^2(t)}{\Delta t} \tag{15.34}$$

作为导数的近似值，则可以把方程(15.33)重写成

$${}_i\mathbf{w}^2(t+\Delta t) = {}_i\mathbf{w}^2(t) + \alpha(\Delta t)n_i^2(t)\{-{}_i\mathbf{w}^2(t) + \mathbf{n}^1(t)\} \tag{15.35}$$

(比较这个等式和第 13 章中由等式(15.33)表示的 instar 规则。)如果整理各项，此式可以转化为 15-24

$${}_i\mathbf{w}^2(t+\Delta t) = \{1-\alpha(\Delta t)n_i^2(t)\}\,{}_i\mathbf{w}^2(t) + \alpha(\Delta t)n_i^2(t)\{\mathbf{n}^1(t)\} \tag{15.36}$$

为了进一步简化分析，假设第二层使用了比线性快的传输函数，因此那个层只有一个神经元能够有非零输出，称之为神经元 $i^*$。于是只有权值矩阵的 $i^*$ 行能够被修改：

$${}_{i^*}\mathbf{w}^2(t+\Delta t) = \{1-\alpha'\}\,{}_{i^*}\mathbf{w}^2(t) + \{\alpha'\}\mathbf{n}^1(t) \tag{15.37}$$

其中 $\alpha' = \alpha(\Delta t) n_{i^*}^2(t)$。

这几乎和我们在第 14 章等式(14.13)介绍的竞争网络的 Kohonen 规则相同。获胜神经元的权值向量(有非零输出)将向当前输入模式的规格化形式 $n^1$ 移近。

在本章提出的 Grossberg 网络和基本的 Kohonen 竞争网络之间有三个主要区别。第一，Grossberg 网络是一种连续网络(满足一组非线性微分方程)。第二，Grossberg 网络的第一层自动规格化输入向量。第三，Grossberg 网络的第二层能够实现一种“软”竞争而不是 Kohonen 网络的那种胜者全得的竞争。这种软竞争使得第二层不只一个神经元能够学习。这使 Grossberg 网络像一个特征图那样运行。

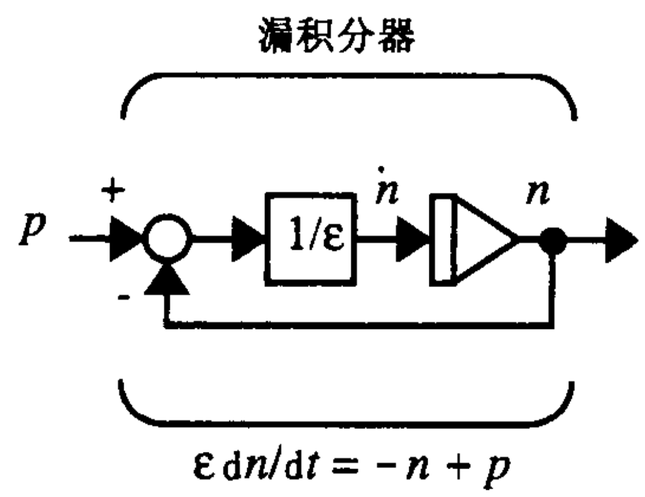
15-25

## 15.3 小结

### 基本的非线性模型

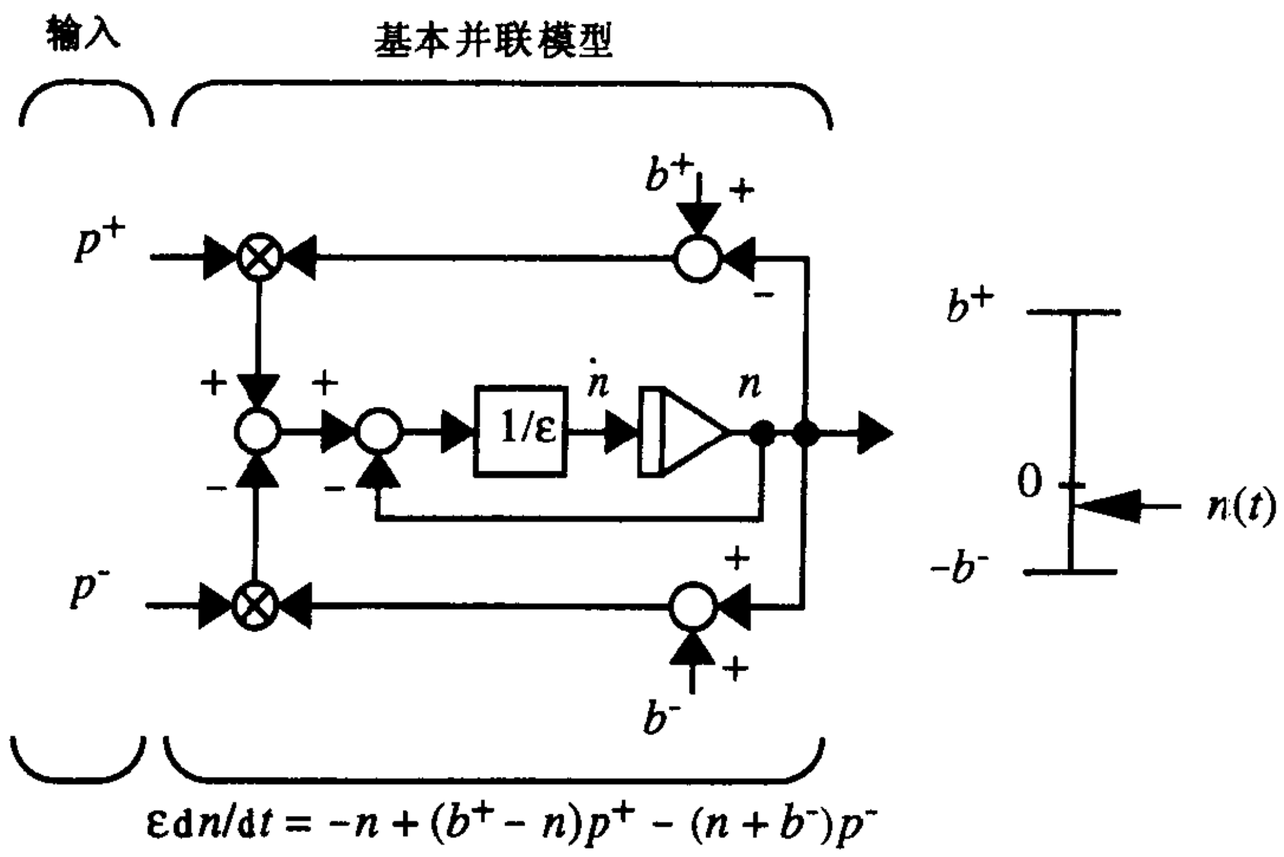
#### 漏积分器

$$\varepsilon \frac{\mathrm{d}n(t)}{\mathrm{d}t} = -n(t) + p(t)$$



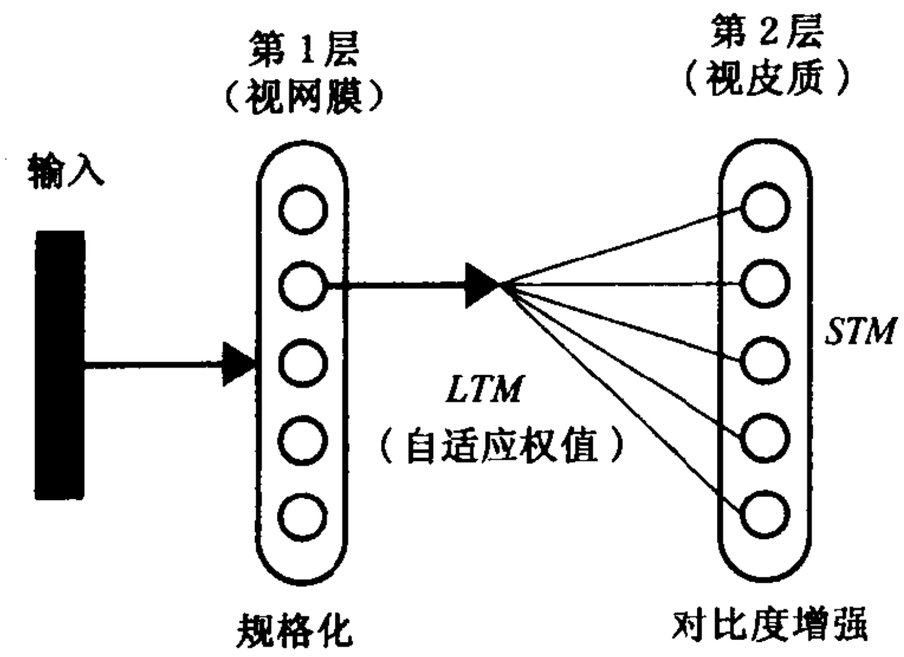

#### 并联模型

$$\varepsilon \frac{\mathrm{d}n(t)}{\mathrm{d}t} = -n(t) + (b^+ - n(t))p^+ - (n(t) + b^-)p^-$$

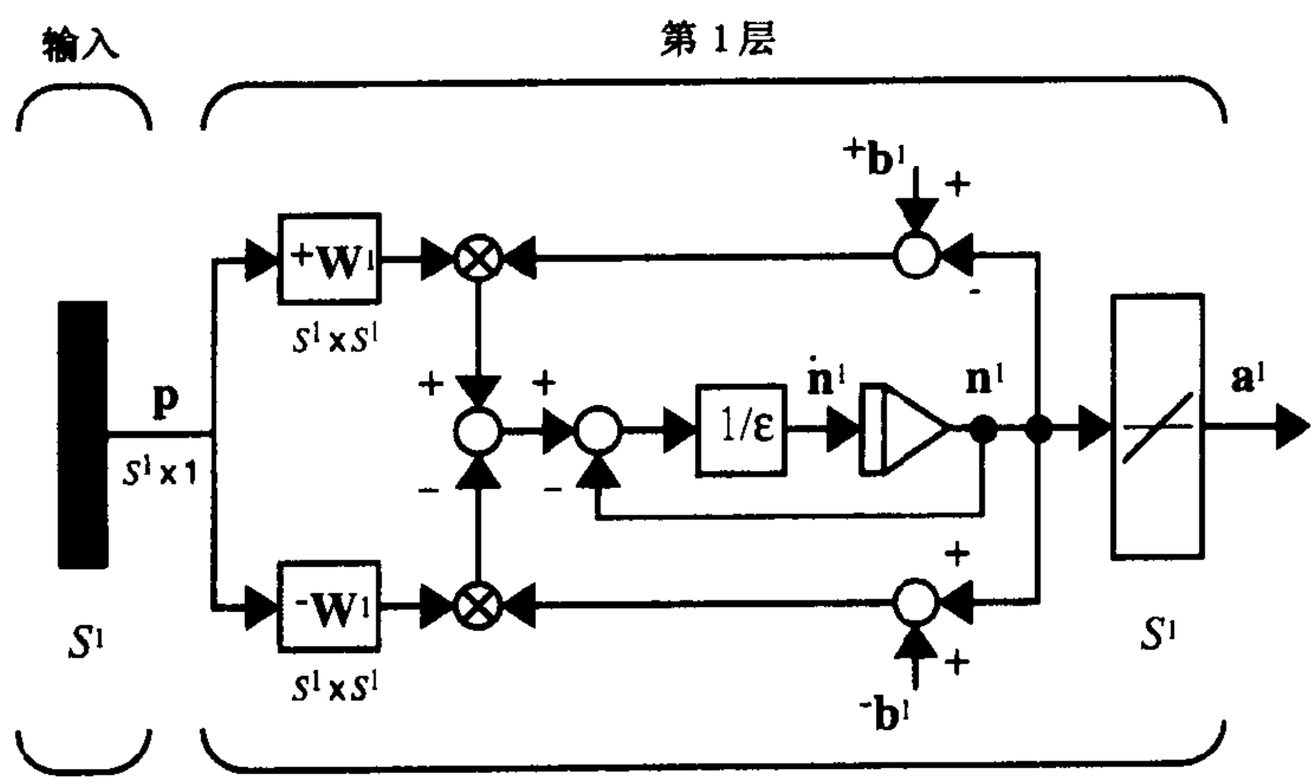



15-26

**两层竞争网络**




**第一层**




$$\varepsilon \frac{\mathrm{d}\mathbf{n}^1(t)}{\mathrm{d}t} = -\mathbf{n}^1(t) + ({}^{+}\mathbf{b}^1 - \mathbf{n}^1(t))[{}^{+}\mathbf{W}^1]\mathbf{p} - (\mathbf{n}^1(t) + {}^{-}\mathbf{b}^1)[{}^{-}\mathbf{W}^1]\mathbf{p}$$

$$ {}^{+}\mathbf{W}^1 = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \qquad {}^{-}\mathbf{W}^1 = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & 0 & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ 1 & 1 & \cdots & 0 \end{bmatrix} \tag{15-27}$$

加强中心　　抑制周围

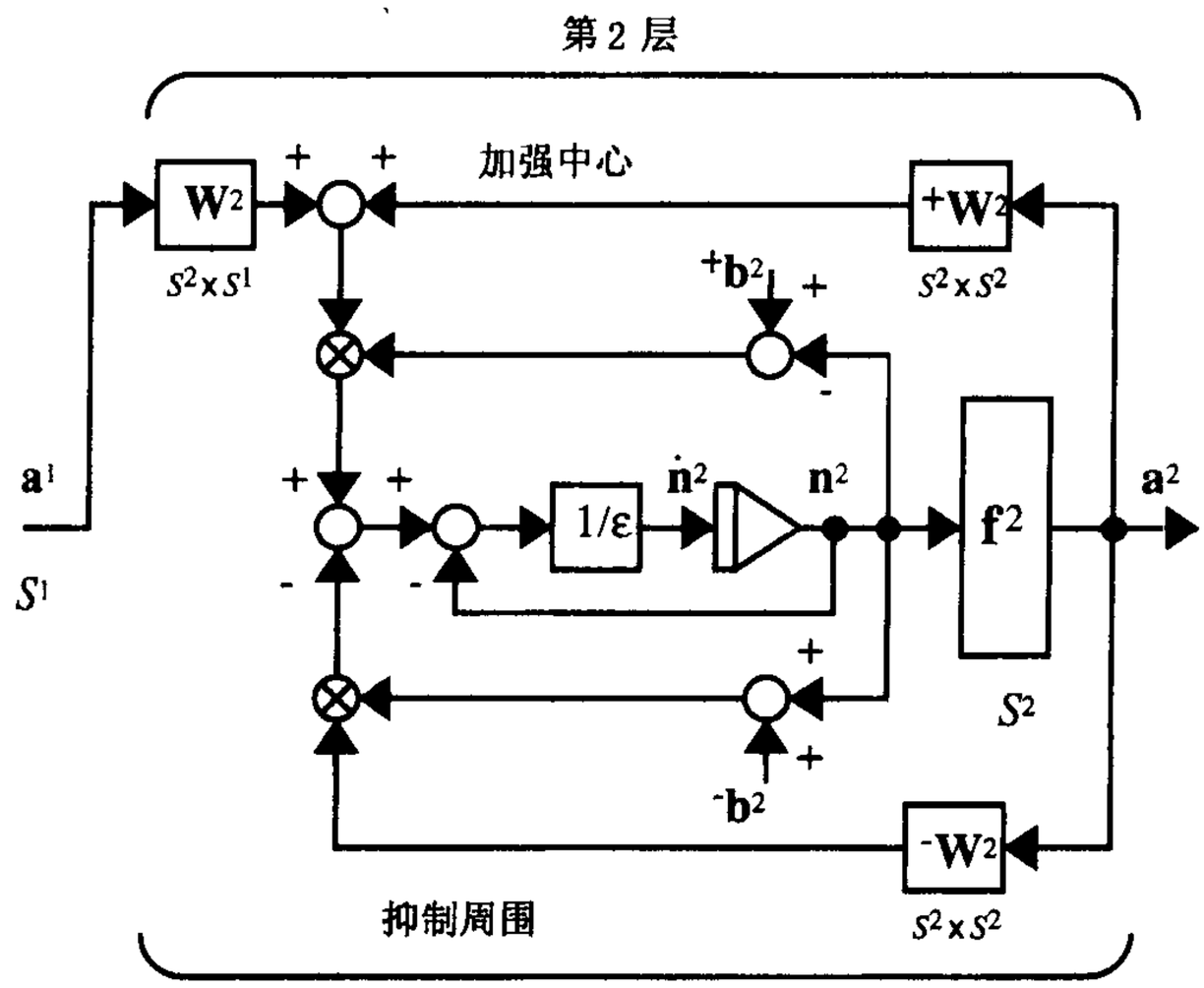
稳定状态神经元的活跃度

$$n_i^1 = \left(\frac{{}^{+}b^1 P}{1+P}\right)\bar{p}_i, \text{其中 } \bar{p}_i = \frac{p_i}{P} \text{ 且 } P = \sum_{j=1}^{s^1} p_j$$

**第二层**

$$\varepsilon \mathrm{d}\mathbf{n}^2/\mathrm{d}t = -\mathbf{n}^2 + ({}^{+}\mathbf{b}^2 - \mathbf{n}^2)\{[{}^{+}\mathbf{W}^2]\mathbf{f}^2(\mathbf{n}^2) + \mathbf{W}^2\mathbf{a}^1\} - (\mathbf{n}^2 + {}^{-}\mathbf{b}^2)[{}^{-}\mathbf{W}^2]\mathbf{f}^2(\mathbf{n}^2)$$

$$\varepsilon \frac{d\mathbf{n}^2(t)}{dt} = -\mathbf{n}^2(t) + ({}^{+}\mathbf{b}^2 - \mathbf{n}^2(t))\{[{}^{+}\mathbf{W}^2]\mathbf{f}^2(\mathbf{n}^2(t)) + \mathbf{W}^2\mathbf{a}^1\}$$
$$- (\mathbf{n}^2(t) + {}^{-}\mathbf{b}^2)[{}^{-}\mathbf{W}^2]\mathbf{f}^2(\mathbf{n}^2(t))$$




15-28

**传输函数的选择**

| $f^2(n)$ | 存储模式 $\mathbf{n}^2(\infty)$ | 注释 |
|---|---|---|
| 线性 | | 完美地存储任何模式，但是放大了噪声 |
| 比线性慢 | | 放大噪声，减少对比度 |
| 比线性快 | | 胜者全得，抑制噪声，量化总活跃度 |
| S型 | | 抑制噪声，增强对比度，但是不量化 |

**学习规则**

$$\frac{d[{}_i\mathbf{w}^2(t)]}{dt} = \alpha n_i^2(t)\{-[{}_i\mathbf{w}^2(t)] + \mathbf{n}^1(t)\}$$

15-29

（连续的 instar 学习）

## 15.4 例题

**P15.1** 演示漏积分器的性能系数 ε 的作用，见图 15-27 中所示，输入 $p = 1$。

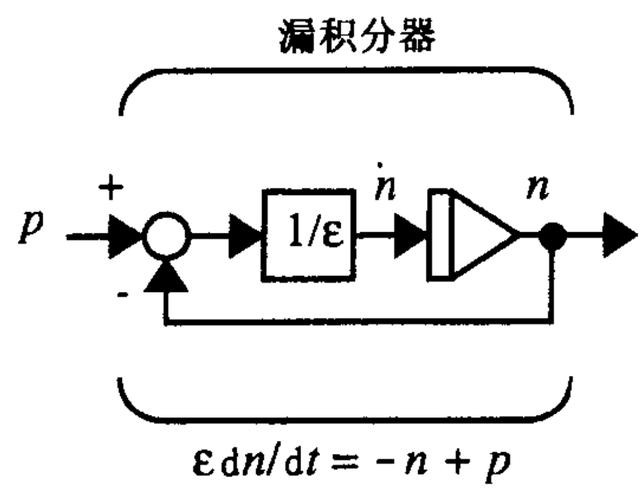



图 15-27 漏积分器

**解**

漏积分器的运算方程是

$$\varepsilon \frac{\mathrm{d}n(t)}{\mathrm{d}t} = -n(t) + p(t)$$

这个微分方程对任一个输入 $p(t)$的解是

$$n(t) = e^{-t/\varepsilon}n(0) + \frac{1}{\varepsilon}\int_0^t e^{-(t-\tau)/\varepsilon}p(t-\tau)\mathrm{d}\tau$$

如果 $p(t) = 1$，解将是

$$n(t) = e^{-t/\varepsilon}n(0) + \frac{1}{\varepsilon}\int_0^t e^{-(t-\tau)/\varepsilon}\mathrm{d}\tau$$

我们想说明这个响应随 ε 而改变。响应将是

$$n(t) = e^{-t/\varepsilon}n(0) + (1 - e^{-t/\varepsilon}) = e^{-t/\varepsilon}(n(0) - 1) + 1$$

响应从 $n(0)$开始，然后以指数形式增长(或以指数形式衰减，取决于 $n(0)$是大于 1 还是小于 1)接近 $n(\infty) = 1$ 的稳态响应。随着 ε 减少，响应变得更快(因为 $e^{-t/\varepsilon}$衰减得更快)，而稳态值保持不变。图 15-28 说明当 $\varepsilon = 1$，0.5，0.25，0.125，$n(0) = 0$ 时的响应。请注意稳态值对每种情况都保持 1。只有反应的速度发生变化。 15-30

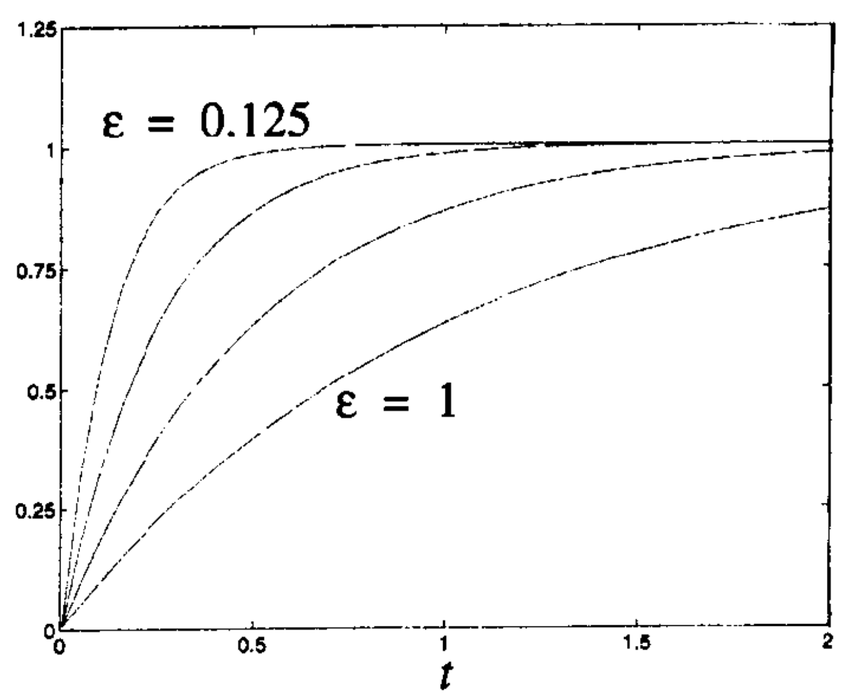



图 15-28 ε 对漏积分器响应的作用

**P15.2** 再次利用图 15-27 中的漏积分器，设 $\varepsilon = 1$。

（i）找出一个近似于漏积分器微分方程的差分方程，通过用下式估计导数。

$$\frac{\mathrm{d}n(t)}{\mathrm{d}t} \approx \frac{n(t+\Delta t) - n(t)}{\Delta t}$$

（ii）用 $\Delta t = 0.1$，比较这个差分方程的响应和 $p(t) = 1$ 和 $n(0) = 0$ 微分方程的响应，在 $0 < t < 1$ 区域比较这两者。

（iii）使用漏积分器的差分方程模型，证明响应是以前输入的加权平均。

**解**

（i）如果对导数作近似，我们发现

$$\frac{n(t+\Delta t) - n(t)}{\Delta t} = -n(t) + p(t)$$

或

$$n(t+\Delta t) = n(t) + \Delta t\{-n(t) + p(t)\} = (1-\Delta t)n(t) + (\Delta t)p(t)$$

（ii）如果令 $\Delta t = 0.1$，我们得到差分方程

$$n(t+0.1) = 0.9n(t) + 0.1p(t)$$

15-31 如果令 $p(t) = 1$ 和 $n(0) = 0$，那么我们可以解 $n(t)$得到：

$$n(0.1) = 0.9n(0) + 0.1p(0) = 0.1$$
$$n(0.2) = 0.9n(0.1) + 0.1p(0.1) = 0.9(0.1) + 0.1(1) = 0.19$$
$$n(0.3) = 0.9n(0.2) + 0.1p(0.2) = 0.9(0.19) + 0.1(1) = 0.271$$
$$n(0.4) = 0.9n(0.3) + 0.1p(0.3) = 0.9(0.271) + 0.1(1) = 0.3439$$
$$n(0.5) = 0.9n(0.4) + 0.1p(0.4) = 0.9(0.3439) + 0.1(1) = 0.4095$$
$$n(0.6) = 0.4686 \qquad n(0.7) = 0.5217 \qquad n(0.8) = 0.5695$$
$$n(0.9) = 0.6126 \qquad n(1.0) = 0.6513$$

从 P15.1 微分方程的解是

$$n(t) = e^{-t/\varepsilon}n(0) + (1 - e^{-t/\varepsilon}) = (1 - e^{-t})$$

图 15-29 展示了微分方程解与差分方程解之间的关系。曲线代表微分方程的解，圆圈代表差分方程的解。这两个解十分接近，并且能够通过缩短间隔 $\Delta t$ 而任意的接近。

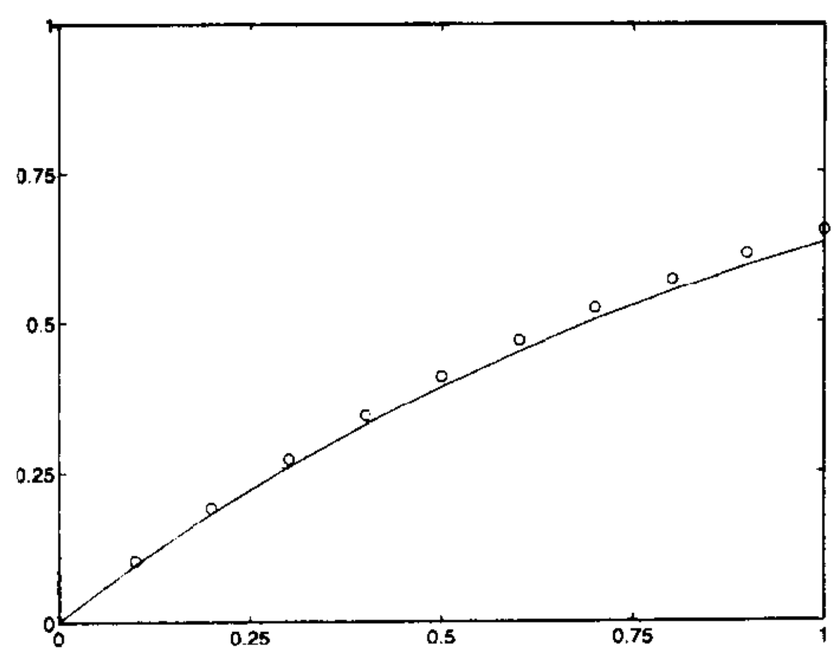



图 15-29 差分方程与微分方程的比较

（iii）再次考虑我们在(ii)题中得到的漏积分器的差分方程模型：

$$n(t+0.1)=0.9n(t)+0.1p(t)$$

如果以 0 为初始条件，我们得到

$$n(0.1)=0.9n(0)+0.1p(0)=0.1p(0)$$ 15-32

$$n(0.2)=0.9n(0.1)+0.1p(0.1)=0.9\{0.1p(0)\}+0.1p(0.1)=0.09p(0)+0.1p(0.1)$$

$$n(0.3)=0.9n(0.2)+0.1p(0.2)=0.081p(0)+0.09p(0.1)+0.1p(0.2)$$

$$\vdots$$

$$n(k0.1)=0.1\{(0.9)^{k-1}p(0)+(0.9)^{k-2}p(0.1)+\cdots+p((k-1)0.1)\}$$

因此漏积分器的响应是以前输入 $p(0)$，$p(0.1)$，…，$p((k-1)0.1)$的加权平均值。注意当前的输入对响应的贡献比早些输入的大。

**P15.3** 找出图 P15.4 所示的并联网络的响应，其中 $\varepsilon=1$，$b^+=1$，$b^{-1}=1$，$p^+=0$，$p^-=10$，$n(0)=0.5$。

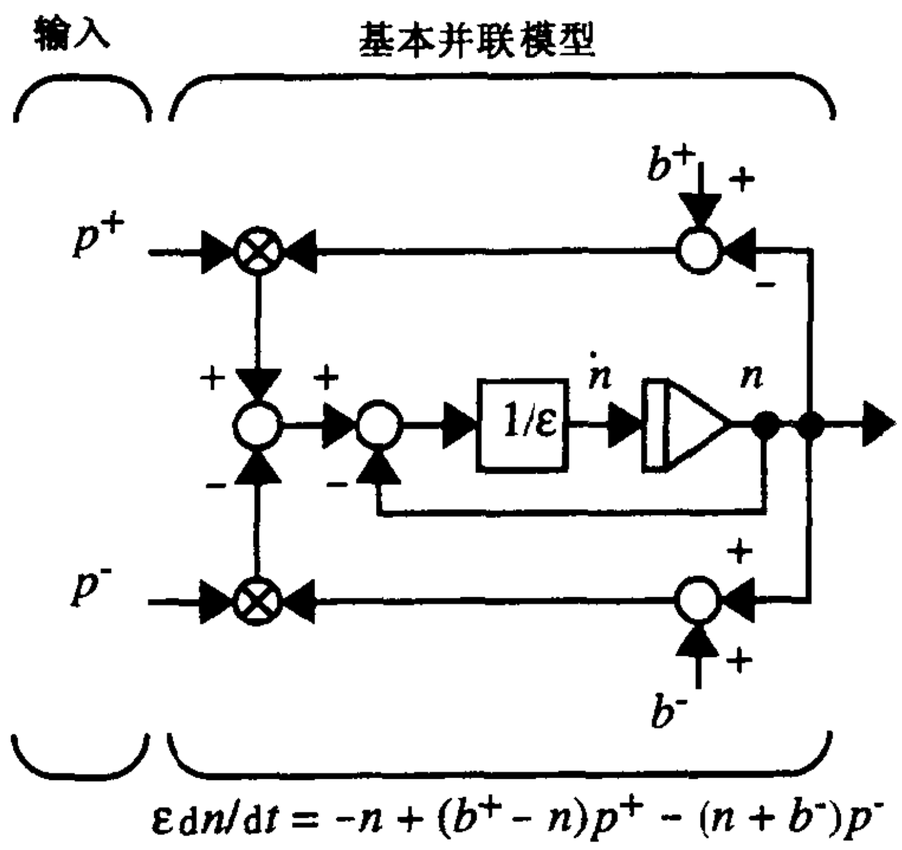



图 15-30 并联网络

**解**

并联网络的运算方程是

$$\varepsilon\frac{dn(t)}{dt}=-n(t)+(b^+-n(t))p^+-(n(t)+b^-)p^-$$

对于那些提供的参数值上式变为

$$\frac{dn(t)}{dt}=-n(t)-(n(t)+1)10=-11n(t)-10$$ 15-33

这个方程的解是

$$n(t)=e^{-11t}n(0)+\int_0^t e^{-11(t-\tau)}(-10)d\tau$$

或

$$n(t)=e^{-11t}0.5+\left(-\frac{10}{11}\right)(1-e^{-11t})$$

响应见图 15-31。

对这种响应有两件事需要注意。第一，和所有的并联网络一样，响应永远不会降到

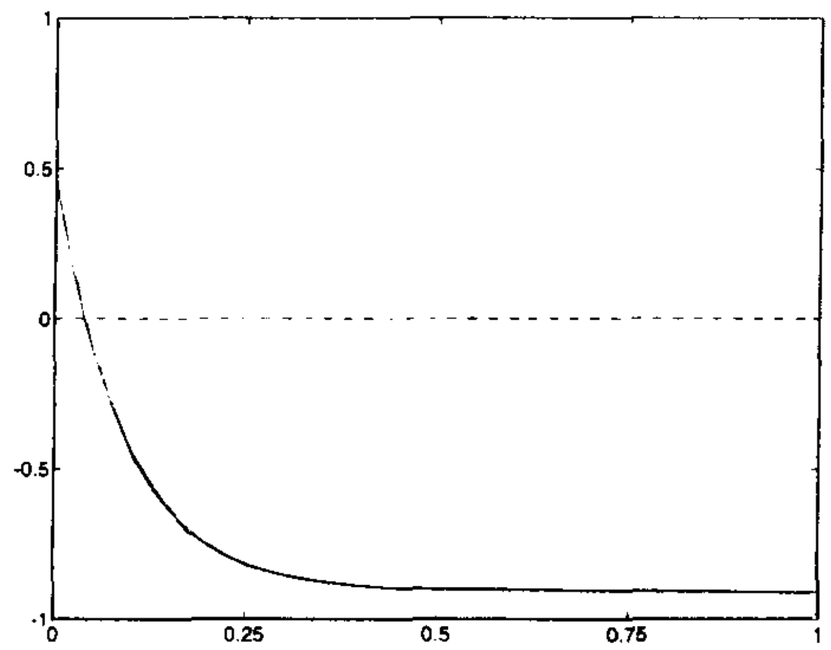



图 15-31 并联网络响应

$-b^-$以下，在本题中就是 $-1$。随着抑制输入 $p^-$增加，稳态响应将会降低，但永远不可能低于 $-b^-$。响应的第二个特点是当输入增加的时候，响应的速度也会随之增加。例如，如果输入从 $p^- = 10$ 增加到 $p^- = 100$，则响应将是

$$n(t) = e^{-101t}0.5 + \left(-\frac{100}{101}\right)(1 - e^{-101t})$$

因为 $e^{-101t}$比 $e^{-11t}$衰减更快，响应将会变得更快。

**P15.4** 找出 Grossberg 网络第一层在 2 个神经元的情况下的响应，其中 $^+b^1 = 1$，$^-b^1 = 0$，$\varepsilon = 1$。输入向量 $\mathbf{p} = [c \quad 2c]^T$。假设初始条件设为 0。演示 $c$ 对响应的作用。

**解**

在这种情况下第一层的微分方程的是

15-34

$$\frac{dn_1^1(t)}{dt} = -n_1^1(t) + (1 - n_1^1(t))(c) - n_1^1(t)(2c) = -(1 + 3c)n_1^1(t) + c$$

$$\frac{dn_2^1(t)}{dt} = -n_2^1(t) + (1 - n_2^1(t))(2c) - n_2^1(t)(c) = -(1 + 3c)n_2^1(t) + 2c$$

这些方程的解是

$$n_1^1(t) = e^{-(1+3c)t}n_1^1(0) + \int_0^t e^{-(1+3c)(t-\tau)}(c)d\tau$$

$$n_2^1(t) = e^{-(1+3c)t}n_2^1(0) + \int_0^t e^{-(1+3c)(t-\tau)}(2c)d\tau$$

如果初始条件设为 0 的话，这些方程简化为

$$n_1^1(t) = \left(\frac{c}{1 + 3c}\right)(1 - e^{-(1+3c)t})$$

$$n_2^1(t) = \left(\frac{2c}{1 + 3c}\right)(1 - e^{-(1+3c)t})$$

注意第一层的输出保留和输入相同的相对强度；神经元 2 的输出通常是神经元 1 输出的 2 倍。这种情况与等式(15.13)一致。而且，总的输出强度($n_1^1(t) + n_2^1(t)$)从来没有超过 $^+b^1 = 1$，如式(15.14)中所预见的那样。随 $c$ 的增加，它对响应有两种影响。第一，稳态值略有增加。第二，响应变得更快了，因为 $e^{-(1+3c)t}$衰减比 $c$ 增加更快。

**P15.5** 考虑 Grossberg 网络的第二层。假设第二层的输入已经加了一段时间然后才撤除

(置为 0)。

(i) 找出在第二层的输入被撤消之后，描述第二层总输出

$$N^2(t) = \sum_{k=1}^{S^2} n_k^2(t)$$

变化的微分方程

(ii) 找出在第二层的输入对消之后，描述第二层相对输出

$$\bar{n}_i^2(t) = \frac{n_i^2(t)}{N^2(t)}$$

变化的微分方程 15-35

**解**

(i) 第二层的运算由方程(15.17)描述：

$$\varepsilon \frac{\mathrm{d}\mathbf{n}^2(t)}{\mathrm{d}t} = -\mathbf{n}^2(t) + ({}^+\mathbf{b}^2 - \mathbf{n}^2(t))\{[{}^+\mathbf{W}^2]\mathbf{f}^2(\mathbf{n}^2(t)) + \mathbf{W}^2\mathbf{a}^1\}$$
$$- (\mathbf{n}^2(t) + {}^-\mathbf{b}^2)[{}^-\mathbf{W}^2]\mathbf{f}^2(\mathbf{n}^2(t))$$

如果输入被撤消，则 $\mathbf{W}^2\mathbf{a}^1$ 是 0。为了简化起见，我们将设抑制偏置值 ${}^-\mathbf{b}^2$ 为 0，而且我们设置激励偏置值 ${}^+\mathbf{b}^2$ 的所有元素为 ${}^+b^2$。神经元 $i$ 的响应由下式给出：

$$\varepsilon \frac{\mathrm{d}n_i^2(t)}{\mathrm{d}t} = -n_i^2(t) + ({}^+b^2 - n_i^2(t))\{f^2(n_i^2(t))\} - n_i^2(t)\{\sum_{k\neq i} f^2(n_k^2(t))\}$$

这可以重新组合产生

$$\varepsilon \frac{\mathrm{d}n_i^2(t)}{\mathrm{d}t} = -n_i^2(t) + {}^+b^2\{f^2(n_i^2(t))\} - n_i^2(t)\left\{\sum_{k=1}^{S^2} f^2(n_k^2(t))\right\}$$

如果定义

$$F^2(t) = \sum_{k=1}^{S^2} f^2(n_k^2(t))$$

我们可以将方程简化为

$$\varepsilon \frac{\mathrm{d}n_i^2(t)}{\mathrm{d}t} = -(1 + F^2(t))n_i^2(t) + {}^+b^2\{f^2(n_i^2(t))\}$$

为了得到总活跃度，这个等式对 $i$ 求和产生

$$\varepsilon \frac{\mathrm{d}N^2(t)}{\mathrm{d}t} = -(1 + F^2(t))N^2(t) + {}^+b^2\{F^2(t)\}$$

这个方程描述了第二层的总活跃度随时间的变化。

(ii) 相对活跃度的导数是

$$\frac{\mathrm{d}}{\mathrm{d}t}[\bar{n}_i^2(t)] = \frac{\mathrm{d}}{\mathrm{d}t}\left[\frac{n_i^2(t)}{N^2(t)}\right] = \frac{1}{N^2(t)}\frac{\mathrm{d}}{\mathrm{d}t}[n_i^2(t)] - \left[\frac{n_i^2(t)}{(N^2(t))^2}\right]\frac{\mathrm{d}}{\mathrm{d}t}[N^2(t)]$$

如果将以前的方程替代这些导数，我们得到 15-36

$$\varepsilon \frac{\mathrm{d}}{\mathrm{d}t}[\bar{n}_i^2(t)] = \frac{1}{N^2(t)}\Big[\{-(1 + F^2(t))n_i^2(t) + {}^+b^2\{f^2(n_i^2(t))\}\}$$
$$- \frac{n_i^2(t)}{N^2(t)}\{-(1 + F^2(t))N^2(t) + {}^+b^2\{F^2(t)\}\}\Big]$$

消去右边两项产生

$$\varepsilon\frac{\mathrm{d}}{\mathrm{d}t}[\bar{n}_i^2(t)] = \frac{1}{N^2(t)}\left[\{+b^2\{f^2(n_i^2(t))\}\} - \frac{n_i^2(t)}{N^2(t)}\{+b^2\{F^2(t)\}\}\right]$$

或

$$\varepsilon\frac{\mathrm{d}}{\mathrm{d}t}[\bar{n}_i^2(t)] = \frac{+b^2F^2(t)}{N^2(t)}\left[\frac{f^2(n_i^2(t))}{F^2(t)} - \frac{n_i^2(t)}{N^2(t)}\right]$$

如果展开括号中的项，可以将此式改写成一种更为有用的形式：

$$\begin{aligned}\left[\frac{f^2(n_i^2(t))}{F^2(t)} - \frac{n_i^2(t)}{N^2(t)}\right] &= \frac{1}{F^2(t)N^2(t)}[f^2(n_i^2(t))N^2(t) - n_i^2(t)F^2(t)]\\ &= \frac{1}{F^2(t)N^2(t)}\left[g^2(n_i^2(t))n_i^2(t)\sum_{k=1}^{s^2}n_k^2(t) - n_i^2(t)\sum_{k=1}^{s^2}g^2(n_k^2(t))n_k^2(t)\right]\\ &= \frac{n_i^2(t)}{F^2(t)N^2(t)}\left[\sum_{k=1}^{s^2}n_k^2(t)[g^2(n_i^2(t)) - g^2(n_k^2(t))]\right]\end{aligned}$$

其中

$$g^2(n_i^2(t)) = \frac{f^2(n_i^2(t))}{n_i^2(t)}$$

将这种表达式与以前的等式相结合我们得到

$$\varepsilon\frac{\mathrm{d}}{\mathrm{d}t}[\bar{n}_i^2(t)] = +b^2\bar{n}_i^2(t)\left[\sum_{k=1}^{s^2}\bar{n}_k^2(t)[g^2(n_i^2(t)) - g^2(n_k^2(t))]\right]$$

这种形式描述输出相对强度展开的微分方程对于展示第二层的特点是很有用的，就像我们在解下一题目时将会见到的那样。

15-37

**P15.6** 假设 Grossberg 网络第二层的传输函数是线性的。

（i）证明当输入被撤消之后，第二层的相对输出不会变化。

（ii）在什么条件下第二层的总输出在输入被撤消之后会衰减到 0?

**解**

（i）从 P15.5 中我们知道在输入被撤消之后第二层的相对输出将根据下式展开：

$$\varepsilon\frac{\mathrm{d}}{\mathrm{d}t}[\bar{n}_i^2(t)] = +b^2\bar{n}_i^2(t)\left[\sum_{k=1}^{s^2}\bar{n}_k^2(t)[g^2(n_i^2(t)) - g^2(n_k^2(t))]\right]$$

如果第二层的传递函数 $f^2(n)$是线性的，则

$$f^2(n) = cn$$

因此

$$g^2(n) = \frac{f^2(n)}{n} = \frac{cn}{n} = c$$

如果把这个表达式代入微分方程，我们得到

$$\varepsilon\frac{\mathrm{d}}{\mathrm{d}t}[\bar{n}_i^2(t)] = +b^2\bar{n}_i^2(t)\left[\sum_{k=1}^{s^2}\bar{n}_k^2(t)[c - c]\right] = 0$$

因而相对输出并没有发生变化。

（ii）在输出被撤消之后，P15.5 中第二层的总输出按下式展开：

$$\varepsilon \frac{\mathrm{d}N^2(t)}{\mathrm{d}t} = -(1 + F^2(t))N^2(t) + {}^{+}b^2\{F^2(t)\}$$

如果 $f^2(n)$是线性的，则

$$F^2(t) = \sum_{k=1}^{S^2} f^2(n_k^2(t)) = \sum_{k=1}^{S^2} cn_k^2(t) = c\sum_{k=1}^{S^2} n_k^2(t) = cN^2(t)$$

因此微分方程可以写作

$$\varepsilon \frac{\mathrm{d}N^2(t)}{\mathrm{d}t} = -(1 + cN^2(t))N^2(t) + {}^{+}b^2\{cN^2(t)\} = -\{1 - {}^{+}b^2c + cN^2(t)\}N^2(t) \qquad \boxed{15\text{-}38}$$

为了找到这个方程的平衡解，我们将导数置为 0：

$$0 = -\{1 - {}^{+}b^2c + cN^2(t)\}N^2(t)$$

因此有两个平衡解：

$$N^2(t) = 0 \quad 或 \quad N^2(t) = \frac{{}^{+}b^2c - 1}{c}$$

我们想知道在哪种条件下总输出将会收敛到这些可能的解。考虑两种情况：

1. $1 \geqslant {}^{+}b^2c$

对于这种情况，总输出的导数是

$$\varepsilon \frac{\mathrm{d}N^2(t)}{\mathrm{d}t} = -\{1 - {}^{+}b^2c + cN^2(t)\}N^2(t)$$

对于正的 $N^2(t)$将永远为负(回想第二层的输出永远非负)。因此，总输出将会衰减至 0：

$$\lim_{t\to\infty} N^2(t) = 0$$

2. $1 < {}^{+}b^2c$

(a) 如果 $N^2(0) > ({}^{+}b^2c - 1)/c$，那么总输出的导数将为负，直到 $N^2(t) = ({}^{+}b^2c - 1)/c$，当导数变为 0 时。因此

$$\lim_{t\to\infty} N^2(t) = \frac{({}^{+}b^2c - 1)}{c}$$

(b) 如果 $N^2(0) < ({}^{+}b^2c - 1)/c$，那么总输出的导数将为正，直到 $N^2(t) = ({}^{+}b^2c - 1)/c$，当导数变为 0 时。因此

$$\lim_{t\to\infty} N^2(t) = \frac{({}^{+}b^2c - 1)}{c}$$

所以，如果第二层的传输函数是线性的，那么如果 $1 \geqslant {}^{+}b^2c$，则总输出将衰减至 0。如果 $1 < {}^{+}b^2c$，则总输出将收敛于$({}^{+}b^2c - 1)/c$。在任何情况下，相对输出将保持不变。 15-39

作为这些结果的例子，考虑如下第二层的方程组：

$$\frac{\mathrm{d}n_1^2(t)}{\mathrm{d}t} = -n_1^2(t) + (1.5 - n_1^2(t))\{n_1^2(t)\} - n_1^2(t)\{n_2^2(t)\}$$

$$\frac{\mathrm{d}n_2^2(t)}{\mathrm{d}t} = -n_2^2(t) + (1.5 - n_2^2(t))\{n_2^2(t)\} - n_2^2(t)\{n_1^2(t)\}$$

在这种情况下，$\varepsilon = 1$，${}^{+}b^2 = 1.5$，$c = 1$，因而 $1 < {}^{+}b^2c$。总输出将收敛于

$$\lim_{t\to\infty} N^2(t) = \frac{({}^{+}b^2c - 1)}{c} = \frac{(1.5 - 1)}{1} = 0.5$$

在图 15-32 中我们能够看到第二层对于两组不同初始条件响应。

$$\mathbf{n}^2(0) = \begin{bmatrix} 0.75 \\ 0.5 \end{bmatrix} \quad 和 \quad \mathbf{n}^2(0) = \begin{bmatrix} 0.15 \\ 0.1 \end{bmatrix}$$

正如所预计的那样，两种初始条件下总输出都收敛于 0.5。而且，因为两种初始条件下的相对值是相同的，两种条件下输出收敛于同样的值。

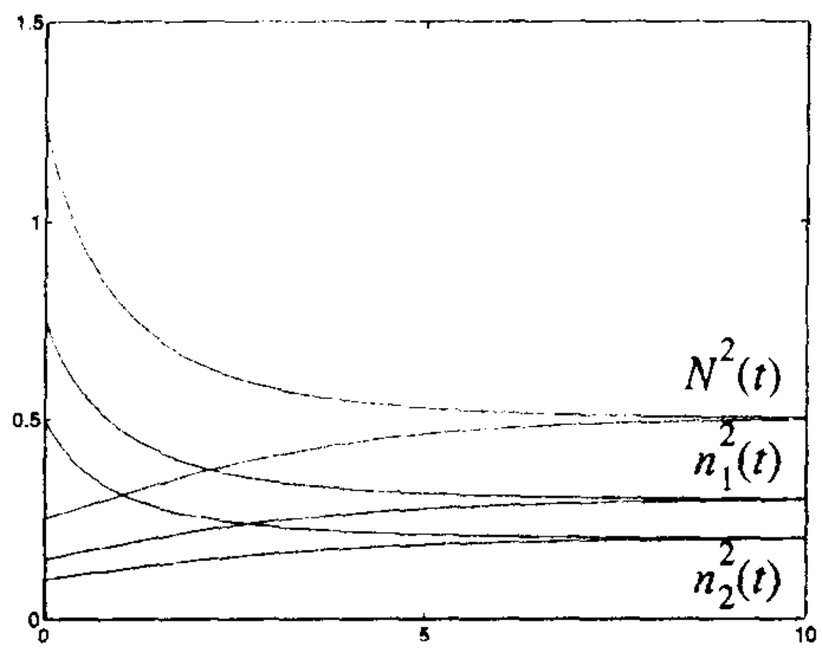



图 15-32 第二层对于线性 $f^2(n)$的响应

**P15.7** 证明由方程(15.24)给出的带衰减的连续 Hebb 规则与由等式(13.18)给出的带衰减的 Hebb 规则等价。

15-40

**解**

带衰减的连续 Hebb 规则是

$$\frac{\mathrm{d}w_{i,j}^2(t)}{\mathrm{d}t} = \alpha\{-w_{i,j}^2(t) + n_i^2(t)n_j^1(t)\}$$

如果我们估计导数为

$$\frac{\mathrm{d}w_{i,j}^2(t)}{\mathrm{d}t} \approx \frac{w_{i,j}^2(t+\Delta t) - w_{i,j}^2(t)}{\Delta t}$$

则 Hebb 规则变为

$$w_{i,j}^2(t+\Delta t) = w_{i,j}^2(t) + \alpha\Delta t\{-w_{i,j}^2(t) + n_i^2(t)n_j^1(t)\}$$

这个等式可以被重新组合得到

$$w_{i,j}^2(t+\Delta t) = [1-\alpha\Delta t]w_{i,j}^2(t) + \alpha\Delta t\{n_i^2(t)n_j^2(t)\}$$

其向量形式为

$$\mathbf{W}^2(t+\Delta t) = [1-\alpha\Delta t]\mathbf{W}^2(t) + \alpha\Delta t\{\mathbf{n}^2(t)\mathbf{n}^1(t)^T\}$$

与式(13.18)

$$\mathbf{W}(q) = (1-\gamma)\mathbf{W}(q-1) + \alpha\mathbf{a}(q)\mathbf{p}^T(q)$$

15-41 比较，可以看出它们有相同的形式。

## 15.5 结束语

本章所介绍的 Grossberg 网络是受较高等的脊椎动物的视觉系统启发。为了激励网络，我们对主要视路进行了简要描述。我们还讨论了某些视觉幻觉，帮助我们理解视觉系统的机制。

Grossberg 网络是两层连续竞争网络，在结构和运算上与第 14 章介绍的 Kohonen 竞争网络十分相似。Grossberg 网络的第一层对输入模式进行规格化。它展示视觉系统如何使用加

强中心/削弱周围的连接方式和并联模型以实现自动增益控制，从而规格化总活跃度。

Grossberg 网络的第二层实现竞争，对比增强输出模式并将其存入短期记忆。它使用非线性反馈和加强中心/削弱周围连接模式以实现竞争和存储。传输函数的选择及反馈连接方式的选择决定竞争的程度(例如胜者全得，适度的对比增强，或对模式无改变)。

Grossberg 网络使用 instar 学习规则调整权值，将原型模式以长期记忆方式存储。当第二层实现"胜者全得"的竞争时，这种学习规则就与第 14 章中介绍的 Kohonen 学习规则是等价的。

与 Kohonen 网络一样，Grossberg 网络的关键问题是学习过程的稳定性。因为更多的输入加给了网络，权值矩阵可能永远不会收敛。这个问题在第 14 章有深入讨论。在第 16 章我们将介绍一类网络设计来解决这个困难：自适应谐振论(ART)网络，ART 网络是本章介绍的 Grossberg 网络的直接后代。在本章并未讨论的 Grossberg 网络的另一个问题，是实现网络的微分方程的稳定性。例如，在第二层，我们有一个非线性反馈的微分方程组。关于这种系统的稳定性我们能够得出什么总的结论？第 17 章将提供一个对此问题的深入讨论。 15-42

## 参考文献

[GrMi89] S. Grossberg, E. Mingolla and D. Todorovic, "A neural network architecture for preattentive vision," *IEEE Transactions on Biomedical Engineering*, vol. 36, no. 1, pp. 65 - 84, 1989.

这篇文章的目的是提出一种通用预视觉的神经网络。该网络包括两个主要的子系统：边界轮廓系统和特征轮廓系统。

[Gros76] S. Grossberg, "Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors," *Biological Cybernetics*, vol. 23, pp. 121 - 134, 1976.

Grossberg 受到视觉皮层生理学的启发描述了一种连续的竞争网络。这种网络的结构形成了其他一些重要网络的基础。

[Gros82] S. Grossberg, *Studies of Mind and Brain*, Boston: D. Reidel Publishing Co., 1982.

这本书收集了 Stephen Grossberg 从 1968 年到 1980 年的文章。其中有很多重要的概念在其后的 Grossberg 网络中得到了运用，例如自适应谐振论网络。

[Hube88] D. H. Hubel, *Eye, Brain, and Vision*, New York: Scientific American Library, 1988.

David Hubel 处于这个领域的研究中心已有 30 年，他的书中对人类的视觉系统做了精彩的介绍。他解释对视觉系统的流行看法，对于任何经过一些科学训练的人来说都是容易读懂的。

[vanT75] H. F. J. M. van Tuijl, " A new visual illusion: Neonlike color spreading and complementtary color induction between subjective contours," *Acta psychologica*, vol. 39, pp. 441 - 445, 1975.

这篇文章介绍了当处于宛如固体形态的 Ehrenstein 图形时，在若干种色彩中产生幻觉的最初发现过程。 15-43

[vond73] C. von der Malsburg, "Self-organization of orientation sensitive cells in the striate cortex," *Kybernetic*, vol. 14, pp. 85 - 100, 1973.

这是第一篇关于自组织特征映射神经网络的论文。这种网络是较高级的脊椎动物视觉皮层的模型。这篇文章对后来 Kohonen 和 Grossberg 关于特征映射的工作有所影响。

15-44

## 习题

**E15.1** 考虑图 15-33 中所示的漏积分器

(i) 求 $n(t)$在 $\varepsilon=1$，$n(0)=1$，$p(t)=0.5$ 时的响应。

(ii) 求 $n(t)$在 $\varepsilon=1$，$n(0)=1$，$p(t)=2$ 时的响应。

(iii) 求 $n(t)$在 $\varepsilon=4$，$n(0)=1$，$p(t)=2$ 时的响应。

(iv) 检验对上面几部分的答案，写一个 MATLAB M-文件模拟漏积分器。使用 **ode45** 例行程序。作图表示每种情况的响应。

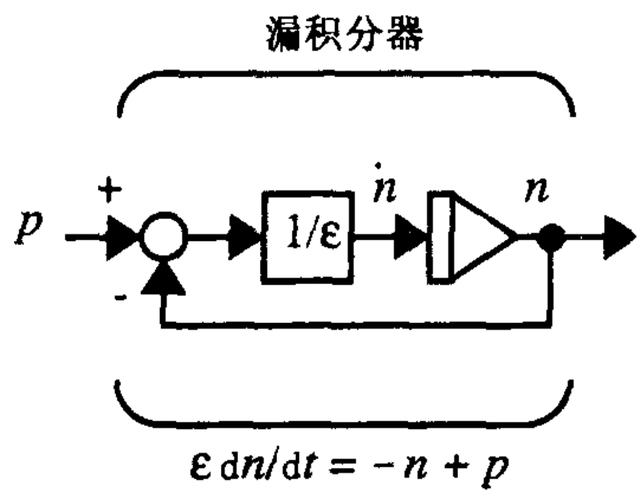



图 15-33 漏积分器

**E15.2** 考虑图 15-34 中所示的并联网络。

(i) 求并联网络在 $\varepsilon=2$，$b^+=3$，$b^-=1$，$p^+=0$，$p^-=5$，$n(0)=1$ 时的响应。

(ii) 求并联网络在 $\varepsilon=2$，$b^+=3$，$b^-=1$，$p^+=0$，$p^-=50$，$n(0)=1$ 时的响应。

(iii) 求并联网络在 $\varepsilon=2$，$b^+=3$，$b^-=1$，$p^+=50$，$p^-=0$，$n(0)=1$ 时的响应。

(iv) 写一个 MATLAB 的 M-文件以模拟并联网络，检验对上面几部分的解答，使用 **ode45** 例行程序。画出每种情况的响应图。

15-45

(v) 解释漏积分器与并联网络在运算上的区别。

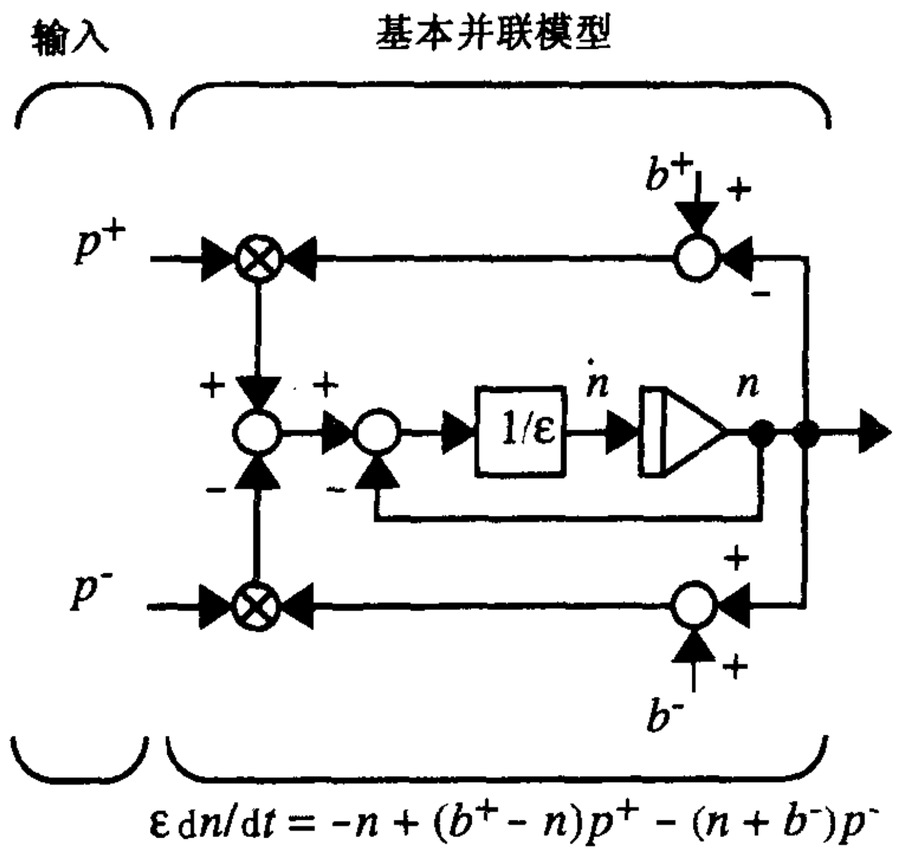



图 15-34 并联网络

**E15.3** 假设 Grossberg 网络的第一层有两个神经元，其中 ${}^{+}b^{1}=0.5$，$\varepsilon=0.5$，输入向量 $\mathbf{p}=[2\quad 1]^{T}$。假设初始状态置为 0。

（i）用等式(15.13)求出第一层的稳定状态的响应

（ii）求出第一层的微分方程的解。验证稳态响应与(i) 的答案相同。

（iii）检验你的答案，写一个 MATLAB 的 M-文件模拟 Grossberg 网络的第一层。使用 **ode45** 例行程序。画出响应图。

**E15.4** 以输入向量 $\mathbf{p}=[20\quad 10]^{T}$ 重做习题 E15.3。

**E15.5** 求出描述第一层总输出变化为

$$N^{1}(t)=\sum_{i=1}^{s^{1}}n_{i}^{1}(t)$$

的微分方程(使用例题 P15.5 中所用的技术)。 15-46

**E15.6** 假设 Grossberg 网络的第二层有 2 个神经元，其中 $f^{2}(n)=2n$，$\varepsilon=1$，${}^{+}b^{2}=1$，${}^{-}b^{2}=D$。输入已经施加了一段时间，然后撤消。

（i）稳定状态总的输出 $\lim_{t\to\infty}N^{2}(t)$ 是多少？

（ii）在 $b^{2}=0.25$ 的情况下重做(i)。

（iii）检验前两部分的答案，通过写 MATLAB M-文件模拟 Grossberg 网络第二层来进行。使用 **ode45** 例行程序。画出下列初始条件下的响应：

$$\mathbf{n}^{2}(0)=\begin{bmatrix}2\\1\end{bmatrix}\quad 和\quad \mathbf{n}^{2}(0)=\begin{bmatrix}0.2\\0.1\end{bmatrix}$$

**E15.7** 假设 Grossberg 网络的第二层的传输函数是 $f^{2}(n)=c\times(n)^{2}$，并且 $\varepsilon=1$，${}^{+}b^{2}=1$。

（i）使用例题 P15.5 的结果，证明在输入被撤消之后，所有第二层的相对输出将衰减至 0，但除了有最大初始条件(胜者全得竞争)的那个输出。

（ii）当 $c$ 为何值时总输出 $N^{2}(t)$ 将有一个非零稳定点(稳定状态值)？

（iii）如果(ii)的条件得到满足，那么 $N^{2}(t)$ 的稳态值将是多少？这依赖于初始条件 $N^{2}(0)=3$ 吗？

（iv）写一个 MATLAB M-文件并模拟在 $c=4$ 和 $N^{2}(0)=3$ 时对第二层的总响应，检验前三部分的答案。

**E15.8** 模拟 Grossberg 网络的自适应权值的响应。假设系数 $\varepsilon=1$。假设两种不同的输入模式被交替地提供给网络每次 0.2 秒。还假设与权值的收敛相比第一层和第二层的收敛极快，因而神经元输出在 0.2 秒之内实际保持不变。第二层和第一层对两种不同的输入模式的输出将是

$$对模式\ 1:\quad \mathbf{n}^{1}=\begin{bmatrix}0.8\\0.2\end{bmatrix},\quad \mathbf{n}^{2}=\begin{bmatrix}1\\0\end{bmatrix}$$

$$对模式\ 2:\quad \mathbf{n}^{1}=\begin{bmatrix}0.5\\0.5\end{bmatrix},\quad \mathbf{n}^{2}=\begin{bmatrix}0\\1\end{bmatrix}$$ 15-47

**E15.9** 用带衰减的 Hebb 规则，即方程(15.24)重做习题 E15.8，而不是用方程(15.25)的 instar 规则。解释两种响应之间的差别。 15-48

# 第16章　自适应谐振理论

## 16.1　目的

在第 14 章和第 15 章里我们知道了竞争性网络的一个关键问题是学习过程的稳定性。不能保证，当很多输入加到一个网络时，权值矩阵会最终收敛。在这一章将给出一个改进型的竞争学习，即自适应谐振理论(ART)，它可以用来克服学习过程的稳定性问题。

16-1

## 16.2　理论和实例

第 15 章提出的 Grossberg 网络和第 14 章的竞争性网络的一个关键问题是它们不能总是形成稳定聚类(或分类)。Grossberg 证明了[Gros76]如果输入模式的数量不是太大，或者这些输入模式不会形成相对于第二层神经元个数太多的聚类，那么学习过程最终是稳定的。然而，他也指出标准的竞争网络对于任意的输入模式，不会有稳定的学习过程。学习过程的不稳定性源于网络的自适应性(或可塑性)，这种自适应性导致先前的学习内容被后面的学习内容破坏掉。

**稳定性/可塑性**　Grossberg 称这个问题为“*稳定性/可塑性二难问题*”。如何能让系统只接纳重要的新模式，而在响应不相关的模式时仍保持稳定性呢？我们知道生物系统就非常擅长这一点。举个例子，哪怕你已很久没见过你的母亲，并在其间见过了许多新面孔，但你还是能很容易的认出她的脸。

Grossberg 和 Gail Carpenter 提出了一个理论，叫做自适应谐振理论(ART)，用来解决稳定性/可塑性两难问题(参见[CaGr87a]，[CaGr87b]，[CaGr90]，[CaGrRe91]和[CaGrMa92])。ART 网络建立在第 15 章的 Grossberg 网络的基础之上，其主要革新是“期望值”的使用。当每个输入模式提供给该网络时，将其与该模式最接近的匹配的原型向量(期望值)相比较。如果该模式向量与原型向量不足以匹配，那么它将作为一个新的原型向量而被选中。通过这种方式，先前学习的记忆内容(原型)就不会被新的学习内容所破坏。

讨论所有的自适应谐振理论的变型超出了本章的范围，但我们将详细讨论一种 ART 网络——ART1(参见[CaGr87a])。这种特别的网络仅为二值输入向量而设计。但是，我们可以从这个体系结构里，了解自适应谐振理论的主要特征。

### 16.2.1　自适应谐振概述

基本的 ART 体系结构如图 16-1 所示。它是第 15 章 Grossberg 网络的一个变型(与图 15-16 比较)，被用来稳定学习过程。ART 体系结构的改进包括三个部分：第二层(L2)到第一层(L1)的期望值、调整子系统和增益控制。在这一节里，我们将描述 ART 系统的一般操作；在以后的几小节里，我们将详细讨论每一个子系统。

16-2

回顾第 15 章我们知道，Grossberg 网络的 L1 – L2 连接为 instar 形态，用来执行聚类(或分类)操作。当一个输入模式被提交给网络时，它(经过规格化后)将与 L1 – L2 权值矩阵相乘。然后，在第二层就会通过竞争决定权值矩阵的哪一列最接近输入向量，这一列即被移向

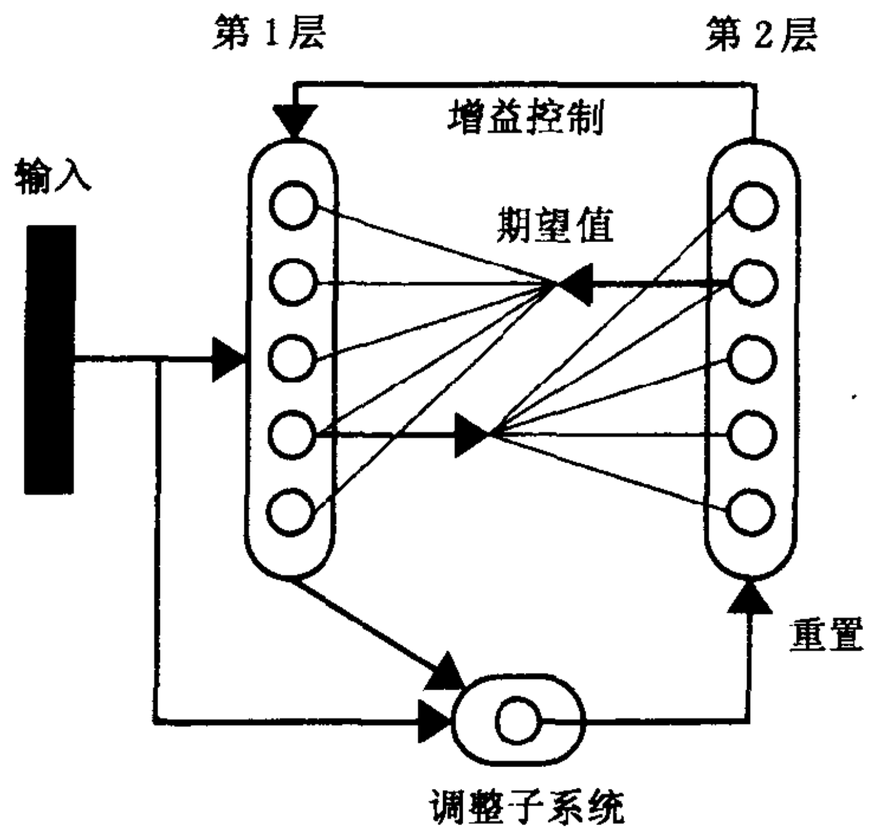



图 16-1　基本的 ART 体系结构

输入向量。在学习结束后，L1 - L2 权值矩阵的每一列都是一个原型模式，它将代表输入向量的一个聚类(或分类)。

在 ART 网络里，学习也发生在从第二层到第一层的一系列反馈连接中。这些连接是 outstar 形态(参见第 13 章)，用来进行模式回忆。当第二层的一个节点被激活时，它将在第一层对应一个原型模式(期望值)。第一层接着将期望值与输入模式进行比较。

当期望值与输入模式不能进行密切匹配时，调整子系统将重置第二层。这种重置将取缔当前的优胜神经元，同时取消当前的期望值。当上次的优胜神经元被取缔之后，第二层里将进行一次新的竞争。第二层里的新的优胜神经元又通过 L2 - L1 连接向第一层产生一个期望值。这个过程会持续到 L2 - L1 期望值与输入模式足够密切地匹配时才结束。

在下面几小结里，我们将分析 ART 系统的每一个系统 ——这些子系统应用到一个特殊的 ART 网络 ART1([CaGr87a])。我们会首先描述反映这些子系统操作的微分方程，然后导出每个子系统稳态响应。最后，总结 ART1 系统的所有操作。 16-3

### 16.2.2　第一层

第一层的主要用途是比较输入模式和来自第二层的期望值模式。(在 ART1 里，两种模式都是二值的。)如果模式不能密切匹配，那么调整子系统会重置第二层。如果模式能足够密切地匹配，第一层将结合期望值和输入形成一个新的原型模式。

ART1 网络的第一层如图 16-2 所示，它非常近似于 Grossberg 网络的第一层(参见图 15-17)。不同之处在于对并联模型的激励输入和抑制输入。对于 ART1 网络，第一层里不执行规格化过程，所以我们不能从输入向量中得到“加强中心/抑制周围”(on-center/off-surround)的连接。ART1 第一层的激励输入由输入模式和 L1 - L2 期望值结合构成。抑制输入则由来自第二层的增益控制信号构成。下面我们将解释这些输入怎样在一起工作。

第一层的运算方程为

$$\varepsilon \frac{d\mathbf{n}^1(t)}{dt} = -\mathbf{n}^1(t) + ({}^{+}\mathbf{b}^1 - \mathbf{n}^1(t))\{\mathbf{p} + \mathbf{W}^{2:1}\mathbf{a}^2(\mathbf{t})\} - (\mathbf{n}^1(t) + {}^{-}\mathbf{b}^1)[{}^{-}\mathbf{W}^1]\mathbf{a}^2(t) \tag{16.1}$$

16-4

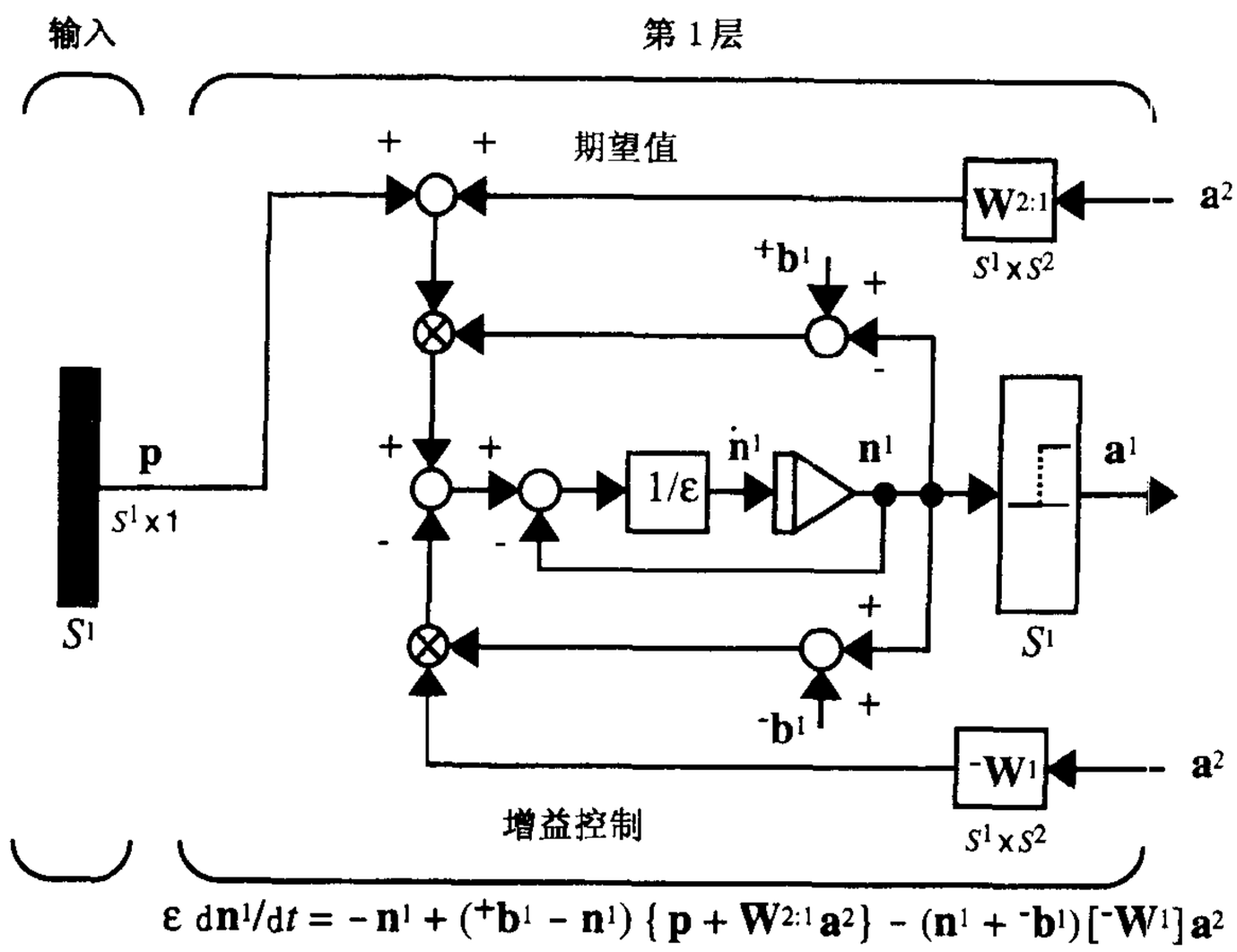



图 16-2 ART1 网络的第一层

而第一层的输出计算为

$$\mathbf{a}^1 = \mathbf{hardlim}^+(\mathbf{n}^1) \tag{16.2}$$

其中

$$hardlim^+(n) = \begin{cases} 1, & n > 0 \\ 0, & n \leqslant 0 \end{cases} \tag{16.3}$$

方程(16.1)是并联模型，拥有激励输入 $\mathbf{p} + \mathbf{W}^{2:1}\mathbf{a}^2(t)$，它是输入向量与 L2 - L1 期望值的和。例如，假设第二层的第 $j$ 个神经元在竞争中获胜，那么它的输出是 1，而其他神经元的输出是 0。由此，我们得到

$$\mathbf{W}^{2:1}\mathbf{a}^2 = \left[\mathbf{w}_1^{2:1}\ \mathbf{w}_2^{2:1}\ \cdots\ \mathbf{w}_j^{2:1}\ \cdots\ \mathbf{w}_{S^2}^{2:1}\right]\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \end{bmatrix} = \mathbf{w}_j^{2:1} \tag{16.4}$$

其中 $\mathbf{w}_j^{2:1}$ 是矩阵 $\mathbf{W}^{2:1}$的第 $j$ 列。(矩阵 $\mathbf{W}^{2:1}$利用 outstar 规则训练，这会在后面的小节里介绍。)现在我们可以看到

$$\mathbf{p} + \mathbf{W}^{2:1}\mathbf{a}^2 = \mathbf{p} + \mathbf{w}_j^{2:1} \tag{16.5}$$

因此对第一层的激励输入是输入模式与 L2 - L1 期望值的和。L2 - L1 矩阵的每一列代表了一个不同的期望值(原型模式)。以后我们会看到，第一层利用 AND 操作将输入模式与期望值结合起来。

对第一层的抑制输入即是增益控制项$[{}^-\mathbf{W}^1]\mathbf{a}^2(t)$，其中

$${}^-\mathbf{W}^1 = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{16.6}$$

可见，对第一层每个神经元的抑制输入即为第二层所有输出的总和。既然我们在第 2 层采用了“胜者全得”(winner - take - all)的竞争方法，那么每当第二层处于活动状态时，竞争后就有一个而且只有一个 $\mathbf{a}^2$ 的非零元素。因此，对第一层的增益控制输入，当第二层活跃时为 1，当第二层不活跃(所有神经元的输出都为 0)时为 0。增益控制的目的在我们分析第一层的稳态行为时将会很明显。 16-5

稳定状态分析

第一层中神经元 $i$ 的响应可描述为

$$\varepsilon \frac{\mathrm{d}n_i^1}{\mathrm{d}t} = -n_i^1 + ({}^+b^1 - n_i^1)\left\{p_i + \sum_{j=1}^{S^2} w_{i,j}^{2:1} a_j^2\right\} - (n_i^1 + {}^-b^1)\sum_{j=1}^{S^2} a_j^2 \tag{16.7}$$

其中 $\varepsilon \ll 1$，所以短期记忆轨迹(short-term memory trace)(神经元的输出)的改变比长期记忆轨迹(long-term memory trace)(权值矩阵)的改变快得多。

我们想检查这个系统在两种情况下的稳态响应。第一种情况是第二层不活跃，因此对所有的 $j$ 有 $a_j^2 = 0$。第二种情况里第二层是活跃的，因此有一个神经元的输出为 1，所有其他神经元的输出为 0。

考虑第二层不活跃的第一种情况。因为所有 $a_j^2 = 0$，所以方程(16.7)简化为

$$\varepsilon \frac{\mathrm{d}n_i^1}{\mathrm{d}t} = -n_i^1 + ({}^+b^1 - n_i^1)\{p_i\} \tag{16.8}$$

在稳定状态($\mathrm{d}n_i^1(t)/\mathrm{d}t = 0$)时，有

$$0 = -n_i^1 + ({}^+b^1 - n_i^1)p_i = -(1 + p_i)n_i^1 + {}^+b^1 p_i \tag{16.9}$$

如果求解稳定状态时神经元输出 $n_i^1$，我们得到

$$n_i^1 = \frac{{}^+b^1 p_i}{1 + p_i} \tag{16.10}$$

因此，如果 $p_i = 0$，则 $n_i^1 = 0$；如果 $p_i = 1$，则 $n_i^1 = {}^+b^1/2 > 0$。由于我们选择第一层的转移函数为函数 $hardlim^+$，故得

$$\mathbf{a}^1 = \mathbf{p} \tag{16.11}$$

所以，当第二层不活跃时，第一层的输出与输入模式相同。

现在我们来考虑第二层活跃的第二种情况。假如神经元 $j$ 是第二层的优胜神经元。那么 $a_j^2 = 1$ 且 $a_k^2 = 0\ (k \neq j)$。此时方程(16.7)简化为 16-6

$$\varepsilon \frac{\mathrm{d}n_i^1}{\mathrm{d}t} = -n_i^1 + ({}^+b^1 - n_i^1)\{p_i + w_{i,j}^{2:1}\} - (n_i^1 + {}^-b^1) \tag{16.12}$$

在稳定状态($\mathrm{d}n_i^1(t)/\mathrm{d}t = 0$)时，有

$$\begin{aligned} 0 &= -n_i^1 + ({}^+b^1 - n_i^1)\{p_i + w_{i,j}^{2:1}\} - (n_i^1 + {}^-b^1) \\ &= -(1 + p_i + w_{i,j}^{2:1} + 1)n_i^1 + ({}^+b^1(p_i + w_{i,j}^{2:1}) - {}^-b^1) \end{aligned} \tag{16.13}$$

求解稳定状态时神经元输出 $n_i^1$，我们得到

$$n_i^1 = \frac{{}^+b^1(p_i + w_{i,j}^{2:1}) - {}^-b^1}{2 + p_i + w_{i,j}^{2:1}} \tag{16.14}$$

回忆第一层应该结合输入向量和来自第二层的期望值(表示为 $\mathbf{w}_j^{2:1}$)。由于我们处理的是

二值模式(无论是输入还是期望值)，我们可以利用逻辑 AND(与)运算结合这两个向量。换句话说，我们希望 $n_i^1$ 在 $p_i$ 或 $w_{i,j}^{2:1}$ 中有一个为 0 时小于 0，$n_i^1$ 在 $p_i$ 和 $w_{i,j}^{2:1}$ 都等于 1 时大于 0。把这个条件用于等式(16.14)中，我们得到如下式子：

$$^{+}b^1(2) - {}^{-}b^1 > 0 \tag{16.15}$$

$$^{+}b^1 - {}^{-}b^1 < 0 \tag{16.16}$$

合在一起即为

$$^{+}b^1(2) > {}^{-}b^1 > {}^{+}b^1 \tag{6.17}$$

例如，我们令 $^{+}b^1 = 1$ 且 $^{-}b^1 = 1.5$，即满足上述条件。

因此，如果式子(16.17)得到满足，并且第二层神经元 $j$ 处于活跃状态，那么第一层的输出为

$$\mathbf{a}^1 = \mathbf{p} \cap \mathbf{w}_j^{2:1} \tag{16.18}$$

其中∩代表逻辑与运算。

16-7 注意，我们需要增益控制来实现 AND 运算。考虑等式(16.14)的分子：

$$^{+}b^1(p_i + w_{i,j}^{2:1}) - {}^{-}b^1 \tag{16.19}$$

其中项 $^{-}b^1$ 与增益控制项相乘，该项这里为 1。如果整个这一项不存在，那么式子(16.19)将会大于 0(因此 $n_i^1$ 将大于 0)，而不管 $p_i$ 或 $w_{i,j}^{2:1}$ 是否大于 0。这就成了 OR(或)运算，而不是 AND(与)运算。我们在讨论调整子系统时将会看到，第一层执行 AND 运算将是关键。

当第二层不活跃时，增益控制项为 0。这是必须的，因为我们希望在第二层没有期望值激活的这种情况下，第一层仅对输入模式作出响应。

小结第一层稳定状态运算：

若第二层不活跃(即所有 $a_j^2 = 0$)，

$$\mathbf{a}^1 = \mathbf{p} \tag{16.20}$$

若第二层活跃(即有一个 $a_j^2 = 1$)，

$$\mathbf{a}^1 = \mathbf{p} \cap \mathbf{w}_j^{2:1} \tag{16.21}$$

为了说明第一层的运算，假设网络参数如下：

$$\varepsilon = 0.1, {}^{+}b^1 = 1, {}^{-}b^1 = 1.5 \tag{16.22}$$

再假设第二层里有两个神经元，输入向量有两个元素并且有如下权值矩阵和输入：

$$\mathbf{W}^{2:1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \quad 和 \quad \mathbf{p} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{16.23}$$

如果我们采用第二层是活跃的这种情况，且第二层第二个神经元赢得了竞争，则第一层的运算方程为

16-8

$$\begin{aligned}(0.1)\frac{dn_1^1}{dt} &= -n_1^1 + (1 - n_1^1)\{p_1 + w_{1,2}^{2:1}\} - (n_1^1 + 1.5) \\ &= -n_1^1 + (1 - n_1^1)\{0 + 1\} - (n_1^1 + 1.5) = -3n_1^1 - 0.5\end{aligned} \tag{16.24}$$

$$\begin{aligned}(0.1)\frac{dn_2^1}{dt} &= -n_2^1 + (1 - n_2^1)\{p_2 + w_{2,2}^{2:1}\} - (n_2^1 + 1.5) \\ &= -n_2^1 + (1 - n_2^1)\{1 + 1\} - (n_2^1 + 1.5) = -4n_2^1 + 0.5\end{aligned} \tag{16.25}$$

化简得

$$\frac{dn_1^1}{dt} = -30n_1^1 - 5 \tag{16.26}$$

$$\frac{dn_2^1}{dt} = -40n_2^1 + 5 \tag{16.27}$$

在这个简单的例子中我们可以求出这两个方程相近形式的解。如果我们假设两个神经元都从零初值开始，那么结果是

$$n_1^1(t) = -\frac{1}{6}[1 - e^{-30t}] \tag{16.28}$$

$$n_2^1(t) = \frac{1}{8}[1 - e^{-40t}] \tag{16.29}$$

它们的图形见图 16-3

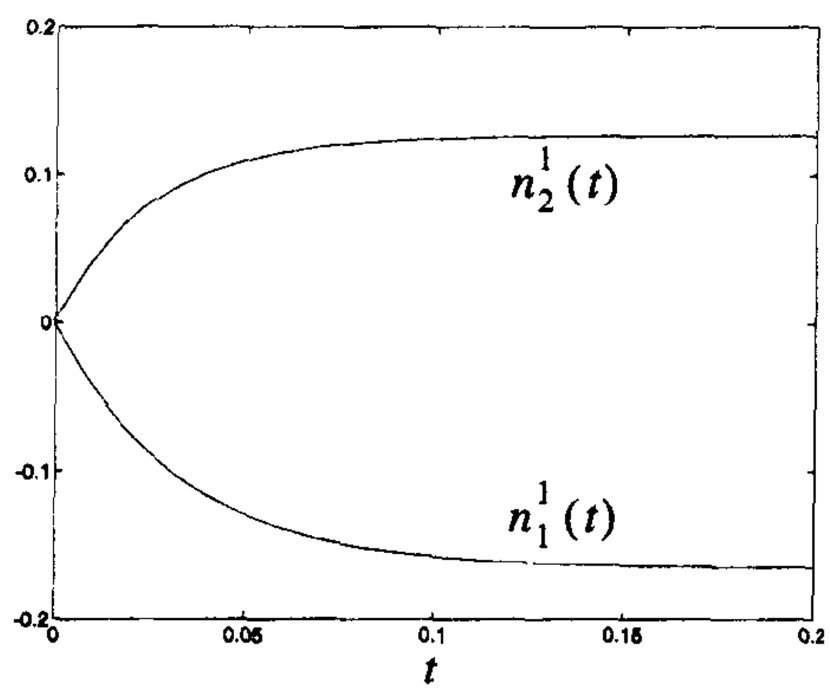
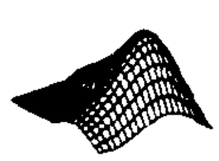



图 16-3　第一层的响应

注意，$n_1^1(t)$收敛于一个负值，而 $n_2^1(t)$收敛于一个正值。因此，$a_1^1(t)$收敛于 0，$a_2^1(t)$收敛于 1(回忆第一层的转移函数为 *hardlim*$^+$)。这与我们的稳态分析一致(参见等式(16.21))，因为 16-9

$$\mathbf{p} \cap \mathbf{w}_2^{2:1} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cap \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \mathbf{a}^1 \tag{16.30}$$



试验 ART1 网络的第一层请用 *Neural Network Design Demonstration ART1 Layer 1* (**nnd16al1**)。

### 16.2.3　第二层

ART1 网络的第二层与第 15 章的 Grossberg 网络第二层几乎相同。它的主要目的在于对比增强它的输出模式。对于我的 ART1 网络的实现，对比增强将是“胜者全得”的竞争方式，所以只有接受到最大输入的神经元才会有非零输出。

Grossberg 和 ART1 网络的第二层之间有一个主要的差别。ART1 的第二层利用了一个可被重置的积分器。在这种积分器之中，如图 16-4 所示，每当 $a^0$ 信号变为正值的时候，任何正的输出都会被重置为 0。这种被重置的输出将保持抑制一段很长的时间，以致它们不会被驱动到 0

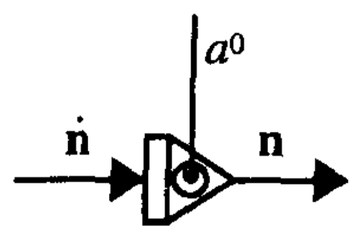



图 16-4

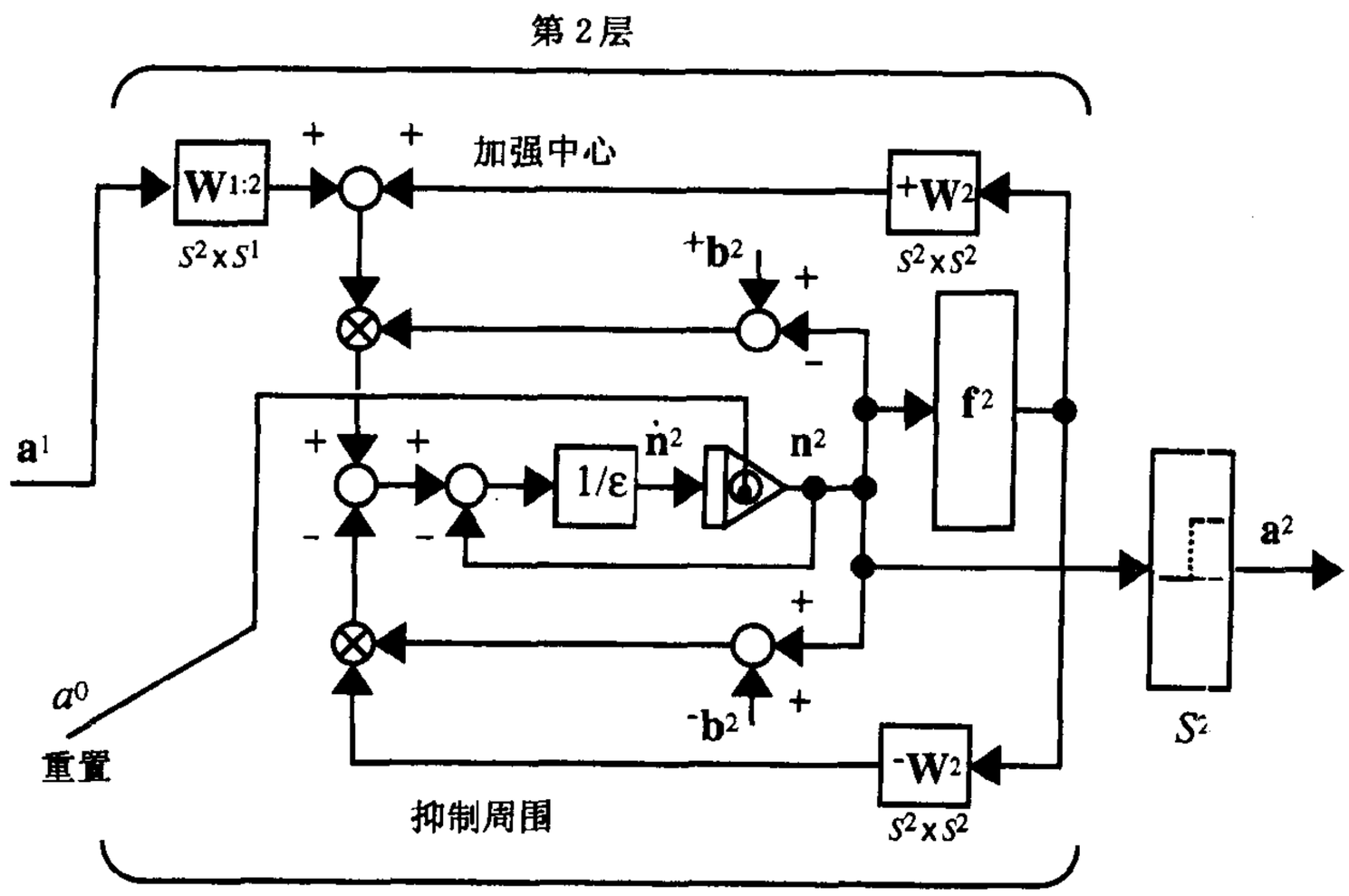
之上。(一段“长”时间指的是直到充分的匹配发生并且权值已被更新。)

在最初的 ART1 论文中，Carpenter 和 Grossberg 提出重置机构用一个门偶极子场来实现[CaGr87]。他们后来提出了一个更为复杂的生物学模型，在他们的 ART3 体系结构中使用了化学的神经传送器[CaGr90]。就这里的目标来说，我们不会涉及专门的生物学实现方法。

图 16-5 完整显示了 ART1 网络的第二层。再次指出，它与第 15 章的 Grossberg 网络第二层几乎相同(参见图 15-20)，除了那个可重置的积分器以外。重置信号 $a^0$ 是调整子系统的输出，这个我们将在后面的小节里讨论。无论何时第一层的输入信号与 L2 - L1 期望值发生不匹配，它都将导致重置。

ART1 网络第二层与 Grossberg 网络第二层的另一个小区别是 ART1 中用到了两个传输函数。传输函数 $\mathbf{f}^2(\mathbf{n}^2)$用于“加强中心/抑制周围”式反馈连接，此时第二层的输出被计算为 $\mathbf{a}^2 = \mathbf{hardlim}^+(\mathbf{n}^2)$。第二个传输函数的使用是因为我们希望第二层的输出信号是一个二值信号。

16-10




$$\varepsilon\, d\mathbf{n}^2/dt = -\mathbf{n}^2 + ({}^+\mathbf{b}^2 - \mathbf{n}^2)\{[{}^+\mathbf{W}^2]\mathbf{f}^2(\mathbf{n}^2) + \mathbf{W}^{1:2}\mathbf{a}^1\} - (\mathbf{n}^2 + {}^-\mathbf{b}^2)[{}^-\mathbf{W}^2]\mathbf{f}^2(\mathbf{n}^2)$$

图 16-5 ART1 网络的第二层

第二层的运算方程为

$$\varepsilon \frac{d\mathbf{n}^2(t)}{dt} = -\mathbf{n}^2(t) + ({}^+\mathbf{b}^2 - \mathbf{n}^2(t))\{[{}^+\mathbf{W}^2]\mathbf{f}^2(\mathbf{n}^2(t)) + \mathbf{W}^{1:2}\mathbf{a}^1\} - (\mathbf{n}^2(t) + {}^-\mathbf{b}^2)[{}^-\mathbf{W}^2]\mathbf{f}^2(\mathbf{n}^2(t)) \tag{16.31}$$

这是一个并联模型，具有激励输入$\{[{}^+\mathbf{W}^2]\mathbf{f}^2(\mathbf{n}^2(t)) + \mathbf{W}^{1:2}\mathbf{a}^1\}$，其中 ${}^+\mathbf{W}^2$ 提供了“加强中心”的反馈连接(与第 15 章的 Grossberg 网络第一层和第二层相同，参见方程(15.6))，$\mathbf{W}^{1:2}$ 由自适应权值构成，类似于 Kohonen 网络的权值。它们按照 instar 规则训练，这在以后的一小节中将会看到。在训练后，$\mathbf{W}^{1:2}$的各行将代表各个原型模式。

并联模型的抑制输入是$[{}^-\mathbf{W}^2]\mathbf{f}^2(\mathbf{n}^2(t))$，其中 ${}^-\mathbf{W}^2$ 提供“抑制周围”的反馈连接(与 Grossberg 网络的第一、二层相同，见方程(15.7))。为了演示第二层的执行过程，考虑拥有两个神经元的一层，具有

16-11

$$\varepsilon = 0.1, {}^{+}\mathbf{b}^2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, {}^{-}\mathbf{b}^2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{W}^{1:2} = \begin{bmatrix} ({}_1\mathbf{w}^{1:2})^T \\ ({}_2\mathbf{w}^{1:2})^T \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 \\ 1 & 0 \end{bmatrix} \tag{16.32}$$

并且

$$f^2(n) = \begin{cases} 10(n)^2, & n \geqslant 0 \\ 0, & n < 0 \end{cases} \tag{16.33}$$

该层的运算方程为

$$\begin{aligned}(0.1)\frac{\mathrm{d}n_1^2(t)}{\mathrm{d}t} &= -n_1^2(t) + (1 - n_1^2(t))\{f^2(n_1^2(t)) + ({}_1\mathbf{w}^{1:2})^T\mathbf{a}^1\} \\ &\quad - (n_1^2(t) + 1)f^2(n_2^2(t))\end{aligned} \tag{16.34}$$

$$\begin{aligned}(0.1)\frac{\mathrm{d}n_2^2(t)}{\mathrm{d}t} &= -n_2^2(t) + (1 - n_2^2(t))\{f^2(n_2^2(t)) + ({}_2\mathbf{w}^{1:2})^T\mathbf{a}^1\} \\ &\quad - (n_2^2(t) + 1)f^2(n_1^2(t))\end{aligned} \tag{16.35}$$

它们在形式上第 15 章 Grossberg 第 2 层的例子(参见方程(15.20)和方程(15.21))相同，除了 ${}^{-}b^2 = 1$ 外。这允许 $n_1^2(t)$和 $n_2^2(t)$的范围在 $-1$ 与 $+1$ 之间。

第二层的输入是原型模式(权值矩阵 $\mathbf{W}^{1:2}$的各行)与第一层输出的内积。(这个权值矩阵的各行已经规格化，在以后的一小节中会有解释。)最大的内积与最接近第一层输出的原型模式相对应。随后在第二层里发生神经元竞争。传输函数 $f^2(n)$被选作为一个“快于线性”的传输函数(参见第 15 章 15.2.3 节中“传输函数的选择”对于 $f^2(n)$的影响的讨论)。这个选择强迫拥有最大输入的神经元具有正的 $n$ 值，而其他神经元具有负的 $n$ 值(适当选择网络参数)。竞争结束后，由于采用了传输函数 *hardlim*$^+$ 计算层的输出，故有一个神经元的输出为 1，而其他的神经元输出都为 0。

图 16-6 显示当输入向量为 $\mathbf{a}^1 = [1 \quad 0]^T$ 时第二层的响应。$\mathbf{W}^{1:2}$的第二行由于 $\mathbf{a}^1$ 而具有较之第一行更大的内积，所以神经元 2 赢得了竞争。在稳定状态下，$n_2^2(t)$具有一个正的值，而 $n_1^2(t)$具有一个负的值。稳定状态下第二层的输出因而是 16-12

$$\mathbf{a}^2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{16.36}$$

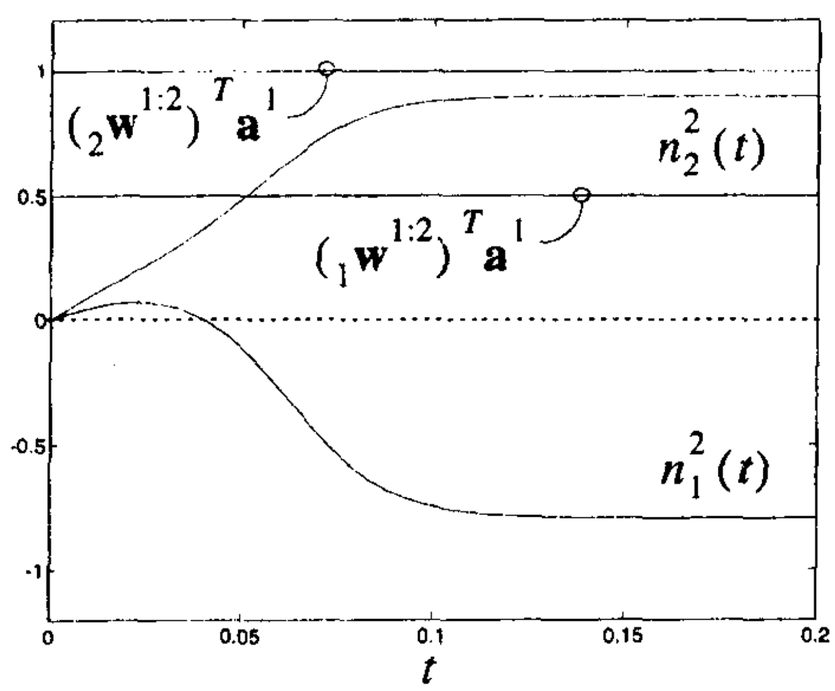



图 16-6　第二层的响应

我们可以小结第二层的稳定状态运算如下：

$$a_i^2 = \begin{cases} 1, & \text{如果}((_i\mathbf{W}^{1:2})^T\mathbf{a}^1 = max[(_j\mathbf{w}^{1:2})^T\mathbf{a}^1]) \\ 0, & \text{其他} \end{cases} \tag{16.37}$$

试验第二层 ART1 网络请用 *Neural Network Design Demonstration* ***ART1 Layer 2*** (**nnd16a12**)。

### 16.2.4 调整子系统

ART 体系结构的一个关键元素是"调整子系统"。它的作用是判定 L2 - L1 期望值与输入模式之间是否充分匹配。当不充分匹配时，调整子系统会向第二层发出一个重置信号。重置信号将导致前一个获胜神经元长时期抑制，从而使另一个神经元在竞争中获胜。

16-13 图 16-7 显示了调整系统。

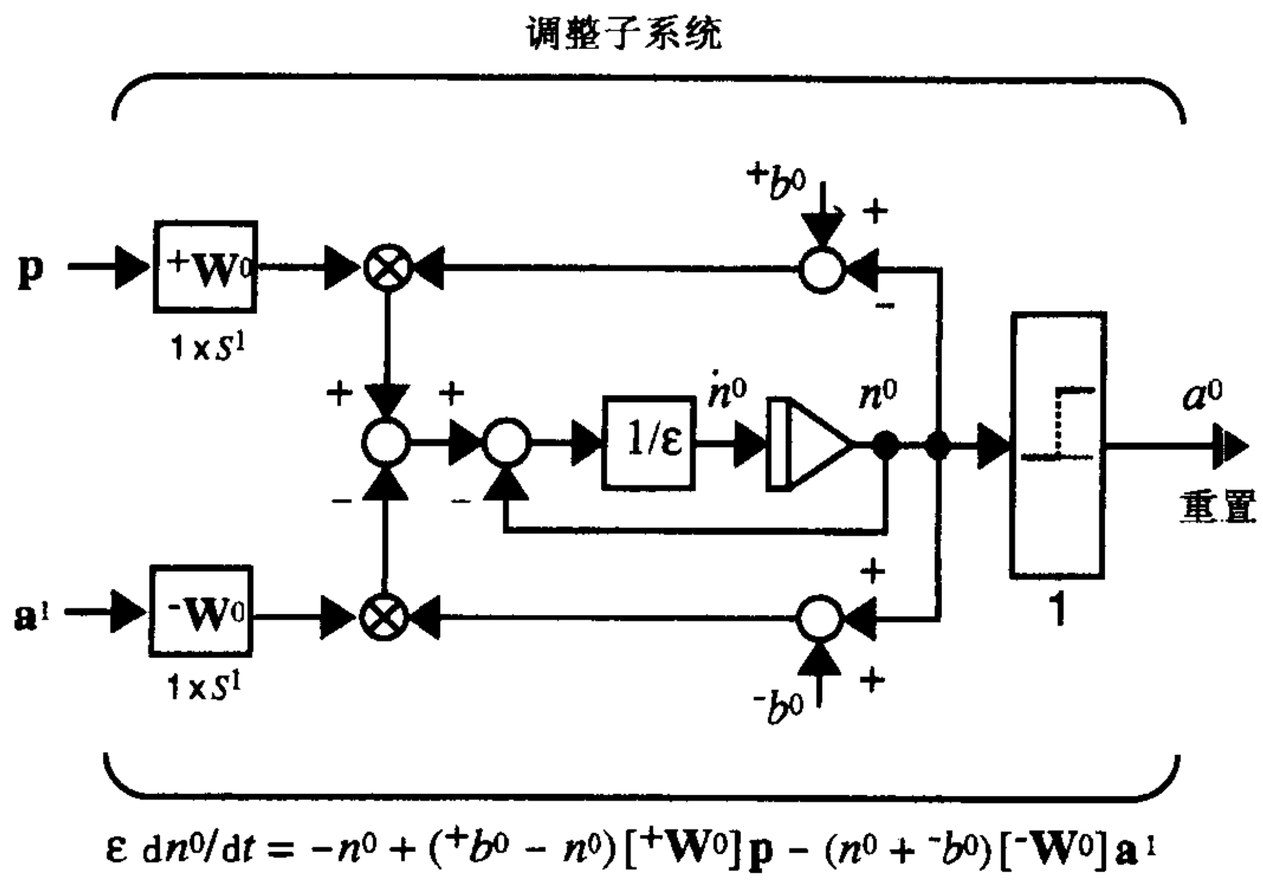



图 16-7 ART1 网络的调整子系统

调整子系统的运算方程为

$$\varepsilon\frac{\mathrm{d}n^0(t)}{\mathrm{d}t} = -n^0(t) + ({}^+b^0 - n^0(t))\{{}^+\mathbf{W}^0\mathbf{p}\} - (n^0(t) + {}^-b^0)\{{}^-\mathbf{W}^0\mathbf{a}^1\} \tag{16.38}$$

这是一个并联模型，具有激励输入 ${}^+\mathbf{W}^0\mathbf{p}$，其中

$${}^+\mathbf{W}^0 = [\alpha \quad \alpha \quad \cdots \quad \alpha] \tag{16.39}$$

因此，此激励输入可写成

$${}^+\mathbf{W}^0\mathbf{p} = [\alpha \quad \alpha \quad \cdots \quad \alpha]\mathbf{p} = \alpha\sum_{j=1}^{S^1} p_j = \alpha\|\mathbf{p}\|^2 \tag{16.40}$$

其中最后一个等式成立是因为 $\mathbf{p}$ 是一个二值向量。

调整子系统的抑制输入是 ${}^-\mathbf{W}^0\mathbf{a}^1$，其中

$${}^-\mathbf{W}^0 = [\beta \quad \beta \quad \cdots \quad \beta] \tag{16.41}$$

因此，抑制输入可以写成

16-14

$${}^-\mathbf{W}^0\mathbf{a}^1 = [\beta \quad \beta \quad \cdots \quad \beta]\mathbf{a}^1 = \beta\sum_{j=1}^{S^1} a_j^1(t) = \beta\|\mathbf{a}^1\|^2 \tag{16.42}$$

一旦激励输入大过抑制输入，调整子系统就会被驱动。考虑下面的稳态运算：

$$\begin{aligned}0 &= -n^0 + ({}^+b^0 - n^0)\{\alpha\|\mathbf{p}\|^2\} - (n^0 + {}^-b^0)\{\beta\|\mathbf{a}^1\|^2\} \\ &= -(1 + \alpha\|\mathbf{p}\|^2 + \beta\|\mathbf{a}^1\|^2)n^0 + {}^+b^0(\alpha\|\mathbf{p}\|^2) - {}^-b^0(\beta\|\mathbf{a}^1\|^2)\end{aligned} \tag{16.43}$$

若求解 $n^0$，得到

$$n^0 = \frac{{}^+b^0(\alpha\|\mathbf{p}\|^2) - {}^-b^0(\beta\|\mathbf{a}^1\|^2)}{(1 + \alpha\|\mathbf{p}\|^2 + \beta\|\mathbf{a}^1\|^2)} \tag{16.44}$$

令 ${}^+b^0 = {}^-b^0 = 1$，则当 $\alpha\|\mathbf{p}\|^2 - \beta\|\mathbf{a}^1\|^2 > 0$ 时，$n^0 > 0$，也即

$$n^0 > 0, \quad 若\frac{\|\mathbf{a}^1\|^2}{\|\mathbf{p}\|^2} < \frac{\alpha}{\beta} = \rho \tag{16.45}$$

**警戒**　由于 $a^0 = hardlim^+(n^0)$，故上式即为导致第二层重置的条件。项 $\rho$ 被称为警戒参数，必须落在范围 $0<\rho<1$ 内。如果警戒值接近 1，那么除非 $\mathbf{a}^1$ 接近于 $\mathbf{p}$，否则将引起重置。如果警戒值接近 0，$\mathbf{a}^1$ 不接近 $\mathbf{p}$ 也能防止重置。警戒参数决定了由原型向量创建的分类(或聚类)的粗略情况。

回忆等式(16.21)，无论何时第二层处于活跃状态，都有 $\mathbf{a}^1 = \mathbf{p} \cap \mathbf{w}_j^{2:1}$。因此总有 $\|\mathbf{p}\|^2$ 大于等于 $\|\mathbf{a}^1\|^2$。当每个输入 $\mathbf{p}$ 有 1 的地方期望值 $\mathbf{w}_j^{2:1}$ 也有 1 时，它们两个相等。因此，当 $\mathbf{p}$ 和 $\mathbf{w}_j^{2:1}$ 的不匹配足够显著时，调整子系统会导致重置的发生。发生重置所需要的不匹配程度由警戒参数 $\rho$ 决定。

为了演示调整子系统的运算过程，假设 $\varepsilon=0.1$，$\alpha=3$，$\beta=4$ ($\rho=0.75$)，

$$\mathbf{p} = \begin{bmatrix}1\\1\end{bmatrix} \quad 且 \quad \mathbf{a}^1 = \begin{bmatrix}1\\0\end{bmatrix} \tag{16.46}$$ 16-15

则运算方程变为

$$\begin{aligned}(0.1)\frac{dn^0(t)}{dt} &= -n^0(t) + (1 - n^0(t))\{3(p_1 + p_2)\} \\ &\quad - (n^0(t) + 1)\{4(a_1^1 + a_2^1)\}\end{aligned} \tag{16.47}$$

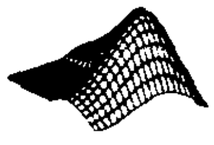
或

$$\frac{dn^0(t)}{dt} = -110n^0(t) + 20 \tag{16.48}$$

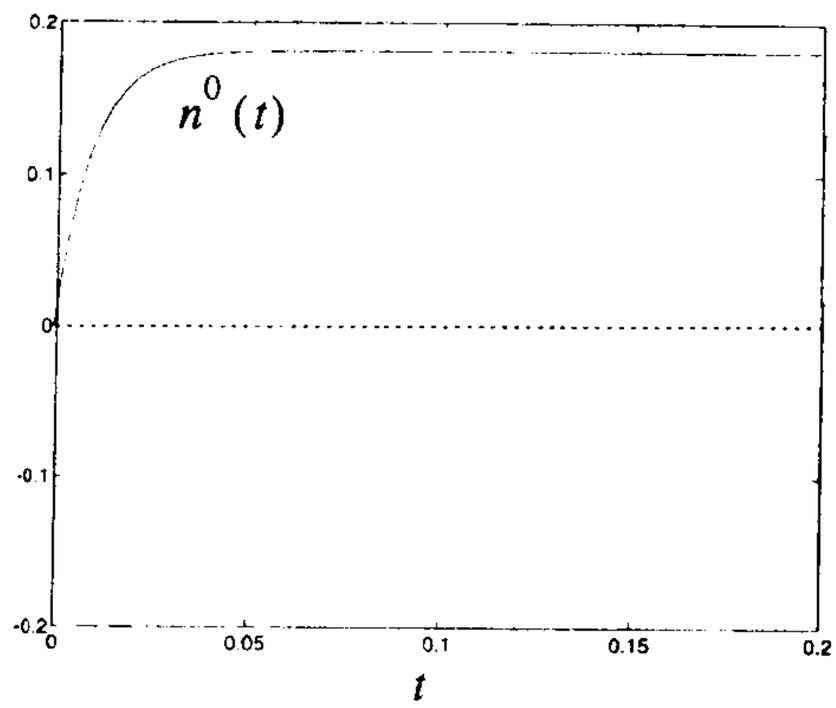
图 16-8 画出了响应过程。此例中，由于 $n^0(t)$ 为正，所以一个重置信号将被发向第二层。进一步，因为警戒参数 $\rho=0.75$，而且 $\mathbf{p}$ 只有两个元素，所以无论什么时候 $\mathbf{p}$ 和 $\mathbf{a}^1$ 不相等，都会发生重置。(如果警戒参数被设成 $\rho=0.25$，由于 $\|\mathbf{a}^1\|^2/\|\mathbf{p}\|^2 = 1/2$，故对于等式(16.46)中的 $\mathbf{p}$ 和 $\mathbf{a}^1$，将不会发生重置。)

对稳定状态下调整子系统的运算小结如下：

$$a^0 = \begin{cases}1, & 当[\|\mathbf{a}^1\|^2/\|\mathbf{p}\|^2 < \rho]时 \\ 0, & 其他\end{cases} \tag{16.49}$$



试验调整子系统请用 *Neural Network Design Demonstration Orienting Subsystem* (**nnd16os**)。




16-16

图 16-8 调整子系统的响应

### 16.2.5 学习规则：L1 - L2

ART1 网络有两个分别的学习规则：一个用于 L1 - L2 连接，一个用于 L2 - L1 连接。L1 - L2 连接使用一种 instar 学习过程学习识别一系列原型模式。L2 - L1 连接收使用一种 outstar 学习过程重演(或回忆)一系列原型模式。这一节里，我们将讨论 L1 - L2 instar 学习规则，下一节将讨论 L2 - L1 outstar 学习规则。

**谐振** 我们应该注意到 L1 - L2 连接与 L2 - L1 连接是同时更新的。每当输入模式和期望值发生了适当的匹配，在调整子系统的控制下，$\mathbf{W}^{1:2}$和 $\mathbf{W}^{2:1}$都会被更新。这个匹配过程，以及随后的适应过程，被称为谐振，自适应谐振理论由此而得名。

1. 子集/超集二难问题

除了一个主要区别外，ART1 网络 L1 - L2 连接的学习与 15 章的 Grossberg 网络学习十分接近。在 Grossberg 网络中，输入模式在第一层会被规格化，因此所有的原型模式都有相等的长度。在 ART1 网络的第一层中并没有规格化过程。因此当某个原型模式是另一个原型模式的子集时就会出现问题。例如，假设 L1 - L2 连接矩阵为

$$\mathbf{W}^{1:2}\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{16.50}$$

那么其原型模式为

$$_1\mathbf{w}^{1:2} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad 且 \quad _2\mathbf{w}^{1:2} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{16.51}$$

因为，$_1\mathbf{w}^{1:2}$中有 1 的地方$_2\mathbf{w}^{1:2}$中也有 1，我们就认为$_1\mathbf{w}^{1:2}$是$_2\mathbf{w}^{1:2}$的子集。

如果第一层的输出为

16-17

$$\mathbf{a}^1 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \tag{16.52}$$

那么第二层的输入为

$$\mathbf{W}^{1:2}\mathbf{a}^1 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \tag{16.53}$$

这时，两个原型向量都有与 $\mathbf{a}^1$ 相同的内积，即使第一个原型与 $\mathbf{a}^1$ 相等而第二个不等。这就叫做“子集/超集二难问题”。

子集/超集二难问题的一个解决办法是对原型模式进行规格化。即是说，当一个原型模式具有很大数量的非零项时，每个项的量值应该被减小。比如说，仍用上面遇到的问题，我们可以将 L1 – L2 矩阵改变如下：

$$\mathbf{W}^{1:2} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix} \tag{16.54}$$

则第二层的输入将是

$$\mathbf{W}^{1:2}\mathbf{a}^1 = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{2}{3} \end{bmatrix} \tag{16.55}$$

现在我们得到结论：第一个原型与 $\mathbf{a}^1$ 有较大的内积。第二层的第一个神经元将被激活。

在第 15 章的 Grossberg 网络中我们利用在第一层里规格化输入模式得到规格化的原型模式。在 ART1 网络中，利用 L1 – L2 学习规则中的“加强中心/抑制周围”的竞争来规格化原型模式。

2. 学习规则

$\mathbf{W}^{1:2}$的学习规则是

$$\begin{aligned}\frac{\mathrm{d}[{}_i\mathbf{w}^{1:2}(t)]}{\mathrm{d}t} = {} & a_i^2(t)[\{{}^+b - {}_i\mathbf{w}^{1:2}(t)\}\zeta[{}^+\mathbf{W}]\mathbf{a}^1(t) \\ & - \{{}_i\mathbf{w}^{1:2}(t) + {}^-b\}[{}^-\mathbf{W}]\mathbf{a}^1(t)]\end{aligned} \tag{16.56}$$ 16-18

其中

$${}^+\mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, {}^-\mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, {}^+\mathbf{W} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}, {}^-\mathbf{W} = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & 0 & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ 1 & 1 & \cdots & 0 \end{bmatrix} \tag{16.57}$$

这是一个 instar 学习规则的改进型。当第二层的神经元 $i$ 活跃时，$\mathbf{W}^{1:2}$的第 $i$ 行${}_i\mathbf{w}^{1:2}$在 $\mathbf{a}^1$ 的方向移动。方程(16.56)与标准的 instar 学习过程的区别是${}_i\mathbf{w}^{1:2}$的元素参与竞争，因此被规格化。在方程(16.56)右边的括号里，我们看到了一个并联模型的形式，它具有来自 $\mathbf{a}^1$ 的加强中心/抑制周围的输入连接。激励偏置值是 ${}^+\mathbf{b} = 1$(一个全为 1 的向量)，而抑制偏置值是 ${}^-\mathbf{b} = 0$，它们保证了${}_i\mathbf{w}^{1:2}$的元素在 0 与 1 之间。(回忆我们在 15 章对并联模型的讨论。)

**快速学习**　为证实方程(16.56)产生了规格化的原型模式。我们来考察一下其稳态运算。为了分析，我们假定第一层和第二层的输出在权值到达稳定状态前保持恒定不变。这被称为快速学习。

对元素 $w_{i,j}^{1:2}$的方程为

$$\frac{\mathrm{d}w_{i,j}^{1:2}(t)}{\mathrm{d}t} = a_i^2(t)\left[(1 - w_{i,j}^{1:2}(t))\zeta a_j^1(t) - w_{i,j}^{1:2}(t)\sum_{k\neq j} a_k^1(t)\right] \tag{16.58}$$

若假设在第二层($a_i^2(t) = 1$)神经元 $i$ 是活跃的，并且在方程(16.58)中置导数等于0，则有

$$0 = \left[(1 - w_{i,j}^{1:2})\zeta a_j^1 - w_{i,j}^{1:2}\sum_{k\neq j} a_k^1\right] \tag{16.59}$$

要找出 $w_{i,j}^{1:2}$的稳定状态值，我们考虑两种情况。第一，假设 $a_j^1 = 1$。于是有

$$0 = (1 - w_{i,j}^{1:2})\zeta - w_{i,j}^{1:2}(\|\mathbf{a}^1\|^2 - 1) = -(\zeta + \|\mathbf{a}^1\|^2 - 1)w_{i,j}^{1:2} + \zeta \tag{16.60}$$

或者

$$w_{i,j}^{1:2} = \frac{\zeta}{\zeta + \|\mathbf{a}^1\|^2 - 1} \tag{16.61}$$

16-19 (注意，由于 $\mathbf{a}^1$ 是一个二值向量，故有 $\sum_{k=1}^{S^1} a_k^1 = \|\mathbf{a}^1\|^2$。)

另一种情况，若 $a_j^1 = 0$，则等式(16.59)简化为

$$0 = -w_{i,j}^{1:2}\|\mathbf{a}^1\|^2 \tag{16.62}$$

或

$$w_{i,j}^{1:2} = 0 \tag{16.63}$$

归纳等式(16.61)和(16.63)得

$$_i\mathbf{w}^{1:2} = \frac{\zeta\mathbf{a}^1}{\zeta + \|\mathbf{a}^1\|^2 - 1} \tag{16.64}$$

其中 $\zeta > 1$，以保证分母不等于0。

这样，原型模式会是经过规格化的，这就解决了子集/超集二难问题。(这里的“规格化”，并不意味着所有的原型向量都具有单位欧几里德距离长度，而只是简单地指 $\mathbf{W}^{1:2}$含有较多非零元素的各行将具有较小的量值。在本例中，含有较多非零元素的向量实际上可以比含有较少非零元素的向量具有更短的长度。)

### 16.2.6 学习规则：L2 - L1

在 ART1 体系结构中，L2 - L1 连接 $\mathbf{W}^{1:2}$是用 outstar 规则训练的。L2 - L1 连接旨在回忆相应的原型模式(期望值)，以便它可以在第一层中与输入模式相比较或结合。当期望值与输入模式不匹配时，一个重置信号传到第二层，于是一个新的原型模式将被选中(正如我们前面几节里讨论的那样)。

$\mathbf{W}^{1:2}$的学习规则是一个典型的 outstar 方程：

$$\frac{\mathrm{d}[\mathbf{w}_j^{2:1}(t)]}{\mathrm{d}t} = a_j^2(t)[-\mathbf{w}_j^{2:1}(t) + \mathbf{a}^1(t)] \tag{16.65}$$

因此，如果第二层中神经元 $j$ 是活跃的(赢得了竞争)，那么 $\mathbf{W}^{2:1}$的第 $j$ 列被移向 $\mathbf{a}^1$ 模式。为了说明这一点，我们来考查方程(16.65)的稳定状态运算。

分析中我们假设采用快速学习方案，即第一层和第二层的输出在权值达到稳定状态之前

保持恒定不变。假设第二层神经元 $j$ 是活跃的，故 $a_j^2 = 1$。置等式(16.65)中导数为 0，则有 16-20

$$0 = -\mathbf{w}_i^{2:1} + \mathbf{a}^1 \quad 或 \quad \mathbf{w}_j^{2:1} = \mathbf{a}^1 \tag{16.66}$$

因此 $\mathbf{W}^{2:1}$的第 $j$ 列收敛于第一层的输出 $\mathbf{a}^1$。回忆等式(16.20)和等式(16.21)，我们知道 $\mathbf{a}^1$ 是输入模式与相应的原型模式的结合。因此，原型模式被修改为结合当前的输入模式(如果存在足够密切的匹配)。

始终牢记 $\mathbf{W}^{1:2}$和 $\mathbf{W}^{2:1}$是同时更新的。当第二层的神经元 $j$ 是活跃的，并且在期望值与输入模式之间存在着充分的匹配(这表明形成了谐振条件)时，$\mathbf{W}^{1:2}$的第 $j$ 行与 $\mathbf{W}^{2:1}$的第 $j$ 列即被调整。在快速学习中，$\mathbf{W}^{2:1}$的第 $j$ 列设设成 $\mathbf{a}^1$，而 $\mathbf{W}^{1:2}$的第 $j$ 行被设成 $\mathbf{a}^1$ 的规格化版本。

### 16.2.7 ART1 算法小结

至此我们分析了 ART1 体系结构的每个子系统。如果总结一下关键的稳定状态运算并把它们组织成一个算法，我们就可以一览 ART1 的所有运算。

1. *初始化*

ART1 算法从权值矩阵 $\mathbf{W}^{1:2}$和 $\mathbf{W}^{2:1}$的初始化开始。矩阵 $\mathbf{W}^{2:1}$初始化为全 1。这样，第二层中的一个新神经元首次赢得了竞争，谐振就会发生。事实上，$\mathbf{a}^1 = \mathbf{p} \cap \mathbf{w}_j^{2:1} = \mathbf{p}$，因此得到 $\|\mathbf{a}^1\|^2 / \|\mathbf{p}\|^2 = 1 > \rho$。这意味着 $\mathbf{W}^{2:1}$中的任一未经训练的列都是一个有效的空白记录从而可以和任何输入模式发生匹配。

既然矩阵 $\mathbf{W}^{1:2}$的每一行都应该是 $\mathbf{W}^{2:1}$中各列的规格化版本，那么矩阵 $\mathbf{W}^{1:2}$的每个元素都被初始化为 $\zeta/(\zeta + S^1 - 1)$。

2. *算法*

初始化后，ART1 算法执行如下：

1) 首先，我们向网络提交一个输入模式。因为第二层被初始化为不活跃的(即每个 $a_j^2 = 0$)，故第一层的输出为(等式(16.20))

$$\mathbf{a}^1 = \mathbf{p} \tag{16.67}$$ 16-21

2) 其次，我们计算第二层的输入

$$\mathbf{W}^{1:2}\mathbf{a}^1 \tag{16.68}$$

并且用最大的输入(等式(16.37))激活第二层神经元：

$$a_i^2 = \begin{cases} 1, & 如果((_i\mathbf{w}^{1:2})^T\mathbf{a}^1 = max[(_k\mathbf{w}^{1:2})^T\mathbf{a}^1]) \\ 0, & 其他 \end{cases} \tag{16.69}$$

在平局的情况下，具有最小下标的神经元被宣布为获胜神经元。

3) 然后我们计算 L2 - L1 期望值(假定第二层中神经元 $j$ 被激活)：

$$\mathbf{W}^{2:1}\mathbf{a}^2 = \mathbf{w}_j^{2:1} \tag{16.70}$$

4) 现在第二层已被激活，我们调整第一层的输出使它包含 L2 - L1 期望值(等式(16.21))：

$$\mathbf{a}^1 = \mathbf{p} \cap \mathbf{w}_j^{2:1} \tag{16.71}$$

5) 然后，由调整子系统判定期望值与输入模式(等式(16.49))的匹配程度：

$$a^0 = \begin{cases} 1, & \text{如果}[\|\mathbf{a}^1\|^2/\|\mathbf{p}\|^2 < \rho] \\ 0, & \text{其他} \end{cases} \tag{16.72}$$

6) 若 $a^0 = 1$，则令 $a_j^2 = 0$，抑制它直到发生足够的匹配(谐振)，返回第 1 步。若 $a^0 = 0$，继续第 7 步。

7) 谐振发生。更新 $\mathbf{W}^{1:2}$的第 $j$ 行(等式(16.61))：

$$_j\mathbf{w}^{1:2} = \frac{\zeta \mathbf{a}^1}{\zeta + \|\mathbf{a}^1\|^2 - 1} \tag{16.73}$$

8) 更新 $\mathbf{W}^{2:1}$的第 $j$ 列(等式(16.61))：

$$\mathbf{w}_j^{2:1} = \mathbf{a}^1 \tag{16.74}$$

9) 撤消输入模式，恢复第二层中所有抑制的神经元，然后返回第 1 步接收新的输入模式。

16-22

输入模式继续被应用到网络中直到所有权值稳定(不再改变)。Carpenter 和 Grossberg 已证明[CaGr87a]ART1 算法对任何一组的输入模式都能形成稳定的聚类。

ART1 运算法则的详细例子参见例题 P16.5，P16.6 和 P16.7。

试验 ART1 算法请用 *Neural Network Design Demonstration ART1* (**nnd16a1**)。

### 16.2.8 其他 ART 体系结构

ART1 网络仅仅只是自适应谐振理论的一个例子。Carpenter 和 Grossberg 以及他们研究小组的其他人提出了这一主题的许多变型。

ART1 网络的一个不足是它只适用于二值输入模式。Carpenter 和 Grossberg 提出了 ART1 的一种变型，叫做 ART2，可用来处理二值或模拟的输入模式[CaGr87b]。除第一层外 ART2 的基本结构与 ART1 非常相似。ART2 中第一层被几个子层的代替。这些子层是必需的，因为模拟向量，不像二值向量，可以彼此任意地靠近。子层规格化过程与清除噪声的工作相结合，同时还执行调整子系统所需要的输入向量与期望值的比较工作。

Carpenter 和 Grossberg 后来提出了 ART3 网络[CaGr90]，其中介绍了一种比 ART 所要求的重置机构的更为复杂的生物学模型。直到现在，这种网络尚未被广泛地应用。1991 年，Carpenter，Grossberg 和 Reynolds 介绍了 ARTMAP 网络[CaGrRe91]。与以前的所有 ART 网络相比，它是一个有监督的网络。ARTMAP 体系结构由两个 ART 模块构成，两模块由一个称为“中间 ART”的联想存储器相连接。一个 ART 模块用来接受输入向量，而另一个模块用来接受预定的输出向量。这种网络学习的是，每当有输入向量的时候它能预测正确的输出。

近来，Carpenter，Grossberg，Markuzon，Reynolds 和 Rosen 又修改了 ARTMAP 的体系结构，结合进了模糊逻辑。其结果称为“模糊 ARTMAP”[CaGrMa92]。它看上去性能有所提高，尤其是对含噪音的输入模式。

16-23

所有这些 ART 体系结构都结合了本章讨论的主要模块，包括：

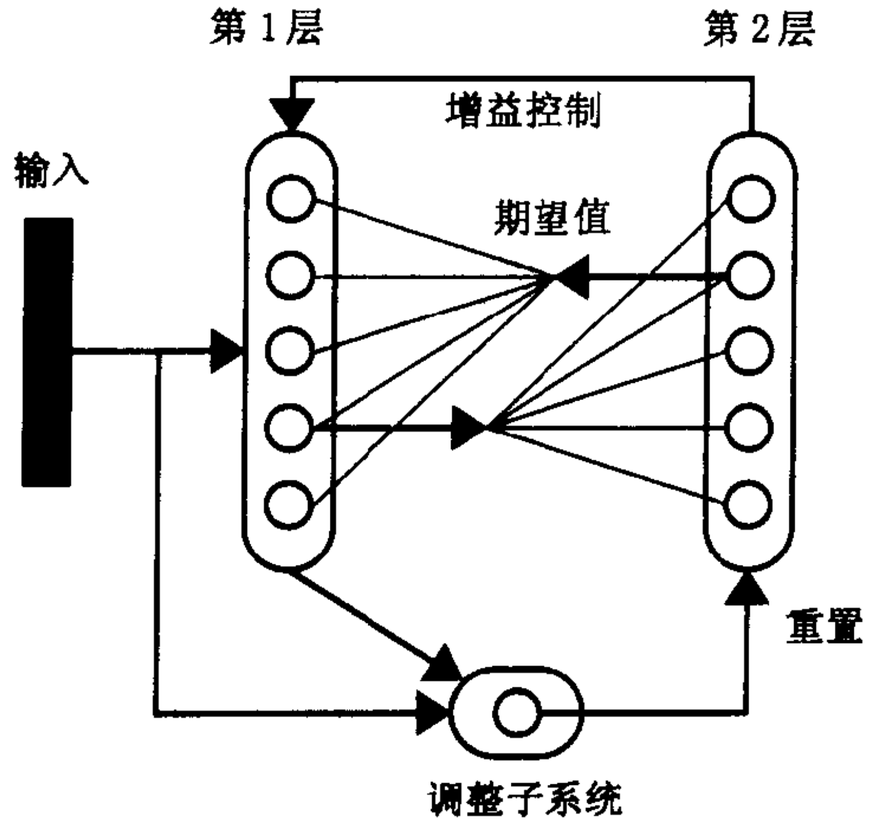
- L1 – L2 instar 模式识别
- L2 – L1 outstar 模式回忆

- 用于对比增强(竞争)的第二层
- 用于输入与期望值比较的第一层
- 当模式不匹配时用于重置的调整子系统

16-24

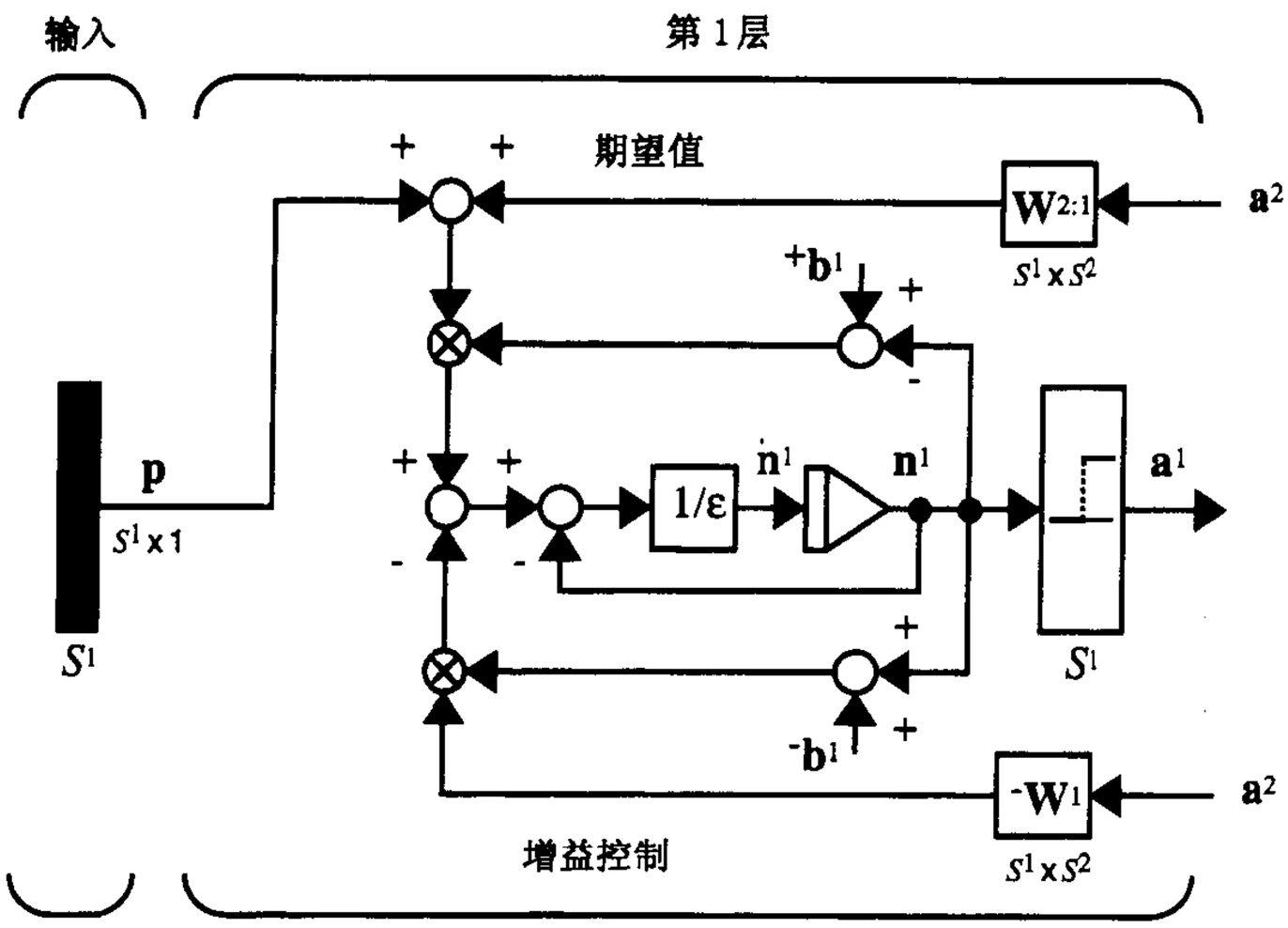
## 16.3 小结

### 基本的 ART 体系结构




### ART1 网络(二值模式)

#### ART1 第一层




$$\varepsilon\, dn^1/dt = -n^1 + ({}^+b^1 - n^1)\{p + W^{2:1}a^2\} - (n^1 + {}^-b^1)[{}^-W^1]a^2$$

16-25

第一层方程

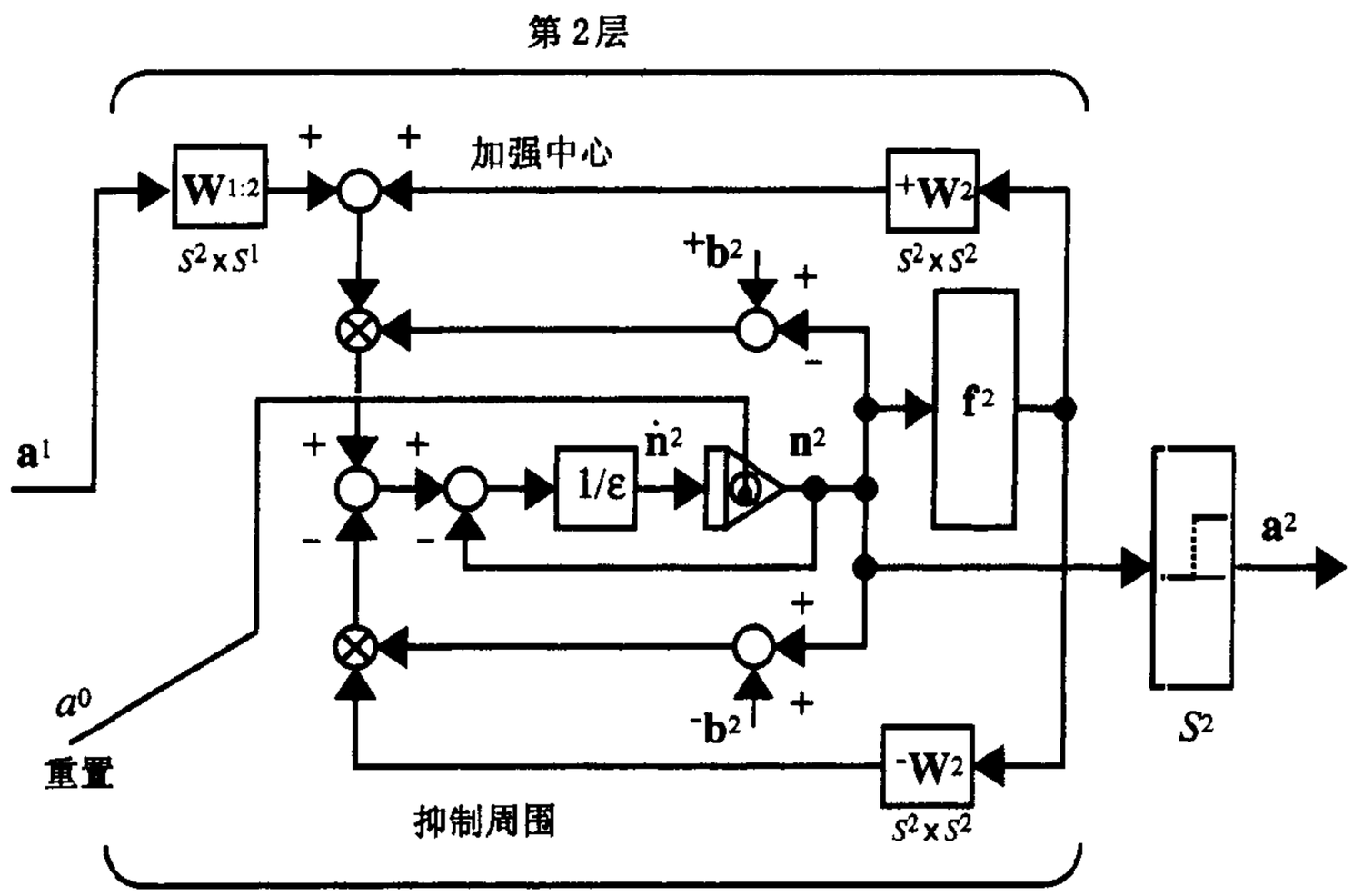
$$\varepsilon\frac{d\mathbf{n}^1(t)}{dt}=-\mathbf{n}^1(t)+({}^+\mathbf{b}^1-\mathbf{n}^1(t))\{\mathbf{p}+\mathbf{W}^{2:1}\mathbf{a}^2(t)\}-(\mathbf{n}^1(t)+{}^-\mathbf{b}^1)[{}^-\mathbf{W}^1]\mathbf{a}^2(t)$$

稳定状态运算

若第二层不活跃(即每个 $a_j^2=0$)，$\mathbf{a}^1=\mathbf{p}$。

若第二层活跃(即有一个 $a_j^2=1$)，$\mathbf{a}^1=\mathbf{p}\cap\mathbf{w}_j^{2:1}$)。

**ART1 第二层**




$$\varepsilon\, d\mathbf{n}^2/dt=-\mathbf{n}^2+({}^+\mathbf{b}^2-\mathbf{n}^2)\{[{}^+\mathbf{W}^2]\mathbf{f}^2(\mathbf{n}^2)+\mathbf{W}^{1:2}\mathbf{a}^1\}$$
$$-(\mathbf{n}^2+{}^-\mathbf{b}^2)[{}^-\mathbf{W}^2]\mathbf{f}^2(\mathbf{n}^2)$$

第二层方程

$$\varepsilon\frac{d\mathbf{n}^2(t)}{dt}=-\mathbf{n}^2(t)+({}^+\mathbf{b}^2-\mathbf{n}^2(t))\{[{}^+\mathbf{W}^2]\mathbf{f}^2(\mathbf{n}^2(t))+\mathbf{W}^{1:2}\mathbf{a}^1\}$$
$$-(\mathbf{n}^2(t)+{}^-\mathbf{b}^2)[{}^-\mathbf{W}^2]\mathbf{f}^2(\mathbf{n}^2(t))$$

稳定状态运算

16-26

$$a_i^2=\begin{cases}1, & \text{如果}((_i\mathbf{w}^{1:2})^T\mathbf{a}^1=max[(_j\mathbf{w}^{1:2})^T\mathbf{a}^1])\\ 0, & \text{其他}\end{cases}$$

**调整子系统**

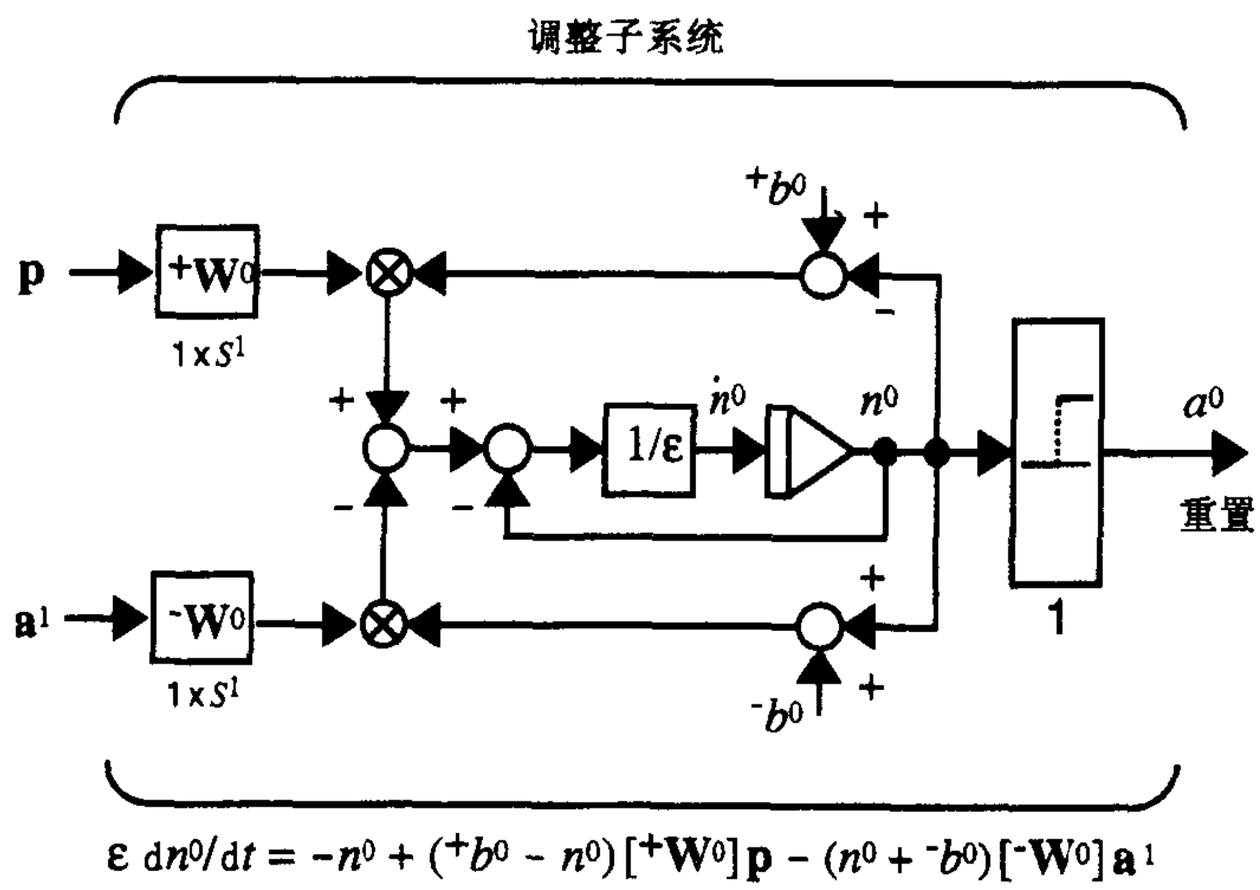
调整子系统方程

$$\varepsilon\frac{dn^0(t)}{dt}=-n^0(t)+({}^+b^0-n^0(t))\{{}^+\mathbf{W}^0\mathbf{p}\}-(n^0(t)+{}^-b^0)\{{}^-\mathbf{W}^0\mathbf{a}^1\}$$

其中 ${}^+\mathbf{W}^0=[\alpha\ \ \alpha\ \ \cdots\ \ \alpha]$，${}^-\mathbf{W}^0=[\beta\ \ \beta\ \ \cdots\ \ \beta]$，${}^+b^0={}^-b^0=1$。

稳定状态运算

$$a^0=\begin{cases}1, & \text{如果}[\|\mathbf{a}^1\|^2/\|\mathbf{p}\|^2<\rho]\\ 0, & \text{其他}\end{cases}$$




**L1－L2 学习规则**

$$\frac{\mathrm{d}[{}_i\mathbf{w}^{1:2}(t)]}{\mathrm{d}t} = a_i^2(t)\left[\{{}^+\mathbf{b} - {}_i\mathbf{w}^{1:2}(t)\}\zeta[{}^+\mathbf{W}]\mathbf{a}^1(t) - \{{}_i\mathbf{w}^{1:2}(t) + {}^-\mathbf{b}\}[{}^-\mathbf{W}]\mathbf{a}^1(t)\right]$$

$${}^+\mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, {}^-\mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, {}^+\mathbf{W} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}, {}^-\mathbf{W} = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & 0 & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ 1 & 1 & \cdots & 0 \end{bmatrix} \tag{16-27}$$

稳定状态运算(快速学习)

$${}_i\mathbf{w}^{1:2} = \frac{\zeta\mathbf{a}^1}{\zeta + \|\mathbf{a}^1\|^2 - 1}\text{(第二层神经元 } i \text{ 活跃)}$$

**L2－L1 学习规则**

$$\frac{\mathrm{d}[\mathbf{w}_j^{2:1}(t)]}{\mathrm{d}t} = a_j^2(t)[-\mathbf{w}_j^{2:1}(t) + \mathbf{a}^1(t)]$$

稳定状态运算(快速学习)

$$\mathbf{w}_j^{2:1} = \mathbf{a}^1\text{(第二层神经元 } j \text{ 活跃)}$$

**ART1 算法(快速学习)小结**

*初始化*

矩阵 $\mathbf{W}^{2:1}$初始化为全 1。

矩阵 $\mathbf{W}^{1:2}$的每个元素初始化为 $\zeta/(\zeta + S^1 - 1)$。

*算法*

1) 首先，向网络提交一个输入模式。既然第二层初始化为不活跃(即每个 $a_j^2 = 0$)，则第一层输出为

$$\mathbf{a}^1 = \mathbf{p}$$

2) 其次，计算第二层的输入

$$\mathbf{W}^{1:2}\mathbf{a}^1$$

并且用输入中的最大值激活第二层的神经元：

$$a_i^2 = \begin{cases} 1, & \text{如果}((_i\mathbf{w}^{1:2})^T\mathbf{a}^1 = max[(_k\mathbf{w}^{1:2})^T\mathbf{a}^1]) \\ 0, & \text{其他} \end{cases}$$

在平局的情形下，具有最小下标的神经元被宣布为优胜者。

3）计算 L2 - L1 期望值(其中假设第二层神经元 $j$ 活跃)

16-28

$$\mathbf{W}^{2:1}\mathbf{a}^2 = \mathbf{w}_j^{2:1}$$

4）现在第二层是活跃的，调整第一层的输出以包含 L2 - L1 期望值：

$$\mathbf{a}^1 = \mathbf{p} \cap \mathbf{w}_j^{2:1}$$

5）然后，调整子系统判定期望值与输入模式的匹配程度：

$$a^0 = \begin{cases} 1, & \text{如果}[\|\mathbf{a}^1\|^2/\|\mathbf{p}\|^2 < \rho] \\ 0, & \text{其他} \end{cases}$$

6）若 $a^0 = 1$，则令 $a_j^2 = 0$，抑制它直到适当的匹配发生(谐振)，返回第 1 步。若 $a^0 = 0$，继续第 7 步。

7）谐振发生，因此更新 $\mathbf{W}^{1:2}$的第 $j$ 行：

$$_j\mathbf{w}^{1:2} = \frac{\zeta\mathbf{a}^1}{\zeta + \|\mathbf{a}^1\|^2 - 1}$$

8）现在更新 $\mathbf{W}^{2:1}$的第 $j$ 列：

$$\mathbf{w}_j^{2:1} = \mathbf{a}^1$$

9）撤消输入模式，恢复第二层中所有被抑制的神经元，然后返回第 1 步，接受新的输入模式。

16-29

## 16.4 例题

**P16.1** 考虑 ART1 网络的第一层具有如下参数：

$$\varepsilon = 0.01, {}^+b^1 = 2, {}^-b^1 = 3$$

假设第二层有两个神经元，输入向量中有两个元素，并有如下的权值矩阵和输入：

$$\mathbf{W}^{2:1} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \quad \mathbf{p} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

再假设第二层神经元 1 活跃。

（i）找出且描绘响应 $\mathbf{n}^1$。

（ii）检查(i)的答案是否满足稳定状态运算方程(16.21)的预测。

**解**

（i）因为第二层活跃，且第二层神经元 1 赢得了竞争，所以第一层的运算方程为

$$(0.01)\frac{dn_1^1}{dt} = -n_1^1 + (2 - n_1^1)\{p_1 + w_{1,1}^{2:1}\} - (n_1^1 + 3)$$
$$= -n_1^1 + (2 - n_1^1)\{1 + 0\} - (n_1^1 + 3) = -3n_1^1 - 1$$
$$(0.01)\frac{dn_2^1}{dt} = -n_2^1 + (2 - n_2^1)\{p_2 + w_{2,1}^{2:1}\} - (n_2^1 + 3)$$
$$= -n_2^1 + (2 - n_2^1)\{1 + 1\} - (n_2^1 + 3) = -4n_2^1 + 1$$

化简为

$$\frac{dn_1^1}{dt} = -300n_1^1 - 100$$

$$\frac{dn_2^1}{dt} = -400n_2^1 + 100$$

若假设两个神经元的初始条件都为 0，则结果为 16-30

$$n_1^1(t) = -\frac{1}{3}[1 - e^{-300t}]$$

$$n_2^1(t) = \frac{1}{4}[1 - e^{-400t}]$$

它们的图形见图 16-9。

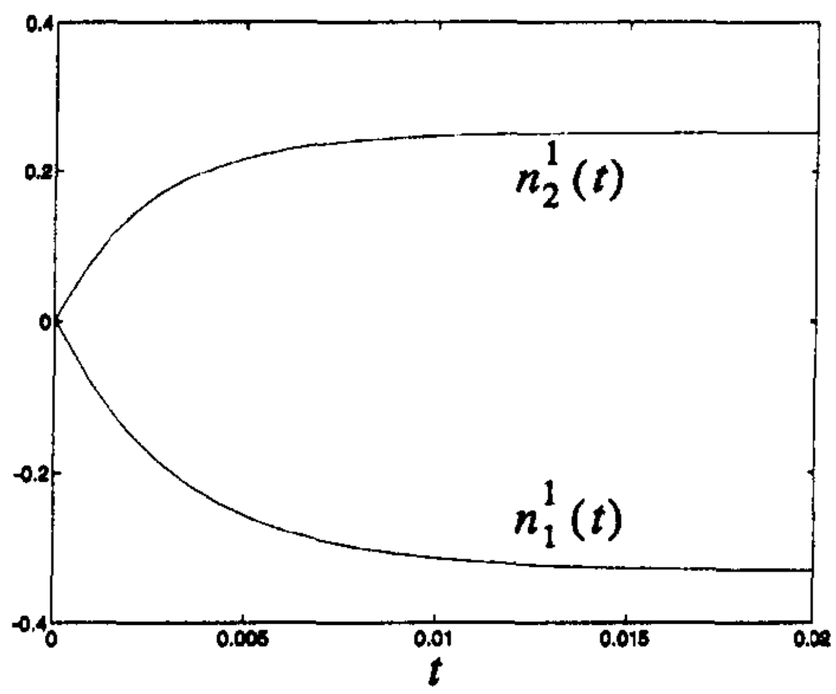



图 16-9 第一层的响应

（ii）请注意，$n_1^1(t)$收敛于负值，而 $n_2^1(t)$收敛于正值。因此 $a_1^1(t)$收敛于 0，$a_2^1(t)$收敛于 1(回忆第一层的转移函数为 *hardlim*$^+$)。这与我们的稳态分析一致(见等式(16.21))，这是因为

$$\mathbf{p} \cap \mathbf{w}_1^{2:1} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \cap \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \mathbf{a}^1 \tag{16.75}$$

**P16.2** 考虑 ART1 网络的第二层具有如下参数：

$$\varepsilon = 0.1, {}^+\mathbf{b}^2 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, {}^-\mathbf{b}^2 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \mathbf{W}^{1:2} = \begin{bmatrix} ({}_1\mathbf{w}^{1:2})^T \\ ({}_2\mathbf{w}^{1:2})^T \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 \\ 1 & 0 \end{bmatrix}$$

且

$$f^2(n) = \begin{cases} 10(n^2), & n \geq 0 \\ 0, & n < 0 \end{cases}$$ 16-31

假设第一层的输出为

$$\mathbf{a}^1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

除了偏置值不同以外，它和正文中第二层的例子是一样的。

（i）写出第二层的运算方程，并且模仿描绘出其响应图。解释偏置值增加带来的影响。

（ii）证实第二层稳定状态方程的正确性。

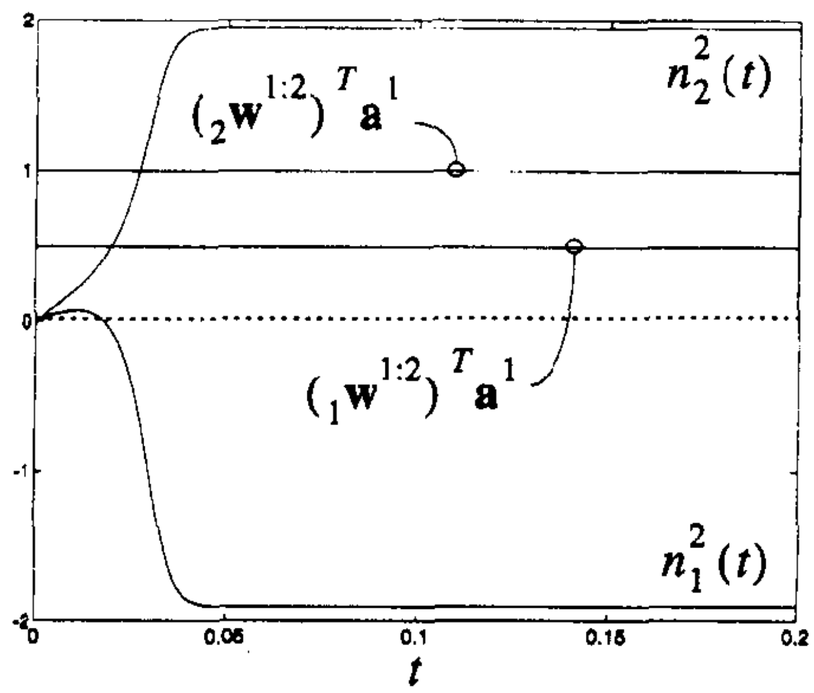
**解**

（i）该层的运算方程为

$$(0.1)\frac{\mathrm{d}n_1^2(t)}{\mathrm{d}t} = -n_1^2(t) + (2 - n_1^2(t))\{f^2(n_1^2(t)) + ({}_1\mathbf{w}^{1:2})^T\mathbf{a}^1\} - (n_1^2(t) + 2)f^2(n_2^2(t))$$

$$(0.1)\frac{\mathrm{d}n_2^2(t)}{\mathrm{d}t} = -n_2^2(t) + (2 - n_2^2(t))\{f^2(n_2^2(t)) + ({}_2\mathbf{w}^{1:2})^T\mathbf{a}^1\} - (n_2^2(t) + 2)f^2(n_1^2(t))$$

图 16-10 展示了当输入向量为 $\mathbf{a}^1 = [1 \quad 0]^T$ 时第二层的响应。$\mathbf{W}^{1:2}$的第二行与 $\mathbf{a}^1$ 作用有比第一行更大的内积，因此第二个神经元赢得了竞争。




16-32

图 16-10 第二层的响应

如果我们将图 16-10 与图 16-6 作个比较，可以看到偏置值有三个影响。第一，响应速度增加了；神经元的输出更快地转向它们的稳定状态值。第二，响应的范围从[ - 1，1] 增加到[ - 2，2]。（回忆第 15 章并联模型中上限是激励偏置值 $^+b$，而下限是抑制偏置值 $^-b$。）第三，神经元的响应更加接近上限或下限。

（ii）在稳定状态，$n_1^2(t)$有一个正值，$n_2^2(t)$有一个负值。第二层的稳定状态输出会是：

$$\mathbf{a}^2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

这与第二层要求的稳态响应特征一致：

$$a_i^2 = \begin{cases} 1, & \text{当}(({}_i\mathbf{w}^{1:2})^T\mathbf{a}^1 = max[({}_j\mathbf{w}^{1:2})^T\mathbf{a}^1])\text{时} \\ 0, & \text{其他} \end{cases}$$

**P16.3** 考虑 ARTI 网络具有如下参数的调整子系统：

$$\varepsilon = 0.1, \quad \alpha = 0.5, \quad \beta = 2(\rho = 0.25), \quad {}^+b^0 = {}^-b^0 = 0.5$$

调整整子系统的输入为

$$\mathbf{p} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \mathbf{a}^1 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

（i）找出并描绘调整子系统的响应 $n^0(t)$。

（ii）证实其满足稳定状态条件。

**解**

（i）调整子系统的运算方程为

$$(0.1)\frac{\mathrm{d}n^0(t)}{\mathrm{d}t} = -n^0(t) + (0.5 - n^0(t))\{0.5(p_1 + p_2 + p_3)\} - (n^0(t) + 0.5)\{2(a_1^1 + a_2^1 + a_3^1)\} \quad \boxed{16\text{-}33}$$

或者

$$\frac{\mathrm{d}n^0(t)}{\mathrm{d}t} = -65n^0(t) - 12.5$$

故其响应为

$$n^0(t) = -0.1923[1 - e^{-65t}]$$

该响应如图 16-11 所示。此例中，因为 $n^0(t)$是负值，$a^0 = hardlim^+(n^0) = 0$，所以不会有重置信号发向第二层。

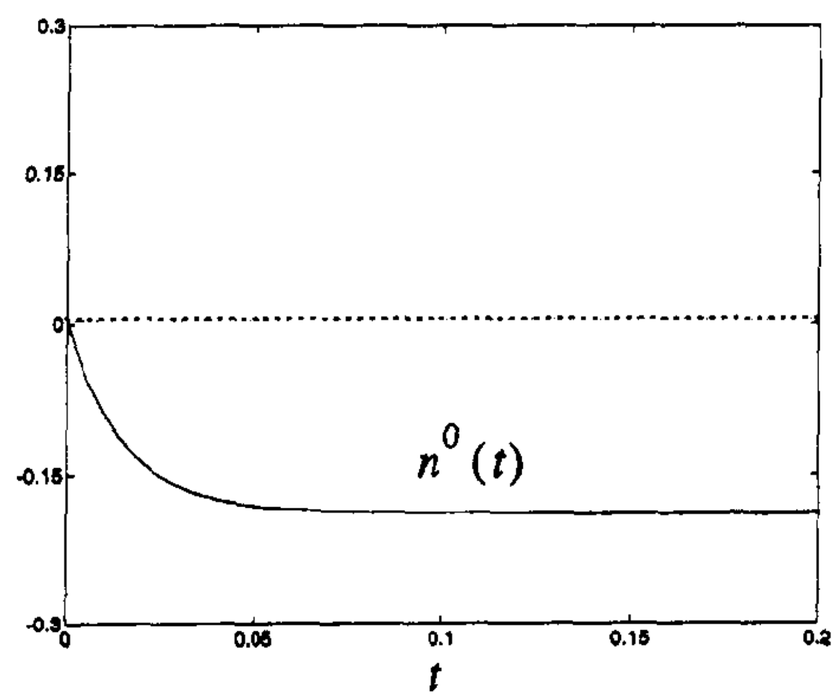



图 16-11　调整子系统的响应

（ii）调整子系统的稳定状态运算可以总结如下：

$$a^0 = \begin{cases} 1, & 当[\|\mathbf{a}^1\|^2/\|\mathbf{p}\|^2 < \rho]时 \\ 0, & 其他 \end{cases}$$

在本题中

$$\|\mathbf{a}^1\|^2/\|\mathbf{p}\|^2 = \left\|\begin{bmatrix}1\\0\\1\end{bmatrix}\right\|^2 / \left\|\begin{bmatrix}1\\1\\1\end{bmatrix}\right\|^2 = \frac{2}{3} > \rho = 0.25 \quad \boxed{16\text{-}34}$$

故 $a^0 = 0$，与(i)的结果一致。

**P16.4** 说明 L2 - L1 连接的学习方程与第 13 章所述 outstar 方程等价。

L2 - L1 学习规则为(等式(16.65))

$$\frac{\mathrm{d}[\mathbf{w}_j^{2:1}(t)]}{\mathrm{d}t} = a_j^2(t)[-\mathbf{w}_j^{2:1}(t) + \mathbf{a}^1(t)]$$

若用

$$\frac{\mathrm{d}[\mathbf{w}_j^{2:1}(t)]}{\mathrm{d}t} \approx \frac{\mathbf{w}_j^{2:1}(t+\Delta t)-\mathbf{w}_j^{2:1}(t)}{\Delta t}$$

作为导数部分的近似值，则等式(16.65)可重写为

$$\mathbf{w}_j^{2:1}(t+\Delta t) = \mathbf{w}_j^{2:1}(t) + (\Delta t)a_j^2(t)\{-\mathbf{w}_j^{2:1}(t)+\mathbf{a}^1(t)\}$$

这就是第 13 章的 outstar 规则(等式(13.51))。这里，L2 - L1 连接的输入是 $a_j^2(t)$，L2 - L1 连接的输出是 $\mathbf{a}^1$。

**P16.5** 用下面的输入向量训练 ART1 网络：

$$\mathbf{p}_1 = \begin{bmatrix}0\\1\\0\end{bmatrix},\quad \mathbf{p}_2 = \begin{bmatrix}1\\0\\0\end{bmatrix},\quad \mathbf{p}_3 = \begin{bmatrix}1\\1\\0\end{bmatrix}$$

利用参数 $\zeta=2$，$\rho=0.4$，选择 $S^2=3$(3 个分类)。

**解**

我们的初始权值将是

$$\mathbf{W}^{2:1} = \begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix},\quad \mathbf{W}^{1:2} = \begin{bmatrix}0.5&0.5&0.5\\0.5&0.5&0.5\\0.5&0.5&0.5\end{bmatrix}$$

现在来讨论算法。

1) 计算第一层的响应：

$$\mathbf{a}^1 = \mathbf{p}_1 = \begin{bmatrix}0\\1\\0\end{bmatrix}$$

16-35

2) 然后计算第二层的输入：

$$\mathbf{W}^{1:2}\mathbf{a}^1 = \begin{bmatrix}0.5&0.5&0.5\\0.5&0.5&0.5\\0.5&0.5&0.5\end{bmatrix}\begin{bmatrix}0\\1\\0\end{bmatrix} = \begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}$$

由于所有神经元都有相同的输入，选取第一个神经元作为优胜者。(在平局的情况下，选取下标最小的神经元作为优胜者。)

$$\mathbf{a}^2 = \begin{bmatrix}1\\0\\0\end{bmatrix}$$

3) 现在计算 L2 - L1 期望值：

$$\mathbf{W}^{2:1}\mathbf{a}^2 = \begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}\begin{bmatrix}1\\0\\0\end{bmatrix} = \begin{bmatrix}1\\1\\1\end{bmatrix} = \mathbf{w}_1^{2:1}$$

4) 调整第一层输出以包含 L2 - L1 期望值：

$$\mathbf{a}^1 = \mathbf{p}_1 \cap \mathbf{w}_1^{2:1} = \begin{bmatrix}0\\1\\0\end{bmatrix} \cap \begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}0\\1\\0\end{bmatrix}$$

5）然后，调整子系统判定期望值与输入模式的匹配程度：

$$\|\mathbf{a}^1\|/\|\mathbf{p}_1\|^2 = \frac{1}{1} > \rho = 0.4,\text{因此 } a^0 = 0(\text{不重置})$$

6）既然 $a^0=0$，继续第 7 步。

7）谐振发生，因而更新 $\mathbf{W}^{1:2}$的第 1 行：

$$_1\mathbf{w}^{1:2} = \frac{2\mathbf{a}^1}{2+\|\mathbf{a}^1\|^2-1} = \mathbf{a}^1 = \begin{bmatrix}0\\1\\0\end{bmatrix},\quad \mathbf{W}^{1:2} = \begin{bmatrix}0 & 1 & 0\\0.5 & 0.5 & 0.5\\0.5 & 0.5 & 0.5\end{bmatrix}$$

8）更新 $\mathbf{W}^{2:1}$的第 1 列：

16-36

$$\mathbf{w}_1^{2:1} = \mathbf{a}^1 = \begin{bmatrix}0\\1\\0\end{bmatrix},\quad \mathbf{W}^{2:1} = \begin{bmatrix}0 & 1 & 1\\1 & 1 & 1\\0 & 1 & 1\end{bmatrix}$$

9）撤消 $\mathbf{p}_1$，返回第 1 步，接收输入模式 $\mathbf{p}_2$。

1）计算新的第一层响应(第二层不活跃)：

$$\mathbf{a}^1 = \mathbf{p}_2 = \begin{bmatrix}1\\0\\0\end{bmatrix}$$

2）然后，计算第二层的输入

$$\mathbf{W}^{1:2}\mathbf{a}^1 = \begin{bmatrix}0 & 1 & 0\\0.5 & 0.5 & 0.5\\0.5 & 0.5 & 0.5\end{bmatrix}\begin{bmatrix}1\\0\\0\end{bmatrix} = \begin{bmatrix}0\\0.5\\0.5\end{bmatrix}$$

由于神经元 2 与神经元 3 有相同输入，取神经元 2 作为优胜者：

$$\mathbf{a}^2 = \begin{bmatrix}0\\1\\0\end{bmatrix}$$

3）现在计算 L2 – L1 期望值：

$$\mathbf{W}^{2:1}\mathbf{a}^2 = \begin{bmatrix}0 & 1 & 1\\1 & 1 & 1\\0 & 1 & 1\end{bmatrix}\begin{bmatrix}0\\1\\0\end{bmatrix} = \mathbf{w}_2^{2:1} = \begin{bmatrix}1\\1\\1\end{bmatrix}$$

4）调整第一层输出以包含 L2 – L1 期望值：

$$\mathbf{a}^1 = \mathbf{p}_2 \cap \mathbf{w}_2^{2:1} = \begin{bmatrix}1\\0\\0\end{bmatrix} \cap \begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}1\\0\\0\end{bmatrix}$$

5）然后，调整子系统判定期望值与输入模式的匹配程度：

$$\|\mathbf{a}^1\|^2/\|\mathbf{p}_2\|^2 = \frac{1}{1} > \rho = 0.4,\text{因此 } a^0 = 0(\text{不重置})。$$

16-37

6）既然 $a^0=0$，继续第 7 步。

7）谐振发生，因而更新 $\mathbf{W}^{1:2}$的第二行：

$$ {}_2\mathbf{w}^{1:2} = \frac{2\mathbf{a}^1}{2 + \|\mathbf{a}^1\|^2 - 1} = \mathbf{a}^1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{W}^{1:2} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0.5 & 0.5 & 0.5 \end{bmatrix} $$

8) 更新 $\mathbf{W}^{2:1}$的第二列：

$$ \mathbf{w}_2^{2:1} = \mathbf{a}^1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{W}^{2:1} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} $$

9) 撤消 $\mathbf{p}_2$，返回第 1 步，接收输入模式 $\mathbf{p}_3$。

1) 计算第一层对新输入模式的响应：

$$ \mathbf{a}^1 = \mathbf{p}_3 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} $$

2) 然后，计算第二层的输入：

$$ \mathbf{W}^{1:2}\mathbf{a}^1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0.5 & 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} $$

由于所有神经元都有相同输入，选取神经元 1 作为优胜者：

$$ \mathbf{a}^2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} $$

3) 现在计算 L2 - L1 期望值：

$$ \mathbf{W}^{2:1}\mathbf{a}^2 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \mathbf{w}_1^{2:1} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} $$

16-38

4) 调整第一层输出以包含 L2 - L1 期望值：

$$ \mathbf{a}^1 = \mathbf{p}_3 \cap \mathbf{w}_1^{2:1} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \cap \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} $$

5) 然后，调整子系统判定期望值与输入模式之间的匹配程度：

$$ \|\mathbf{a}^1\|^2 / \|\mathbf{p}_3\|^2 = \frac{1}{2} > \rho = 0.4, \text{因此 } a^0 = 0(\text{不重置}) $$

6) 既然 $a^0 = 0$，继续第 7 步。

7) 谐振发生，因而更新 $\mathbf{W}^{1:2}$的第 1 行：

$$ {}_1\mathbf{w}^{1:2} = \frac{2\mathbf{a}^1}{2 + \|\mathbf{a}^1\|^2 - 1} = \mathbf{a}^1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{W}^{1:2} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0.5 & 0.5 & 0.5 \end{bmatrix} $$

8) 更新 $\mathbf{W}^{2:1}$的第一列：

$$\mathbf{w}_2^{2:1} = \mathbf{a}^1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{W}^{2:1} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

这时训练就结束了。即使你再训练这三个模式中的任一个，权值都不会改变。这些模式已被成功的聚类。这种结果形式(稳定的学习过程)对于 ARTL 算法是必然的，事实上，它被证明总能形成稳定的聚类。

**P16.6** 重复例题 P16.5，但是改变警戒参数为 $\rho = 0.6$。

**解**

训练过程与例题 P16.5 完全一样，直到出现模式 $\mathbf{p}_3$。让我们从这里继续算法。

1) 计算第一层的响应：

$$\mathbf{a}^1 = \mathbf{p}_3 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

16-39

2) 然后，计算第二层的输入：

$$\mathbf{W}^{1:2}\mathbf{a}^1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

既然所有的神经元都有相同的输入，选取神经元 1 作为优胜者：

$$\mathbf{a}^2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

3) 现在计算 L2 - L1 期望值：

$$\mathbf{W}^{2:1}\mathbf{a}^2 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \mathbf{w}_1^{2:1} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

4) 调整第一层的输出以包括 L2 - L1 期望值：

$$\mathbf{a}^1 = \mathbf{p}_3 \cap \mathbf{w}_1^{2:1} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \cap \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

5) 然后，调整子系统判定期望值与输入模式的匹配程度：

$$\|\mathbf{a}^1\|^2 / \|\mathbf{p}_3\|^2 = \frac{1}{2} < \rho = 0.6, \text{因而 } a^0 = 1(\text{重置})$$

6) 因为 $a^0 = 1$，令 $a_1^2 = 0$，抑制它直到出现足够的匹配(谐振)，然后返回第 1 步。

1) 重新计算第一层的响应(第二层不活跃)：

$$\mathbf{a}^1 = \mathbf{p}_3 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

16-40

2) 其次，计算第二层的输入：

$$\mathbf{W}^{1:2}\mathbf{a}^1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

既然神经元 1 被抑制，选取神经元 2 作为优胜者：

$$\mathbf{a}^2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

3）现在，计算 L2 - L1 期望值：

$$\mathbf{W}^{2:1}\mathbf{a}^2 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \mathbf{w}_2^{2:1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

4）调整第一层输出以包含 L2 - L1 期望值：

$$\mathbf{a}^1 = \mathbf{p}_3 \cap \mathbf{w}_2^{2:1} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \cap \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

5）然后，调整子系统判定期望值与输入模式之间的匹配程度：

$$\|\mathbf{a}^1\|^2 / \|\mathbf{p}_3\|^2 = \frac{1}{2} < \rho = 0.6, \text{因而 } a^0 = 1(\text{重置})$$

6）因为 $a^0 = 1$，令 $a_2^2 = 0$，抑制它直到出现足够的匹配(谐振)，然后返回第 1 步。

1）重新计算第一层的响应：

$$\mathbf{a}^1 = \mathbf{p}_3 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

2）其次，计算第二层的输入

16-41

$$\mathbf{W}^{1:2}\mathbf{a}^1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

由于神经元 1 和神经元 2 被抑制，故选神经元 3 为优胜者：

$$\mathbf{a}^2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

3）现在计算 L2 - L1 期望值：

$$\mathbf{W}^{2:1}\mathbf{a}^2 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \mathbf{w}_3^{2:1} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

4）调整第一层输出以包含 L2 - L1 期望值：

$$\mathbf{a}^1 = \mathbf{p}_3 \cap \mathbf{w}_3^{2:1} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \cap \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

5）然后，调整子系统判定期望值与输入模式的匹配程度：

$$\|\mathbf{a}^1\|^2 / \|\mathbf{p}_3\|^2 = \frac{2}{2} > \rho = 0.6, \text{因而 } a^0 = 0(\text{不重置})$$

6）由于 $a^0 = 0$，继续第 7 步。

7）谐振发生，因而更新 $\mathbf{W}^{1:2}$的第 3 行：

$$ {}_3\mathbf{w}^{1:2} = \frac{2\mathbf{a}^1}{2 + \left\|\mathbf{a}^1\right\|^2 - 1} = \frac{2}{3}\mathbf{a}^1 = \begin{bmatrix} \frac{2}{3} \\ \frac{2}{3} \\ 0 \end{bmatrix}, \quad \mathbf{W}^{1:2} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ \frac{2}{3} & \frac{2}{3} & 0 \end{bmatrix} $$

8）更新 $\mathbf{W}^{2:1}$的第 3 列：

$$ \mathbf{w}_3^{2:1} = \mathbf{a}^1 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{W}^{2:1} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} $$

这时训练就结束了，即使再对这三个输入模式中的任一个进行训练，权值也不会发生改变。(对网络应用每一个输入模式自己证明这点。)这些模式已被成功地聚类。 16-42

注意在例题 P16.5 中，警戒值 $\rho = 0.4$，所以这些模式被聚类成两类。在本题中，警戒值 $\rho = 0.6$，这些模式被聚类成三类。警戒值越接近 1，就会聚类成越多的分类。这是因为输入模式必须很接近原型，以使其被相应的原型结合。当警戒值接近 0 时，许多不同的输入模式会被结合进同一个原型。警戒参数调节分类的近似程度。

**P16.7** 用下面的输入向量训练 ART1 网络(参见[CaGr87a])：

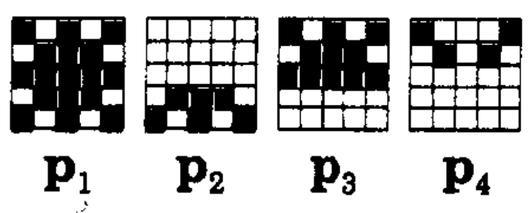



图 16-12

提交向量的顺序为 $\mathbf{p}_1 - \mathbf{p}_2 - \mathbf{p}_3 - \mathbf{p}_1 - \mathbf{p}_4$(即是说，$p_1$ 在每一个轮回中被提交了两次)。使用参数 $\zeta = 2$，$\rho = 0.6$，选择 $S^2 = 3$(3 个分类)。训练网络直到权值收敛。

**解**

我们从初始化权值矩阵开始。初始矩阵 $\mathbf{W}^{2:1}$是一个 $S^1 \times S^2 = 25 \times 3$ 的全 1 矩阵。初始矩阵 $\mathbf{W}^{1:2}$经过了规格化，因此它是一个 $S^2 \times S^1 = 3 \times 25$ 矩阵，其中每个元素等于

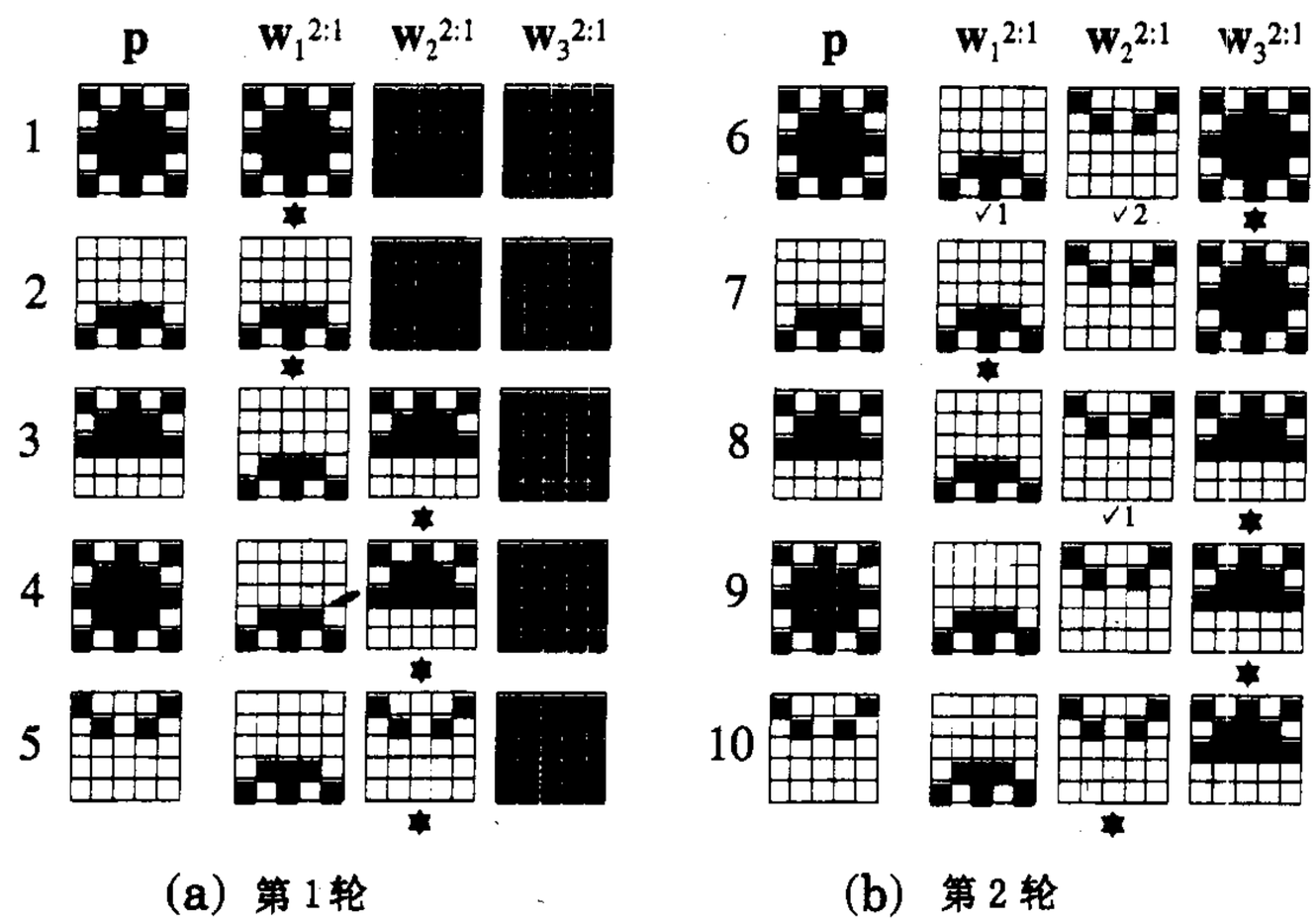
$$ \frac{\zeta}{(\zeta + S^1 - 1)} = \frac{2}{(2 + 25 - 1)} = 0.0769 $$

为了创建输入向量，我们要一行一行地扫描每个模式，其中每个蓝色方块都代表 1，每个白色方块都代表 0。因为输入模式是 $5 \times 5$ 网格，这将创建 25 维的输入向量。

我们现在开始训练。由于在向量数目如此大的情况下显示计算的全过程并不实际，故我们已在图 16-13 中总结了算法的结果。图中每一行代表 ATR1 算法的一次重复(一个输入向量的提交)。每行中最左边的模式是输入向量。剩下的模式代表 $\mathbf{W}^{2:1}$矩阵的三列。在每一次重复中，一个星号指出了谐振点——$\mathbf{W}^{2:1}$中与输入模式相匹配的那一列。每当发生重置，均被一个检查标志(勾号)反映出来。当在一次重复中不只一次重置发生，检查标志旁边的数字便记录了重置发生的次数。 16-43

共有 10 次算法的重复执行(依照次序 $\mathbf{p}_1 - \mathbf{p}_2 - \mathbf{p}_3 - \mathbf{p}_1 - \mathbf{p}_4$ 进行了两个轮回)。权值最终稳定。(读者可以自己提交每个输入模式进行检查。)

本例中有几点有趣的地方得注意。首先，注意在第 4 次重复时 $\mathbf{p}_1$ 和 $\mathbf{p}_3$ 都被 $\mathbf{w}_2^{2:1}$ 编码。




(a) 第 1 轮　　(b) 第 2 轮

图 16-13　本例的 ART1 重复

然而，在第 5 次重复时，当 $\mathbf{p}_4$ 被提交后，$\mathbf{w}_2^{2:1}$ 改变为包含了 $\mathbf{p}_4$。这个新的 $\mathbf{w}_2^{2:1}$ 不再提供与 $\mathbf{p}_1$ 和 $\mathbf{p}_3$ 的精确匹配，正如我们在重复 6 和 8 中看到的。这就要求它们接受神经元 3，而它在第一次轮回中并未用到。

通过修改警戒参数可以改变算法的结果。要多小的警戒值才能使仅有两个神经元的第二层，能按要求对 4 个输入向量进行编码呢？要多大的警戒值就会使第二层不得不增加第 4 个神经元呢？ 16-44

## 16.5 结束语

竞争性学习，以及其他许多类型的神经网络训练算法，都遇到了一个被称为“稳定性/可塑性二难问题”的难题。如果一个学习算法对新输入很敏感(可塑性强)，那么它也处于忘记以前学习内容的危险中(不稳定)。ART 网络被设计成既保持对新输入的敏感性，又保证学习的稳定性。

在这一章里，ART1 网络被用来说明自适应谐振理论的主要概念。ART1 网络是建立在第 15 章 Grossberg 竞争网络基础上的，只有少量修改。ART1 网络的主要改进是“期望值”的使用。当每一个输入模式提交给网络时，它会与匹配程度最接近的原型向量(期望值)进行比较。如果原型与输入向量不足以匹配，一个新的原型就会被选中。用这种方式，前面学习的记忆(原型)就不会被新的学习所破坏。

分析 ART 网络时一个应铭记在心的重要点，是它们被设计成似乎是合理的生物学上的学习机构。它们在理解人脑怎样工作方面，与令人鼓舞的实际模式识别系统非常相近。基于这种原因，这些学习机制要求在每个神经元只使用本地信息。而我们在文中讨论的学习规则，并非全部都是这样。

虽然 ART 网络解决了学习不稳定的困难，在它里面网络权值从来是不稳定的，这是我们尚未讨论过的另外一种稳定性问题。那就是实现网络短期记忆方程的微分方程的稳定性。例如，在第二层中，我们有一个具有非线性反馈的微分方程组。我们能否做出关于这种系统

的稳定性的一般说明呢？第 17 章将对这个问题进行广泛的讨论。 16-45

## 参考文献

[GaGr87a] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine." *Computer Vision, Graphics, and Image Processing*, vol. 37, pp. 54 - 115, 1987.

这篇文章最初提出了 ART1 结构。该文展示了这种结构对于任意多个二进制输入模式是自组织的和自稳定的。

[GaGr87b] G. A. Carpenter and S. Grossberg, "ART2: Self-organization of stable category recognition codes for analog input patterns," *Applied Optics*, vol. 26, no. 23, pp. 4019 - 4930, 1987.

为了能解决近似的输入模式，这篇文章提出了对 ART1 结构的扩展。

[GaGr90] G. A. Carpenter and S. Grossberg, "ART3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures," *Neural Networks*, vol. 3, no. 23, pp. 129 - 152, 1990.

这篇文章介绍了如何通过化学传输装置将 ART 网络的调整子系统运用到生物神经元上。

[GaGrMa92] G. A. Carpenter, S. Grossberg, N. Markuzon, J. Reynolds and D. Rosen, "Fuzzy ARTMAP: An adaptive resonance architecture for incremental learning of analog maps," *Proceedings of the International Joint Conference on Neural Networks*, Baltimore, MD, vol. 3, no. 5, pp. 309 - 314, 1992 .

为了能在噪声环境下获得更好的性能，作者们提出了 ARTMAP 结构的修改，包括使用模糊逻辑。

[GaGrRe91] G. A. Carpenter, S. Grossberg and J. Reynolds, " ARTMAP: Supervised real-time learning and classification of nonstationary data by a self - organizing neural network," *Neural Networks*, vol. 4, no. 5, pp. 169 - 181, 1991.

该文提出了一种有监督学习的自适应谐振网络。网络包括两个互联的 ART 模块。一个模块接受输入向量，另一个模块接受期望的输出向量。 16-46

[Gros76] S. Grossberg, "Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors," *Biological Cybernetics*, vol. 23, pp. 121 - 134, 1976.

描述了一种在视觉皮层生理学进展影响下的时间连续竞争网络。这种网络结构形成了其他一些重要网络的基础。

[Gros82] S. Grossberg, *Studies of Mind and Brain*, Boston: D. Reidel Publishing Co., 1982.

这本书收集了 Stephen Grossberg 从 1968 年到 1980 年的论文。其中有很多基本概念在其后的 Grossberg 网络中得到了运用，如自适应谐振理论网络。 16-47

## 习题

**E16.1** 考虑 ART1 网络的第一层，$\varepsilon = 0.02$。设第二层中有两个神经元，输入向量中有

两个元素，并且有如下的权值矩阵和输入：

$$\mathbf{W}^{2:1} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \quad \mathbf{p} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

再假设第二层神经元 2 是活跃的。

16-48 (i) 求出并描绘 $\mathbf{n}^1$ 的响应，如果 ${}^+b^1 = 2$ 且 ${}^-b^1 = 3$。

(ii) 求出并描绘响应 $\mathbf{n}^1$，如果 ${}^+b^1 = 4$ 且 ${}^-b^1 = 5$。

(iii) 求出并描绘响应 $\mathbf{n}^1$，如果 ${}^+b^1 = 4$ 且 ${}^-b^1 = 4$。

(iv) 检查(i)～(iii)小题的答案是否满足等式(16.21)预测的稳态响应。解释所有不一致的地方。

(v) 通过写出模仿 ART1 网络第一层的 MATLAB M-文件检验(i)～(iii)小题的答案。利用例行程序 **ode45**。画出每种情况的响应图。

**E16.2** 考虑具有如下参数的 ART1 网络第二层：

$$\varepsilon = 0.1 \quad \mathbf{W}^{1:2} = \begin{bmatrix} ({}_1\mathbf{w}^{1:2})^T \\ ({}_2\mathbf{w}^{1:2})^T \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & \frac{2}{3} \\ 1 & 0 \end{bmatrix}$$

且

$$f^2(n) = \begin{cases} 10(n)^2, & n \geq 0 \\ 0, & n < 0 \end{cases}$$

设第一层的输出为

$$\mathbf{a}^1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

(i) 写出第二层的运算方程，模仿并画出使用下列偏置值向量的响应图：

$${}^+\mathbf{b}^2 = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \quad {}^-\mathbf{b}^2 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

(ii) 重复(i)小题，使用如下偏置值向量：

$${}^+\mathbf{b}^2 = \begin{bmatrix} 3 \\ 3 \end{bmatrix} \quad {}^-\mathbf{b}^2 = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

(iii) 重复(i)小题，使用如下偏置值向量：

$${}^+\mathbf{b}^2 = \begin{bmatrix} 3 \\ 3 \end{bmatrix} \quad {}^-\mathbf{b}^2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

(iv) 前面三小题的结果满足等式(16.37)描述的稳态响应吗？如果不满足，为什么？

**E16.3** 考虑 ART1 网络具有如下参数的调整子系统：

$$\varepsilon = 0.1 \quad {}^+b^0 = {}^-b^0 = 2$$

调整子系统的输入为

$$\mathbf{p} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \mathbf{a}^1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

(i) 求出并描绘调整子系统的响应 $n^0(t)$，其中 $\alpha = 0.5$，$\beta = 4$ $(\rho = 0.125)$。

(ii) 求出并描绘调整子系统的响应 $n^0(t)$，其中 $\alpha = 0.5$，$\beta = 2$ $(\rho = 0.25)$。

(iii) 验证小题(i)和(ii)满足稳定状态条件。

(iv) 通过写出模仿调整子系统的 MATLAB M-文件检验第(i)，(ii)小题的答案。 16-49

**E16.4** 为了得到 L1 - L2 和 L2 - L1 学习规则的稳定状态条件，我们假设输入模式和神经元的输出在权值矩阵收敛前保持恒定不变。这叫做“快速学习”。说明这个快速学习假设与出现在第 13 章的 instar 和 outstar 学习规则中和第 14 章的 Kohonen 竞争性学习规则中把学习速度设置为 $\alpha = 1$ 是等价的。

**E16.5** 用下面的输入向量训练 ART1 网络：

$$\mathbf{p}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad \mathbf{p}_3 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{p}_4 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

使用参数 $\zeta = 2$，并选择 $S^2 = 3$(3 个分类)。

(i) 利用 $\rho = 0.3$ 训练网络函收敛。

(ii) 利用 $\rho = 0.6$ 重复(i)小题。

(iii) 利用 $\rho = 0.9$ 重复(ii)小题。

**E16.6** 当原型与输入模式之间不存在精确匹配时，可以修改 ART1 算法使第二层增加一个新的神经元。这将导致在矩阵 $\mathbf{W}^{1:2}$ 中新增一行和在 $\mathbf{W}^{2:1}$ 中新增一列。描述此过程怎样实现。

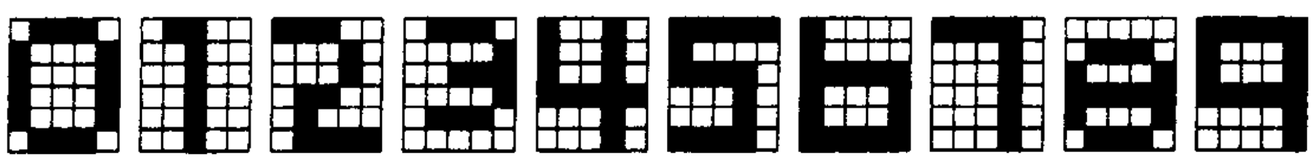
**E16.7** 写出实现 ART1 算法的 MATLAB M-文件(运用习题 E16.6 中所描述的修改过程)。用这个 M-文件训练 ART1 网络，并使用下面的输入向量(参见例题 P16.7)：

$\mathbf{p}_1$　$\mathbf{p}_2$　$\mathbf{p}_3$　$\mathbf{p}_4$

图 16-14

以下面的次序提交输入向量：$\mathbf{p}_1 - \mathbf{p}_2 - \mathbf{p}_3 - \mathbf{p}_1 - \mathbf{p}_4$(即是说，$\mathbf{P}_1$ 在一个轮回中被提交两次)。使用参数 $\zeta = 2$，$\rho = 0.9$，并选择 $S^2 = 3$(3 个分类)。训练网络直到权值收敛。将你的结果与 P16.7 比较。 16-50

**E16.8** 回忆第 7 章描述的数字识别问题。使用数字 0 ~ 9 训练 ART1 网络，它们显示如下：



$\mathbf{p}_1$　$\mathbf{p}_2$　$\mathbf{p}_3$　$\mathbf{p}_4$　$\mathbf{p}_5$　$\mathbf{p}_6$　$\mathbf{p}_7$　$\mathbf{p}_8$　$\mathbf{p}_9$　$\mathbf{p}_{10}$

图 16-15

使用参数 $\zeta = 2$，选择 $S^2 = 5$(5 个分类)。利用习题 E16.7 中的 MATLAB M-文件。

(i) 训练网络至收敛，使用 $\rho = 0.3$。

(ii) 训练网络至收敛，使用 $\rho = 0.6$。

(iii) 训练网络至收敛，使用 $\rho = 0.9$。

(iv) 讨论小题(i) ~ (iii)的结果。解释警戒参数的影响。 16-51

# 第17章 稳 定 性

## 17.1 目的

递归网络中的“收敛性”问题第一次出现在第 3 章讨论的 Hopfield 网络中。注意到递归网络的输出可能收敛于一个稳定点、发生振荡或者甚至可能发散。急剧下降过程和 LMS 算法的“稳定性”已分别在第 9 章和第 10 章中进行了讨论。Grossberg 的持续时间递归网络(continuous - time recurrent network)的稳定性问题也在第 15 章作了讨论。

这一章将对稳定性作更为细致的定义。我们的目的是判断一组特定的非线性方程是否具有一些其输出收敛的点(或轨迹)。为研究这个问题题，我们将介绍 Lyapunov 的稳定性定理，并且把它运用到一个简单却又具有启发性的问题中。然后，提出 Lyapunov 理论的一般形式：LaSalle 不变性定理。这将为第 18 章打下基础，那里 LaSalle 定理被用来证明 Hopfield 网络的稳定性。

17-1

## 17.2 理论和实例

### 17.2.1 递归网络

本书最初讨论递归神经网络，是在第 3 章讨论 Hamming 以及 Hopfield 网络时，它们具有从输出到输入的反馈连接。第 15 章和第 16 章的 Grossberg 网络也含有递归连接。由于递归网络能够识别和回忆时序模式以及空间模式，因而它比前馈网络更具有潜在的能力。然而，这些递归网络的行为比前馈网络更为复杂。

对前馈网络来说，其输出是恒定的(对一个固定的输入)，并且仅是网络输入的函数。但是，对递归网络来说，网络的输出是时间的一个函数。对一个给定的输入和一个给定的初始网络输出，网络的响应可能收敛到一个稳定的输出。然而，它也可能振荡，无限地增大，或者遵循一种混乱的模式。在这一章的剩下部分，我们旨在分析一般的非线性递归网络，用以确定它们的长期行为。

考虑由如下形式的非线性微分方程组描述的递归网络：

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{a}(t) = \mathbf{g}(\mathbf{a}(t),\mathbf{p}(t),t) \tag{17.1}$$

17-2

这里 $\mathbf{p}(t)$是网络输入，$\mathbf{a}(t)$是网络输出(见图 17-1)。

我们希望知道这些系统在稳定状态下如何运作。我们最感兴趣的是网络收敛到一个恒定输出的那些情况，这个恒定输出即为稳定平衡点。一个非线性系统可能有许多稳定点。对于某些神经网络，这些稳定点代表存储的原型模式。可能的话，我们想知道这些稳定点在哪里，以及哪些初始条件 $\mathbf{a}(0)$会敛到一个给定的稳定点。(即什么是一个给定稳定点的吸引区?)

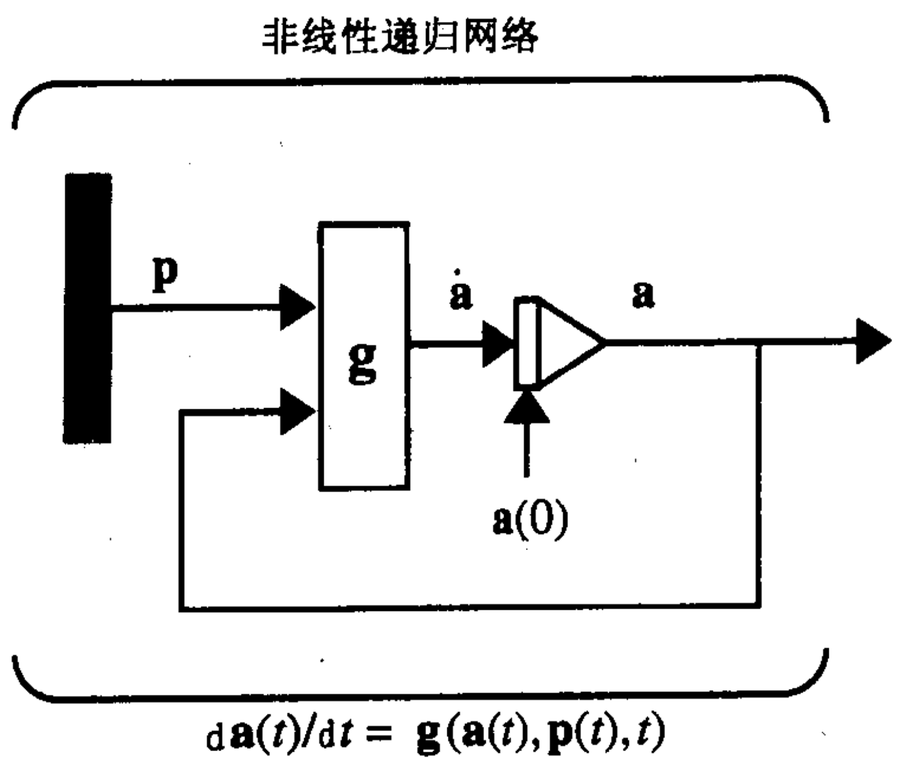



图 17-1 非线性持续时间递归网络

### 17.2.2 稳定性概念

开始讨论前，我们用一个简单而又直观的例子来介绍一些基本的稳定性概念。考虑重力场中一个球形轴承的运动(带有摩擦消耗)。在图 17-2(a)中，槽的底部(点 $\mathbf{a}^*$)放有一个球形轴承。如果我们将轴承移到一个不同的位置，它将会在槽里前后振荡起来。但是，由于摩擦力，它最终将回到槽的底部，我们称这个位置为*渐近稳定点*(asymptotically stable point)，对它将在下一小节中精确地定义 。

现在看图 17-2(b)。在一个平坦表面的中心放置了一个球形轴承。如果我们将轴承移到一个不同的位置，它不会发生运动。既然轴承移走后没有回到原来的位置，因而该表面中心的那个位置不是渐近稳定点。然而，从某种意义上说它又是稳定的，因为至少小球没有离中心点越滚越远。我们称这种点为 Lyapunov *意义上的稳定*，这将在下一小节中定义。

现在考虑图 17-2(c)。球被放在一个小山的顶部。这是一个平衡位置，事实上，只要我们小心地放置小球，它会保持在小山顶部。然而，一旦小球受到一个轻微的干扰，它就会滚下山。这是一个*不稳定的平衡点*。

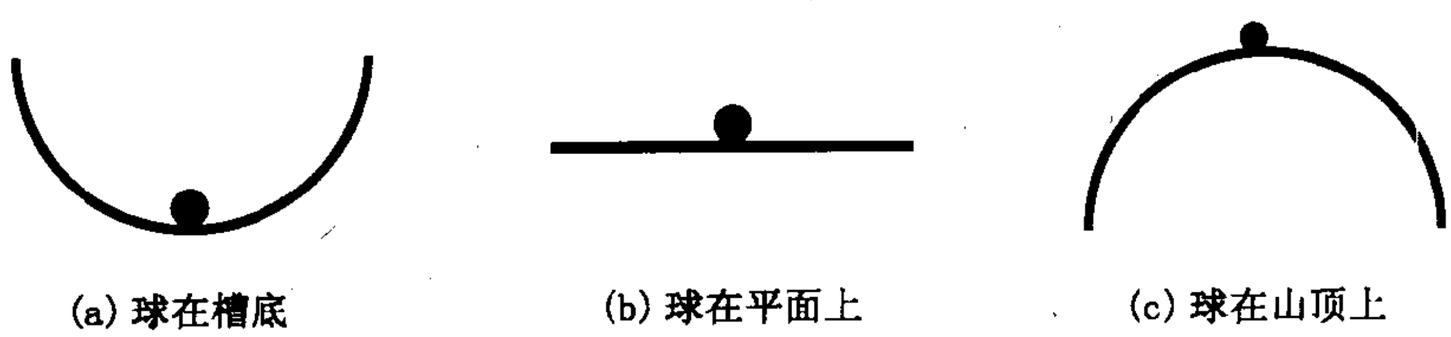



图 17-2

在下一章，我们试图设计 Hopfield 神经网络，它里面存储的原型模式将是一些渐近稳定的平衡点。我们也希望这些稳定点的吸引区尽可能地大。

举个例子，考虑图 17-3。我们希望设计像情况 A 那样拥有很大吸引区的神经网络。一个自然的想法是一个以很大摩擦力滚动的小球(初速为 0)被放在情况 A 的任一个槽区里，它都将留在槽区内并最终找到到达底部(稳定点)的路径。然而，情况 B就相对复杂了。例如，如果带摩擦力的小球处于 P 点的位置，那么不能确定最终是哪一个稳定点会捕获小球。小

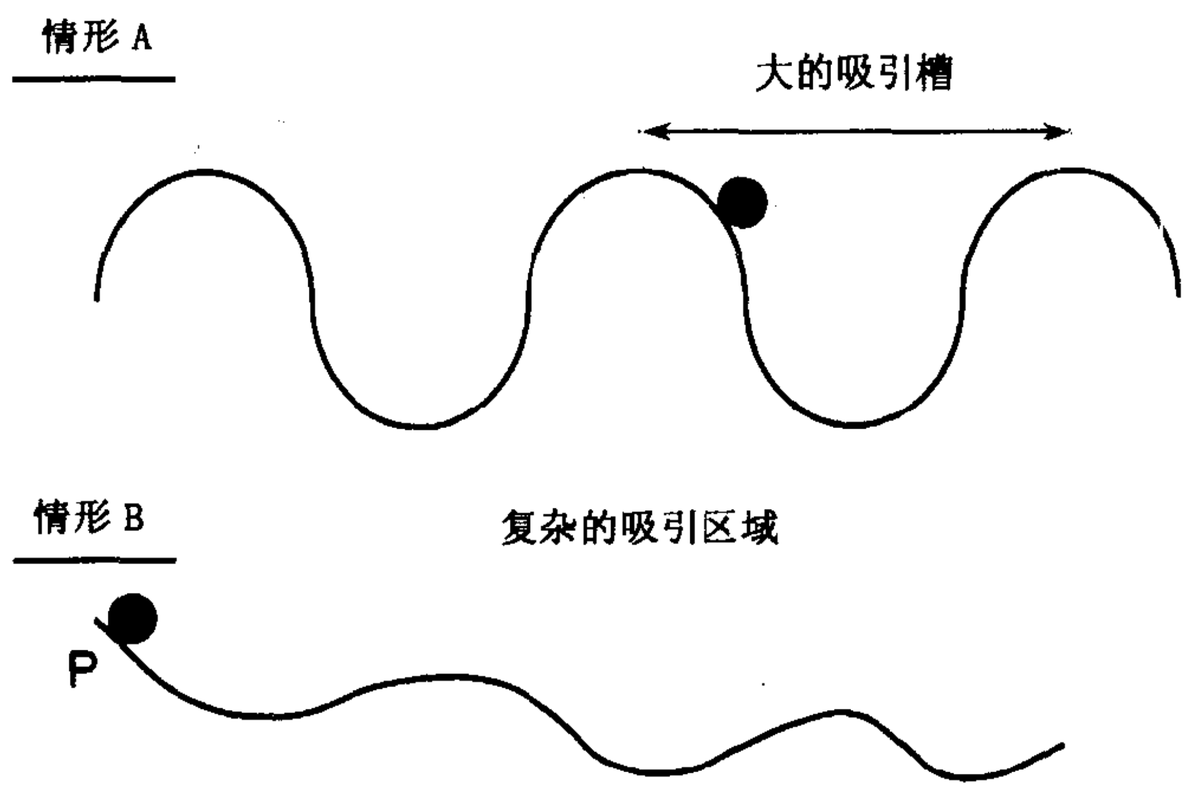



图 17-3 吸引槽

球可能并不在最靠近 P 点的稳定点静止下来。要指出某个特定的稳定点的吸引区有多大也同样困难。

17-3

至此我们已经提出了一个稳定性的直观概念。在本章剩下的部分我们将对它们进行严密的数学推导。

定义

**平衡点** 我们对上一小节讨论的各种类型的稳定性给出专门的数学定义。在这些定义中将讨论平衡点的稳定性。一个平衡点 $\mathbf{a}^*$ 是指使等式(17.1)中导数为 0 的点。为简化问题，我们将特别讨论点 $\mathbf{a}^*=\mathbf{0}$，这个点被称为原点。这一限制并不影响讨论的一般性。

**定义 1** 稳定性(在 Lyapunov 的意义下)

一个原点是稳定的平衡点，如果对于任意给定的值 $\varepsilon>0$，总存在一个数 $\delta(\varepsilon)>0$，使得当 $\|\mathbf{a}(0)\|<\delta$ 时产生的运动 $\mathbf{a}(t)$ 对于 $t>0$ 满足 $\|\mathbf{a}(t)\|<\varepsilon$。

这个定义说明，只要一个系统的输出最初接近一个稳定点，那么它就不会运动到离稳定点太远。讨论的问题是：希望系统的输出保持在距离原点不超过 $\varepsilon$ 的范围内。如果该原点是稳定的，那么总能找到一个距离 $\delta$(可能是 $\varepsilon$ 的函数)，若系统在时间 $t=0$ 时的输出落在离原点 $\delta$ 的范围内，那么它就将总是落在离原点 $\varepsilon$ 的范围内。右图(图 17-2(b))中小球(初速为 0)的位置在 Lyapunov 意义上是稳定的，只要小球会受到摩擦力的作用。如果小球不受到摩擦力影响，那么任意一个初速度都会产生小球运动轨迹 $\mathbf{a}(t)$，其位置永远不固定。(此例中，向量 $\mathbf{a}(t)$ 应由小球的位置和速度构成。)

17-4

下面，让我们来考虑一个更强的渐近稳定性概念。

**定义 2** 渐近稳定性

一个原点是一个渐近稳定的平衡点，如果存在一个值 $\delta>0$，只要 $\|\mathbf{a}(0)\|<\delta$，产生的运动在 $t\to\infty$ 时满足 $\|\mathbf{a}(t)\|\to 0$。

这是稳定性的一个较强的定义。它说的是只要系统的输出最初是在离稳定点距离为 $\delta$ 范围之内，那么输出就最终收敛于稳定点。在右图(图 17-2(a))中，小球(初速为 0)的位置是一个渐近稳定点，只要小球会受摩擦力影响。如果没有摩擦力，这个位置就只是Lyapunov意

义上的稳定点。

我们愿意建立一个拥有很多特定的渐近稳定点的神经网络，它们中的每一个代表一个原型模式。这即是我们将在第18章建立Hopfield网络的设计目标。

除了稳定性的定义，在分析稳定性时我们还会用到另一个概念。那就是定函数(definite function)的概念。下面两个定义将阐明这个概念。

**定义3** 正定

一个标量函数 $V(\mathbf{a})$，当 $V(\mathbf{0})=0$ 且 $V(\mathbf{a})>0(\mathbf{a}\neq 0)$ 时，称为正定的。

**定义4** 半正定

一个标量函数 $V(\mathbf{a})$，当 $V(\mathbf{a})\geqslant 0$ (对于所有的 $\mathbf{a}$)时，称为半正定的。

(这些定义可做适当修改用来定义负定和半负定。)现在我们已定义了稳定性，让我们来考虑一个测试稳定性的方法。

### 17.2.3 Lyapunov稳定性定理

一个最重要的研究非线性系统稳定性的途径之一，是俄罗斯数学家Alexandr Mikhailovich Lyapunov介绍的理论。虽然他的主要著作早在1892年首次出版，但是直到很久以后才引起俄罗斯国外学者的注意。在这一节我们将讨论Lyapunov的一个最强有力的关于稳定性分析的技术——直接法。 17-5

考虑一个自主(无外力，不明显依赖于时间)系统：

$$\frac{\mathrm{d}\mathbf{a}}{\mathrm{d}t}=\mathbf{g}(\mathbf{a}) \tag{17.2}$$

Lyapunov稳定性定理现在可表述如下：

**定理1** Lyapunov稳定性定理

如果能够找到一个正定函数 $V(\mathbf{a})$，使得 $\mathrm{d}V(\mathbf{a})/\mathrm{d}t$ 是半负定的，那么对于方程(17.2)所示系统，原点($\mathbf{a}=\mathbf{0}$)是稳定的。如果能够找到一个正定函数 $V(\mathbf{a})$，使得 $\mathrm{d}V(\mathbf{a})/\mathrm{d}t$ 是一个负定函数，那么原点($\mathbf{a}=\mathbf{0}$)是渐近稳定的。在这种情况下，$V$ 被称为系统的Lyapunov函数。

你可以把 $V(\mathbf{a})$ 看作一般的能量函数。该定理要表明这样一个概念：如果一个系统的能量在持续减小($\mathrm{d}V(\mathbf{a})/\mathrm{d}t$ 负定)，那么它将最终处于某个最小能量状态。Lyapunov的观点使能量的概念一般化，因而该定理可被应用到能量难以表达或没有意义的系统中。

我们应该注意，该定理仅仅说明如果能找到一个合适的Lyapunov函数 $V(\mathbf{a})$，系统就是稳定的。它并没有告诉我们一个不能找到这样的函数的系统的稳定性信息。

### 17.2.4 单摆例子

可以通过将Lyapunov的稳定性定理应用到一个简单的机械系统来领会它。这个系统非常简单，并且其操作容易可视化，它还可以解释在下一章应用到神经网络设计中的一些重要概念。该示例系统即为如图17-4所示的单摆。 17-6

利用牛顿第二定律($F=ma$)，写出单摆的运动方程如下：

$$ml\frac{\mathrm{d}^2}{\mathrm{d}t^2}(\theta)=-c\frac{\mathrm{d}\theta}{\mathrm{d}t}-mg\sin(\theta) \tag{17.3}$$

或者

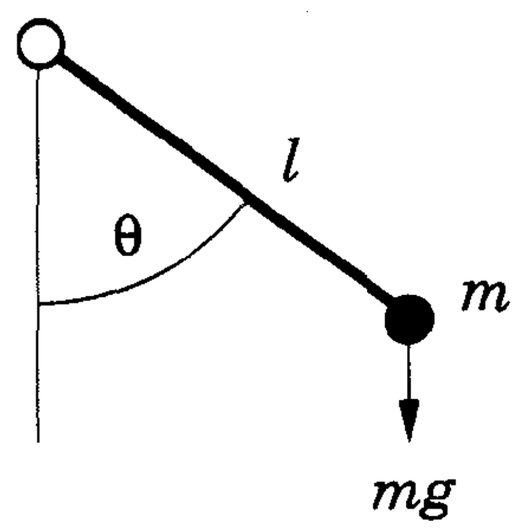



图 17-4 单摆

$$ml\frac{d^2\theta}{dt^2} + c\frac{d\theta}{dt} + mg\sin(\theta) = 0 \tag{17.4}$$

其中 $\theta$ 为摆角，$m$ 是单摆的质量，$l$ 是摆长，$c$ 是阻尼系数，$g$ 是引力常量。

方程(17.3)中等号右边第一项是阻力，它与单摆的速度成正比。正是这一项代表了该系统中的能量消耗。方程右边第二项是重力，它与摆角的正弦成正比。其值在单摆垂直时为0，在单摆水平时最大。

当阻尼系数不为0时，单摆会最终停挂在垂直的位置。这时可以看作 $\theta=0$，而更一般的是 $\theta=2\pi n$，其中 $n=0$，$\pm1$，$\pm2$，$\pm3$，…。也就是说，给定某个合适的初始条件，单摆可以处于 $\theta=0$ 的状态，或者它可能转一圈到达 $\theta=2\pi$ 状态，等等。它有很多平衡状态。(在位置 $\theta=\pi n$ 处，对于奇数 $n$，这些位置是平衡点，但不稳定。)

为了分析该系统的稳定性，将以状态变量的形式写出单摆方程，它们将表现为一对一阶微分方程。选择如下的状态变量：

$$a_1 = \theta, a_2 = \frac{d\theta}{dt} \tag{17.5}$$

以这些状态变量的形式写出单摆方程如下：

$$\frac{da_1}{dt} = a_2 \tag{17.6}$$

17-7

$$\frac{da_2}{dt} = -\frac{g}{l}\sin(a_1) - \frac{c}{ml}a_2 \tag{17.7}$$

现在来考查该单摆系统原点($\mathbf{a}=\mathbf{0}$)的稳定性。(原点对应于摆角为0和摆速为0的状态。)首先检查原点是一个平衡点。将 $\mathbf{a}=\mathbf{0}$ 代入状态方程：

$$\frac{da_1}{dt} = a_2 = 0 \tag{17.8}$$

$$\frac{da_2}{dt} = -\frac{g}{l}\sin(a_1) - \frac{c}{ml}a_2 = -\frac{g}{l}\sin(0) - \frac{c}{ml}(0) = 0 \tag{17.9}$$

既然导数为0，故原点是一个平衡点。

然后要找出单摆的 Lyapunov 函数。在本例中，将用系统的能量作为 Lyapunov 函数 $V$。为得到单摆的总能量，将其动能和势能相加：

$$V(\mathbf{a}) = \frac{1}{2}ml^2(a_2)^2 + mgl(1-\cos(a_1)) \tag{17.10}$$

为了测试系统稳定性，将 $V$ 对时间求导：

$$\frac{\mathrm{d}}{\mathrm{d}t}V(\mathbf{a}) = [\nabla V(\mathbf{a})]^T\mathbf{g}(\mathbf{a}) = \frac{\partial V}{\partial a_1}\left(\frac{\mathrm{d}a_1}{\mathrm{d}t}\right) + \frac{\partial V}{\partial a_2}\left(\frac{\mathrm{d}a_2}{\mathrm{d}t}\right) \tag{17.11}$$

$V(\mathbf{a})$的偏导数可从等式(17.10)中求得，两个状态变量的导数已在等式(17.6)和(17.7)中给出。于是有

$$\frac{\mathrm{d}}{\mathrm{d}t}V(\mathbf{a}) = (mgl\ \sin(a_1))a_2 + (ml^2a_2)\left(-\frac{g}{l}\sin(a_1) - \frac{c}{ml}a_2\right) \tag{17.12}$$

将项$(mgl\ \sin(a_1))a_2$取消，只剩下

$$\frac{\mathrm{d}}{\mathrm{d}t}V(\mathbf{a}) = -cl(a_2)^2 \leqslant 0 \tag{17.13}$$

为了证明原点($\mathbf{a}=\mathbf{0}$)是渐近稳定的，必须证明该导数是负定的。在原点，该导数为0，同时只要 $a_2=0$，无论 $a_1$ 为什么值它也为0。这样，$\mathrm{d}V(\mathbf{a})/\mathrm{d}t$ 是半负定的，而不是负定的。于是由 Lyapunov 定理知，原点是一个稳定点。但是，不能说根据定理和这个 Lyapunov 函数，该原点是渐近稳定的。 17-8

在这种情况下我们知道，只要单摆有摩擦，它就将最终停在垂直位置，因此，原点确是渐近稳定的。但是，Lyapunov 定理，利用 Lyapunov 函数，却只能告诉我们原点是稳定的。要证明原点是渐近稳定的，需要改进 Lyapunov 定理为 LaSalle 不变性定理。LaSalle 不变性定理将在下一小节讨论。

首先，让我们用特殊的数据实例来进一步研究单摆。令 $g=9.8$，$m=1$，$l=9.8$，$c=1.96$。重写单摆状态方程如下：

$$\frac{\mathrm{d}a_1}{\mathrm{d}t} = a_2 \tag{17.14}$$

$$\frac{\mathrm{d}a_2}{\mathrm{d}t} = -\sin(a_1) - 0.2a_2 \tag{17.15}$$

求出 $V$ 和它的导数如下：

$$V = (9.8)^2\left[\frac{1}{2}(a_2)^2 + (1-\cos(a_1))\right] \tag{17.16}$$

$$\frac{\mathrm{d}V}{\mathrm{d}t} = -(19.208)(a_2)^2 \tag{17.17}$$

注意，对任意的 $a_1$ 值，只要 $a_2=0$，即有 $\mathrm{d}V/\mathrm{d}t=0$。

图 17-5 显示了当摆角变化范围为 $-10$ 到 $+10$ 弧度，角速度变化范围为 $-2$到 $+2$弧度每秒时，能量曲面 $V$ 的三维图和等值图。注意在这个范围内能量曲面有三个可能的最小点，在0和 $\pm 2\pi$。

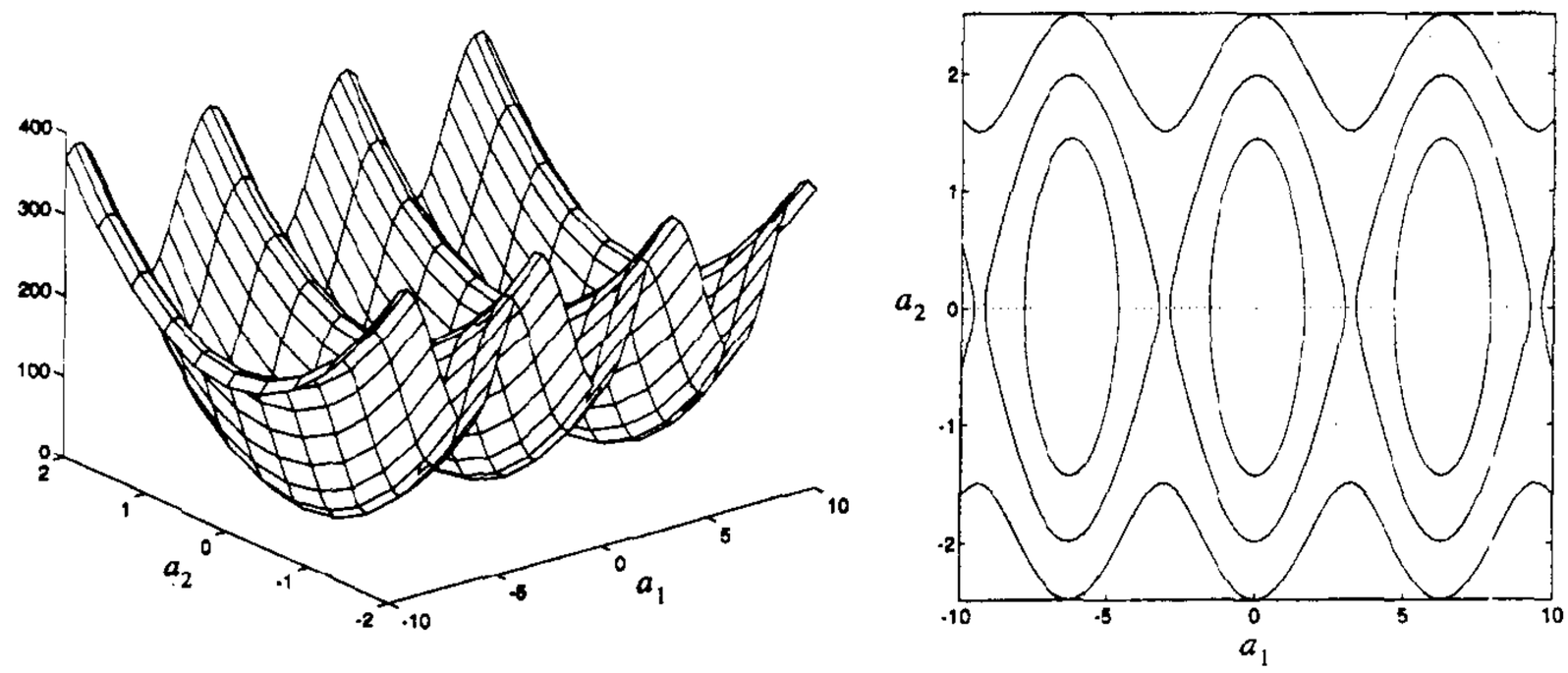



图 17-5 单摆能量曲面

17-9

(我们将在第 18 章发现 Lyapunov 函数的最小点对应于一个自相关神经网络的原型模式。单摆系统，正如递归神经网络，有许多最小点。)

当然，从能量图 17-6 中不知道单摆以什么方式或者由什么路线找到了特殊的能量最小点。为了反映这个情况，在图 17-5 中我们画了一个能量等值图，上面有一条单摆的特殊路径。这条蓝色的响应轨迹，从 1.3 弧度(74°)的初始位置 $a_1(0)$和 1.3 弧度每秒的初始速度 $a_2(0)$开始。轨迹收敛于平衡点 $\mathbf{a}=\mathbf{0}$。

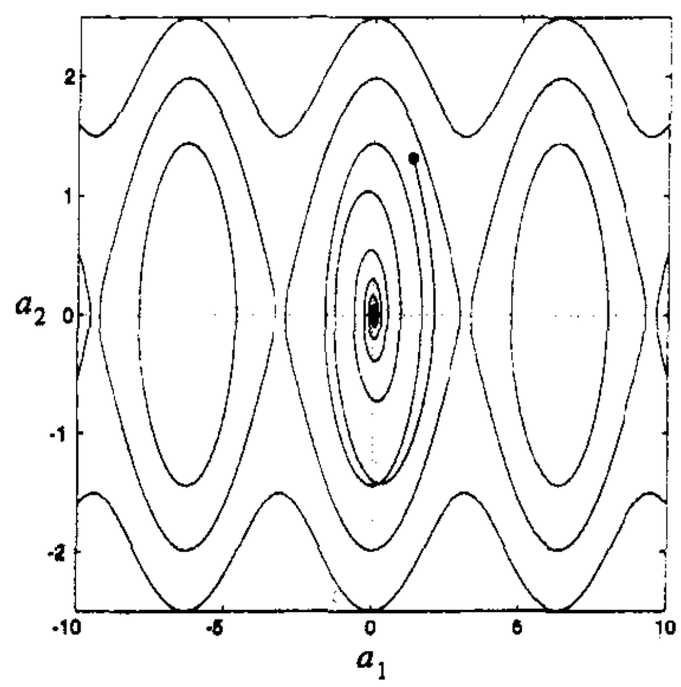



图 17-6 状态变量平面的单摆响应

两个状态变量的时间响应如图 17-7 所示。请注意，因为初速度是正的，故单摆一开始不停运动。(察看是否与图 17-6 一致。)在下落前它到达大约为 2 弧度的最大角。振荡将持续衰减至两个状态变量都收敛于 0。

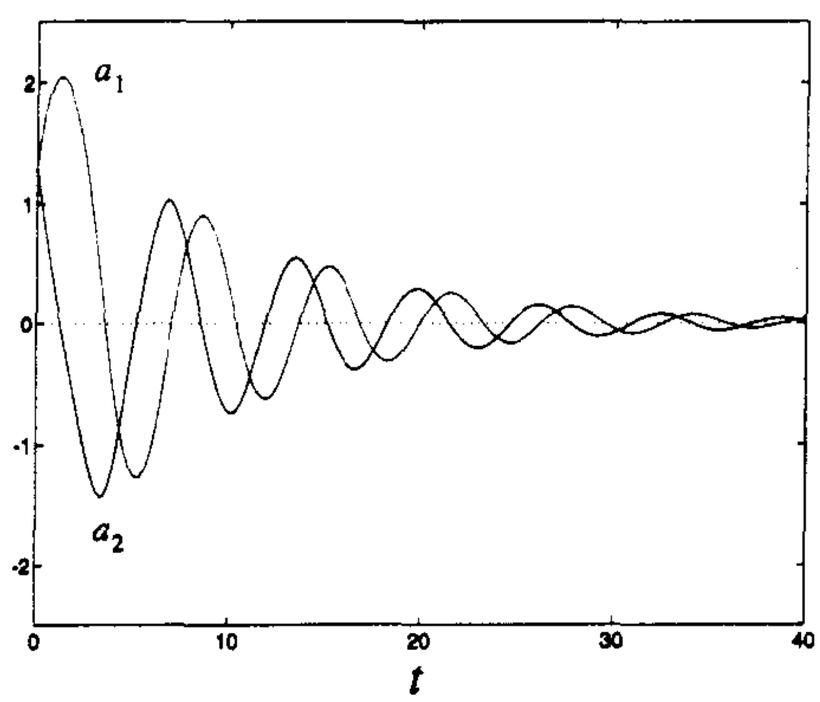



图 17-7 状态变量 $a_1$ 与 $a_2$ 的时间图

在本例中，两个状态变量都收敛于 0。但是，以后会看到，这并不是惟一可能的平衡点。

画出如图 17-8 的单摆能量($V$)图也比较有意思。回忆方程(17.17)能量永远不会增加，这与图 17-8 一致。方程(17.17)还预测能量曲线的导数只有当速度 $a_2$ 为 0 时才会为 0。如果比较一下图 17-8 和图 17-6，这也得到了验证。每一次 $a_2$ 的图形穿过零轴，能量曲线的倾斜率即为 0。

17-10

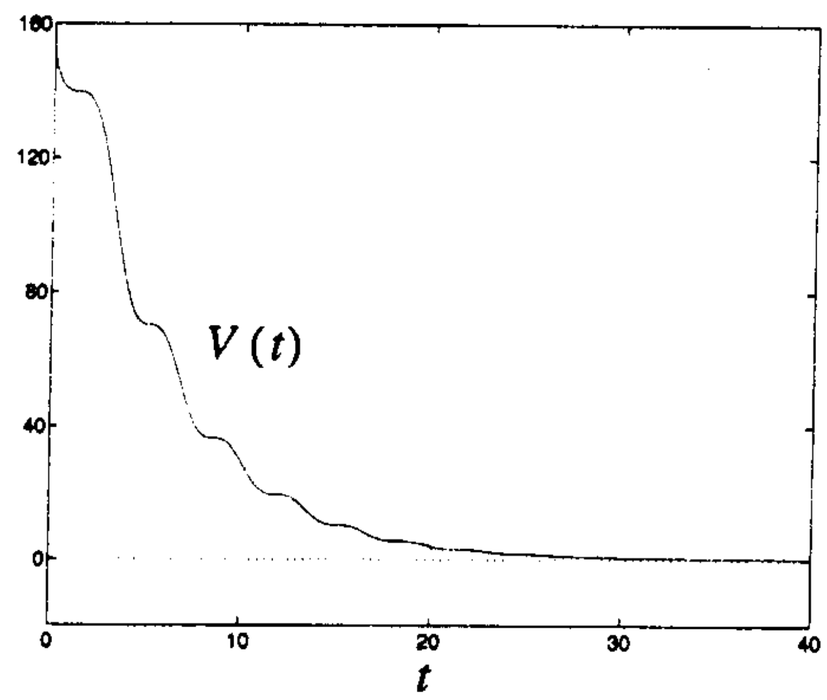



图 17-8 单摆 Lyapunov 函数(能量)的时间图

请注意，虽然有许多点能量曲线的导数为 0，但导数并未保持为 0 直到能量本身也为 0。这个观察将导出 LaSalle 不变性定理，在下一小节会讨论到。该定理的主要思想是确定那些 Lyapunov 函数导数为 0 的点，并判断系统是否会陷入那些点。(那些可能使一条轨迹陷入的地方叫做不变集。)如果只有一个点，能使轨迹陷入，并且具有零导数，又是原点，则这个原点即为渐近稳定的。

这一小节以图的形式展示了基于两个状态变量的初始条件的特殊单摆行为。对于初始条件的不同选择将会导致图中完全不同的结果。我们将在下一小节对这一点作展开讨论。 17-11

试验单摆请用 *Neural Network Design Demonstration Dynamic System* (**nnd17ds**)。

### 17.2.5 LaSalle 不变性定理

单摆的例子展示了 Lyapunov 定理的一个问题。我们找到了一个其导数仅为半负定(而不是负定)的 Lyapunov 函数，而且也知道原点在单摆系统中是渐近稳定的。在这一小节，我们将介绍一个阐明 Lyapunov 定理不确定性的定理。它定义了那些使 Lyapunov 函数的导数为 0 的状态空间中的区域，然后确定区域中能使轨迹发生陷入的那些部分。

在讨论 LaSalle 不变性定理之前，需要首先介绍下面的定义。

1. 定义

**定义 5** Lyapunov 函数

令 $V$ 是一个从$\Re^n$ 到$\Re$的连续可微函数。若 $G$ 是$\Re^n$ 的任一子集，称 $V$ 是系统 $\mathrm{d}\mathbf{a}/\mathrm{d}t = \mathbf{g}(\mathbf{a})$的一个 $G$ 上的 Lyapunov 函数，只要

$$\frac{\mathrm{d}V(\mathbf{a})}{\mathrm{d}t} = (\nabla V(\mathbf{a}))^T\mathbf{g}(\mathbf{a}) \tag{17.18}$$

在 $G$ 上不改变符号。

这是在定理 1 中用到的 Lyapunov 函数前述定义的推广，这里不要求函数是正定的。事实上，对函数本身并没有直接的要求(除了它是连续可微的)。惟一的要求是关于 $V$ 的导数。其导数在集合 $G$ 上任何地方都不改变符号。注意，如果导数是半负定的或半正定的就不会改变符号。

应留意这里仍没有解释怎样选择集合 $G$。我们将利用下面的定义和定理为一个给定的系统选择一个最好的 $G$。

**定义 6** 集合 $Z$

17-12

$$Z = \{\mathbf{a}: \mathrm{d}V(\mathbf{a})/\mathrm{d}t = 0,\ \mathbf{a}\text{在 } G \text{ 的闭包中}\} \tag{17.19}$$

这里"$G$ 的闭包"包括 $G$ 的内部和边界。这是一个关键集合。它包含所有使 Lyapunov 函数导数为 0 的点。以后将决定该集合中的哪些地方会使系统轨迹发生陷入。

**定义 7** 不变集

一个$\Re^n$ 中的点集合关于 $\mathrm{d}\mathbf{a}/\mathrm{d}t = \mathbf{g}(\mathbf{a})$是不变的，如果 $\mathrm{d}\mathbf{a}/\mathrm{d}t = \mathbf{g}(\mathbf{a})$的每一个解开始都在集合中并始终保持在该集合中。

如果一个系统进入了某个不变集，那么它就不能再离开。

**定义 8** 集合 $L$

$L$ 定义为 $Z$ 中的最大不变集。

该集合包括所有可能使解收敛的点。Lyapunov 函数在 $L$ 中不会改变(因为其导数为 0)，并且轨迹将会在 $L$ 中陷入(因为它是不变集)。现在，如果这个集合只有一个稳定点，那么那个点就是渐近稳定的。这即是 LaSalle 定理大体上将告诉我们的。

2. 定理

LaSalle 不变性定理是 Lyapunov 稳定性定理的扩展。在下一章我们将用它设计 Hopfield 网络。该定理推导参见[Lasa67]。

**定理 2** LaSalle 不变性定理

若 $V$ 是 $\mathrm{d}\mathbf{a}/\mathrm{d}t = \mathbf{g}(\mathbf{a})$在 $G$ 上的 Lyapunov 函数，那么对于所有 $t > 0$，保留在 $G$ 中的每一个解 $\mathbf{a}(t)$ 当 $t \to \infty$时，趋于 $L^\circ = L \cup \{\infty\}$。($G$ 具有所有的稳定点，是对 $L$ 的吸引区。)若所有的轨迹都有界的，则当时 $t \to \infty$时，$\mathbf{a}(t) \to L$。

若一条轨迹停留在 $G$ 中，那么它也将收敛于 $L$，或者趋于无限。如果所有的轨迹都是有界的，那么所有的轨迹都将收敛于 $L$。

有一个我们将广泛使用的 LaSalle 定理的推论，它涉及以一种特殊的方式选择集合 $G$ 的问题。

17-13

**推论 1** LaSalle 定理的推论

令 $G$ 包含于集合

$$\Omega_\eta = \{\mathbf{a}: V(\mathbf{a}) < \eta\} \tag{17.20}$$

中(作为一个连通的子集)。假设 $G$ 是有界的，在 $G$ 上 $\mathrm{d}V(\mathbf{a})/\mathrm{d}t \leqslant 0$，且令集合 $L^\circ = closure(L \cap G)$是 $G$ 的一个子集。那么 $L^\circ$是一个吸引子，而 $G$ 在它的吸引区内。

LaSalle 定理以及它的推论都是很有用的。它们不仅告诉了我们哪些点是稳定的($L^\circ$)，而且也给我们提供了部分的吸引区($G$)。(注意 $L^\circ$在推论中的定义与定理中的不一样。)

为阐明 LaSalle 不变性定理，让我们回到先前讨论的单摆例子中。

3. 例子

将推论 1 运用于单摆例子。第一步是要选择集合 $\Omega_\eta$，该集合将被用来选择集合 $G$($\Omega_\eta$的一部分)。

在本例中使用值 $\eta = 100$，因此 $\Omega_{100}$即为能量小于等于 100 的点组成的集合。

$$\Omega_{100} = \{\mathbf{a}: V(\mathbf{a}) \leqslant 100\} \tag{17.21}$$

该集合用黑色显示在图 17-9 中。

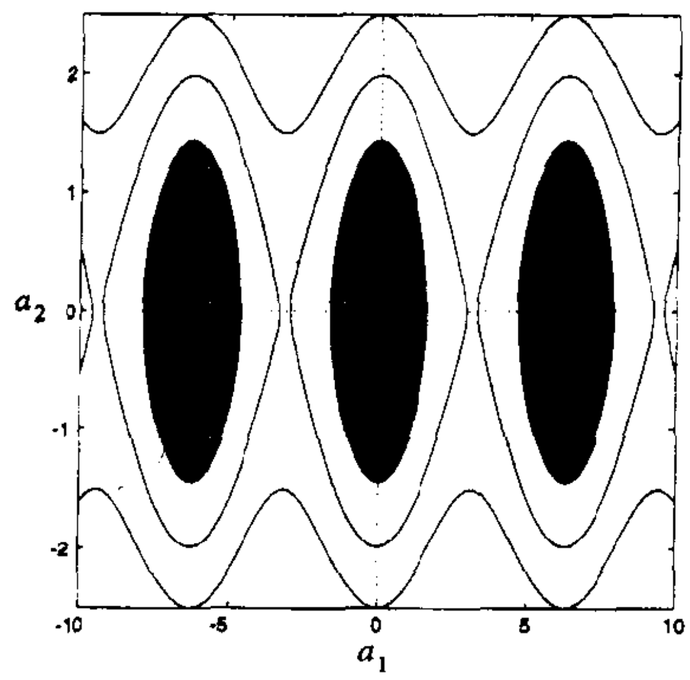



图 17-9 集合 $\Omega_{100}$的图示

分析的下一步是选择 $\Omega_{100}$的一个部分(连通的子集)作为集合 $G$。既然研究的是原点的稳定性，就选择包含 $\mathbf{a}=\mathbf{0}$的 $\Omega_{100}$的那个部分。该子集结果如图 17-10 所示。 17-14

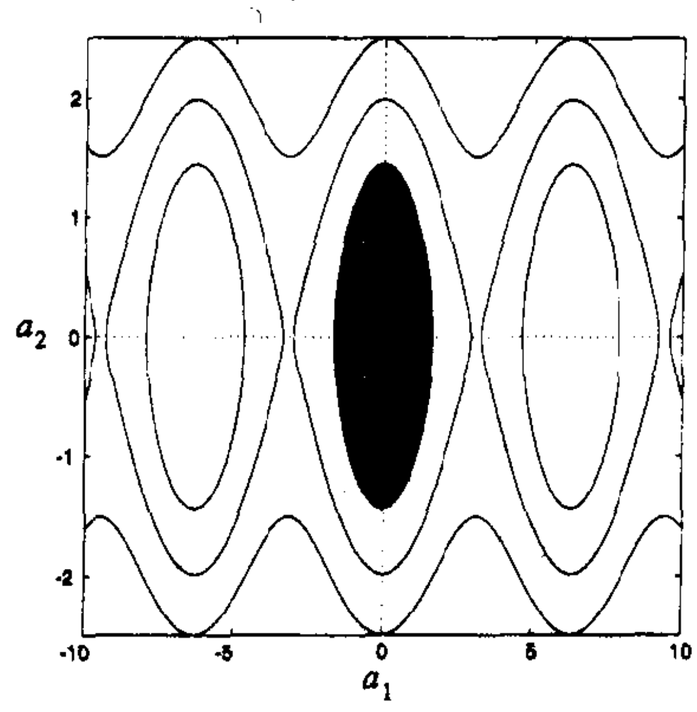



图 17-10 集合 $G$ 的图示

现在已选完了集合 $G$，下面需要检查 Lyapunov 函数的导数在 $G$ 上是否小于等于 0。由方程(17.17)知，$\mathrm{d}V(\mathbf{a})/\mathrm{d}t$ 是半负定的，因此它在 $G$ 上当然小于等于 0。

现在准备确定吸引子集合 $L^{\circ}$。由 $Z$ 中的最大不变集 $L$ 开始。

$$\begin{aligned} Z &= \{\mathbf{a}: \mathrm{d}V(\mathbf{a})/\mathrm{d}t = 0, \mathbf{a}\text{在 } G \text{ 中}\} \\ &= \{\mathbf{a}: a_2 = 0, \mathbf{a}\text{在 } G \text{ 中}\} \end{aligned} \tag{17.22}$$

这也可以写成

$$Z = \{\mathbf{a}: a_2 = 0, -1.6 \leqslant a_1 \leqslant 1.6\} \tag{17.23}$$

由方程(17.17)知 $V(\mathbf{a})$的导数仅当速度为 0 时才为 0，这相当于 $a_2$ 轴。因此 $Z$ 即由落在 $G$ 中的那段 $a_2$ 轴构成。集合 $Z$ 显示在图 17-11 中。

集合 $L$ 是 $Z$ 中的最大不变集。要找到 $L$ 就必须回答这样一个问题：如果在 $-1.6$ 与 1.6 弧度间的初始位置以零初速度释放单摆，那么单摆的速度会保持为零吗？很清楚这样的初始条件只能是在 0 弧度处(垂直悬挂)。如果在 $Z$ 中的其他任意位置释放单摆，单摆都会下落，因而速度不会保持为零并且轨迹将会移出 $Z$ 的范围。因此，集合 $L$ 只含有原点：

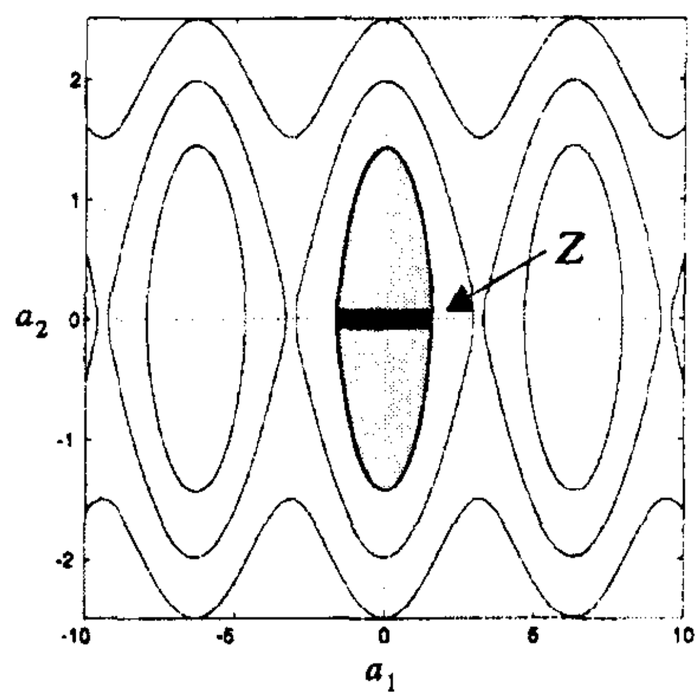



图 17-11 集合 Z 的图示

17-15

$$L = \{\mathbf{a} : \mathbf{a} = 0\} \tag{17.24}$$

集合 $L^\circ$是 $L$ 和 $G$ 的交集的闭包，在本例中，它就是 $L$：

$$L^\circ = closure(L \cap G) = L = \{\mathbf{a} : \mathbf{a} = 0\} \tag{17.25}$$

因此，根据 LaSalle 推论，$L^\circ$是一个吸引子(渐近稳定点)，而 $G$ 是在它的吸引区内。这意味着任何开始于 $G$ 的轨迹都将衰减至原点。

现在，假设有一个更大范围的 $\Omega_\eta$，例如

$$\Omega_{300} = \{\mathbf{a} : V(\mathbf{a}) < 300\} \tag{17.26}$$

该集合在图 17-12 中用灰色表示。

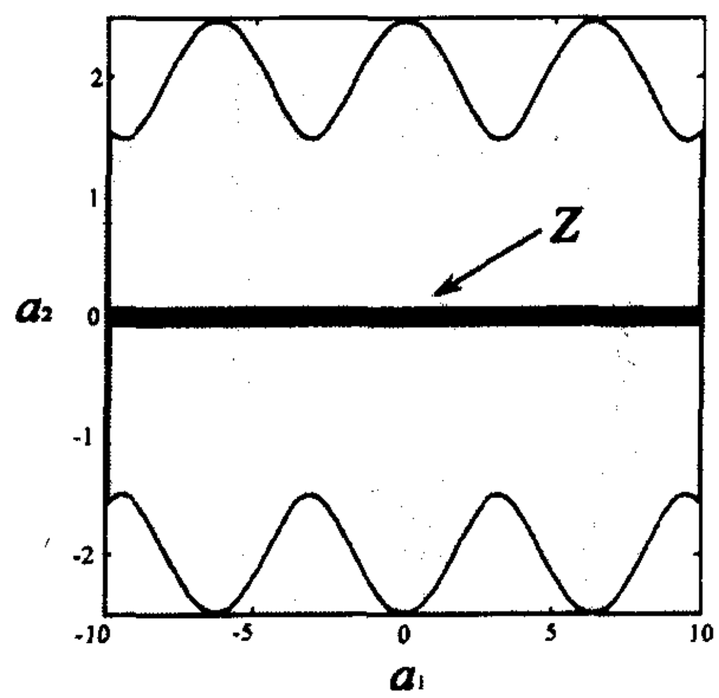



图 17-12 $G = \Omega_{300}$(灰色)和 $Z$ 的图解

17-16

令 $G = \Omega_{300}$，因为 $\Omega_{300}$只有一个部分。集合 $Z$ 由下式给出：

$$Z = \{\mathbf{a} : a_2 = 0\} \tag{17.27}$$

它在图 17-12 中用水平轴上的黑色条表示。这样可以推知

$$L^\circ = L = \{\mathbf{a} : a_1 = \pm n\pi, a_2 = 0\} \tag{17.28}$$

这是因为现在 $Z$ 中有几个不同的位置，在那些地方放置单摆不会导致速度变成非零。单摆可能直接朝上或朝下。这相当于 $\pm n\pi$ 的位置，$n$ 是任意整数。如果将单摆置于这些位置中任一处，初速为零，那么单摆将保持静止。可以令方程(17.14)和(17.15)中导数为零来表明

这点：

$$\frac{\mathrm{d}a_1}{\mathrm{d}t} = a_2 = 0 \tag{17.29}$$

$$\left(\frac{\mathrm{d}a_2}{\mathrm{d}t} = -\sin(a_1) - 0.2a_2 = -\sin(a_1) = 0\right) \Rightarrow (a_1 = \pm n\pi) \tag{17.30}$$

在 $G = \Omega_{300}$这个选择下，很难说轨迹会在哪一个点收敛。我们试图增加已知的原点吸引区的大小，但 $G$ 是一个对所有平衡点的吸引区。我们使 $G$ 过于大了。集合 $L^\circ$在图 17-13 中用黑色点表示。

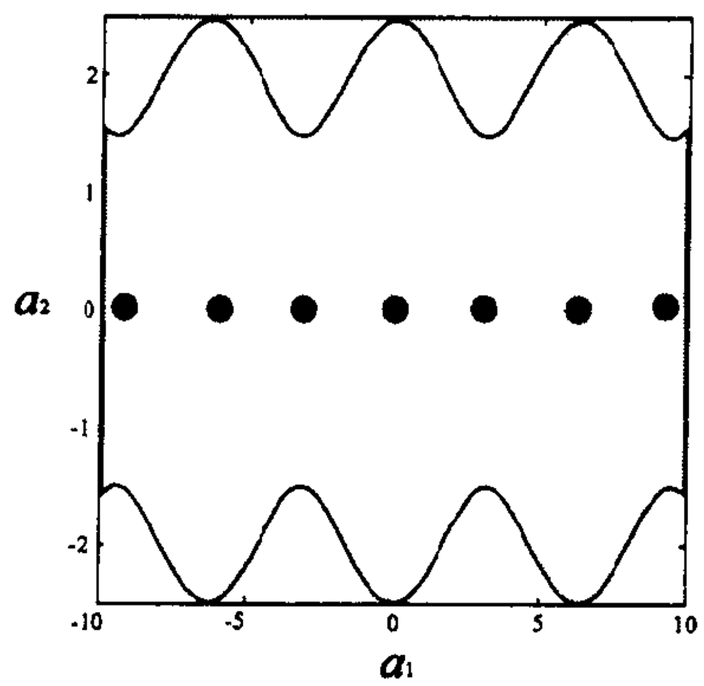



图 17-13 集合 $L^\circ$

我们不能分辨哪一个平衡点(黑色点)将吸引轨迹。所有我们能说的是如果从 $\Omega_{300}$中某个地方开始，有一个平衡点将吸引系统的解，但不能确切地说出是哪个点。例如，考虑如图 17-14 的轨迹。它表示一个初始位置为 2 弧度、初速度为 1.5 弧度每秒的单摆的响应轨迹。这时单摆具有足够大的速度跨过顶部，然后收敛于位于 $2\pi$ 弧度的平衡点。 17-17

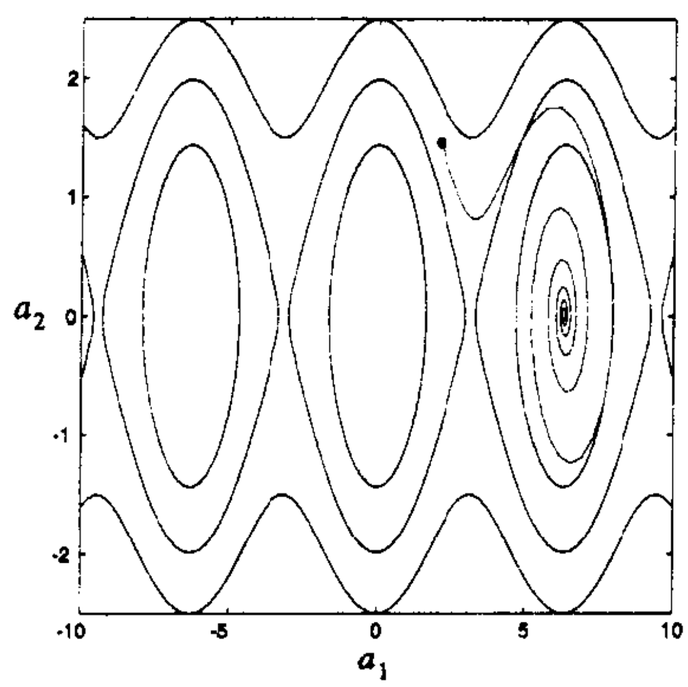



图 17-14 不同起始条件下的单摆轨迹

至此已讨论了 LaSalle 不变性定理，你可能想做些更多的单摆实验，去分析不同的稳定点的吸引区。做单摆实验请用 *Neural Network Design Demonstration Dynamic System* (**nnd17ds**)。

4. 评述

LaSalle 定理的关键是对 Lyapunov 函数 $V$ 和集合 $G$ 的选择。我们希望 $G$ 尽可能地大，因为它指示着吸引区的范围。但是，又希望选择 $V$ 进而使集合 $Z$ 尽可能地小，因其包含有吸引子集合。

举个例子，试令 $V=0$。这是$\Re^n$ 整个空间上的一个 Lyapunov 函数，它的导数无论在何处皆为 0(因此不会改变符号)。但是，由于 $Z=\Re^n$，并未得到什么信息。

注意，如果 $V_1$ 和 $V_2$ 都是 $G$ 上的 Lyapunov 函数，且 $\mathrm{d}V_1/\mathrm{d}t$ 与 $\mathrm{d}V_2/\mathrm{d}t$ 有相同的符号，那么 $V=V_1+V_2$ 也是一个 Lyapunov 函数，其中 $Z=Z_1\cap Z_2$。如果 $Z$ 比 $Z_1$ 和 $Z_2$ 都小，那么 $V$ 是一个比 $V_1$ 或 $V_2$ 都"更好"的 Lyapunov 函数。$V$ 总是至少与 $V_1$ 或 $V_2$ 一样"好"，因为 $Z$ 永远不会比 $Z_1$ 和 $Z_2$ 中的较小者大。因此，如果你发现了两个 Lyapunov 函数，并且它们的导数具有相同的符号，那么将它们加到一起，你将获得一个更好的函数。对于一个给定的系统。其最好的 Lyapunov 函数是那种具有最小的吸引子集合和最大的吸引区的函数。

17-18

## 17.3 小结

### 稳定性概念

**定义**

**定义 1** 稳定性(在 Lyapunov 的意义下)

原点是一个稳定平衡点，如果对于任意给定值 $\varepsilon>0$，总存在一个数 $\delta(\varepsilon)>0$，使得当 $\|\mathbf{a}(0)\|<\delta$时产生的运动 $\mathbf{a}(t)$满足$\|\mathbf{a}(t)\|<\varepsilon(\mathbf{t}>0)$。

**定义 2** 渐近稳定性

原点是一个渐近稳定平衡点，如果存在一个值 $\delta>0$，使得当$\|\mathbf{a}(0)\|<\delta$ 时产生的运动在 $t\to\infty$时满足$\|\mathbf{a}(t)\|\to 0$。

**定义 3** 正定

一个标量函数 $V(\mathbf{a})$，当 $V(\mathbf{0})=0$ 且 $V(\mathbf{a})>0(\mathbf{a}\neq 0)$时是正定的。

**定义 4** 半正定

一个标量函数 $V(\mathbf{a})$，当 $V(\mathbf{a})\geqslant 0$ (对于所有 $\mathbf{a}$)时是半正定的。

### Lyapunov 稳定性定理

考虑一个自主(无外力的不明显地依赖于时间)的系统

$$\frac{\mathrm{d}\mathbf{a}}{\mathrm{d}t}=\mathbf{g}(\mathbf{a})$$

Lyapunov 稳定性定理可表述如下。

**定理 1** Lyapunov 稳定性定理

如果能够找到一个正定函数 $V(\mathbf{a})$，使得 $\mathrm{d}V(\mathbf{a})/\mathrm{d}t$ 是半负定的，那么对于方程(17.2)所示系统，原点($\mathbf{a}=\mathbf{0}$)是稳定的。如果能够找到一个正定函数 $V(\mathbf{a})$，使得 $\mathrm{d}V(\mathbf{a})/\mathrm{d}t$ 是一个负定函数，那么其原点($\mathbf{a}=\mathbf{0}$)是渐近稳定的。在这种情况下，$V$ 被称为系统的 Lyapunov 函数。

17-19

### LaSalle 不变性定理

**定义**

**定义 5** Lyapunov 函数

令 $V$ 是一个从$\Re^n$ 到$\Re$的连续可微函数。若 $G$ 是$\Re^n$ 的任一子集，称 $V$ 是系统 $\mathrm{d}\mathbf{a}/\mathrm{d}t = \mathbf{g}(\mathbf{a})$在 $G$ 上的 Lyapunov 函数，如果

$$\frac{\mathrm{d}V(\mathbf{a})}{\mathrm{d}t} = (\nabla V(\mathbf{a}))^T\mathbf{g}(\mathbf{a})$$

在 $G$ 上不改变符号。

**定义 6** 集合 $Z$

$$Z = \{\mathbf{a} : \mathrm{d}V(\mathbf{a})/\mathrm{d}t = 0,\quad \mathbf{a}\text{在 } G \text{ 的闭包中}\} \tag{17.31}$$

**定义 7** 不变集

$\Re^n$ 中的一个点集合 $G$ 关于 $\mathrm{d}\mathbf{a}/\mathrm{d}t = \mathbf{g}(\mathbf{a})$是不变的，如果 $\mathrm{d}\mathbf{a}/\mathrm{d}t = \mathbf{g}(\mathbf{a})$的每一个开始于 $G$ 中的解始终保持在 $G$ 中。

**定义 8** 集合 $L$

$L$ 定义为 $Z$ 中最大的不变集。

**定理**

**定理 2** LaSalle 不变性定理

若 $V$ 是 $\mathrm{d}\mathbf{a}/\mathrm{d}t = \mathbf{g}(\mathbf{a})$在 $G$ 上的 Lyapunov 函数，那么对于所有 $t > 0$，$G$ 中的每一个解 $\mathbf{a}(t)$当 $t \to \infty$时趋于 $L^\circ = L \cup \{\infty\}$。($G$ 具有所有的稳定点，是对 $L$ 的吸引区。)若所有的轨迹都是有界的，则当时 $t \to \infty$时 $\mathbf{a}(t) \to L$。

**推论 1** LaSalle 定理的推论

令 $G$ 包含于集合

$$\Omega_\eta = \{\mathbf{a} : V(\mathbf{a}) < \eta\} \tag{17.32}$$

中(作为一个连通的子集)。假设 $G$ 是有界的，在 $G$ 上 $\mathrm{d}V(\mathbf{a})/\mathrm{d}t \leqslant 0$，且令集合 $L^\circ = closure(L \cap G)$是 $G$ 的一个子集。那么 $L^\circ$是一个吸引子，而 $G$ 是它的吸引区。 17-20

## 17.4 例题

**P17.1** 测试下面系统中原点的稳定性：

$$\mathrm{d}a_1/\mathrm{d}t = -a_1 + (a_2)^2$$
$$\mathrm{d}a_2/\mathrm{d}t = -a_2(a_1 + 1)$$

**解**

这里基本的工作是找到一个正定的 Lyapunov 函数 $V(\mathbf{a})$，其导数是半负定的，或者，更好是负定的。(后者是一个更强的条件。)

试用 $V(\mathbf{a}) = (a_1)^2 + (a_2)^2$。$V(\mathbf{a})$的导数为

$$\frac{\mathrm{d}V(\mathbf{a})}{\mathrm{d}t} = (\nabla V)^T\left(\frac{\mathrm{d}\mathbf{a}}{\mathrm{d}t}\right) = \frac{\partial V}{\partial a_1}\left(\frac{\mathrm{d}a_1}{\mathrm{d}t}\right) + \frac{\partial V}{\partial a_2}\left(\frac{\mathrm{d}a_2}{\mathrm{d}t}\right)$$

或者

$$\frac{\mathrm{d}V(\mathbf{a})}{\mathrm{d}t} = 2a_1(-a_1 + (a_2)^2) + 2a_2(-a_2(a_1+1)) = -2(a_1)^2 - 2(a_2)^2$$

导数 $\mathrm{d}V(\mathbf{a})/\mathrm{d}t$ 是负定的。因此，原点是渐近稳定的。

**P17.2** 测试下面系统中原点的稳定性：

$$\mathrm{d}a_1/\mathrm{d}t = -(a_1)^5$$
$$\mathrm{d}a_2/\mathrm{d}t = -5(a_2)^7$$

**解**

试用 $V(\mathbf{a}) = (a_1)^2 + (a_2)^2$。于是有

$$\frac{\mathrm{d}V(\mathbf{a})}{\mathrm{d}t} = 2a_1(-(a_1)^5) + 2a_2(-5(a_2)^7) = -2(a_1)^6 - 10(a_2)^8$$

这里 $\mathrm{d}V(\mathbf{a})/\mathrm{d}t$ 也是负定的，因此原点是渐近稳定的。

**P17.3** 考虑图 17-15 所示的机械系统。这是一个具有一个非线性弹簧的弹簧 - 物块 - 阻尼器系统。我们定义 $a_1 = x$ 和 $a_2 = \mathrm{d}x/\mathrm{d}t$，则运动方程为

17-21

$$\mathrm{d}a_1/\mathrm{d}t = a_2$$
$$\mathrm{d}a_2/\mathrm{d}t = -(a_1)^3 - a_2(\text{非线性弹簧})$$

考虑候选 Lyapunov 函数

$$V(\mathbf{a}) = \frac{1}{4}(a_1)^4 + \frac{1}{2}(a_2)^2$$

利用 LaSalle 不变性定理的推论提供尽可能多的关于平衡点和吸引区的信息。

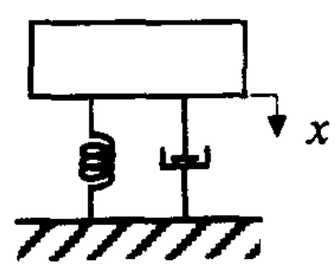



图 17-15 机械系统

**解**

首先计算 $V(\mathbf{a})$的导数，

$$\frac{\mathrm{d}V(\mathbf{a})}{\mathrm{d}t} = \frac{\partial V}{\partial a_1}\left(\frac{\mathrm{d}a_1}{\mathrm{d}t}\right) + \frac{\partial V}{\partial a_2}\left(\frac{\mathrm{d}a_2}{\mathrm{d}t}\right) = (a_1)^3 a_2 + a_2(-(a_1)^3 - a_2) = -(a_2)^2$$

这样，$\mathrm{d}V/\mathrm{d}t$ 在$\Re^2$ 上不改变符号。

现在定义

$$G = \Omega_\eta = \{\mathbf{a}:\ V(\mathbf{a}) \leqslant \eta\}$$

17-22 并且考虑 $\eta = 1$ 的情况。$V(\mathbf{a})$的等值图如图 17-16 所示。集合 $\Omega_1$ 在图中用黑色标志。

现在要判定集合 $Z$。

$$Z = \{\mathbf{a}: \mathrm{d}V/\mathrm{d}t = 0,\quad \mathbf{a}\text{在 } G \text{ 的闭包中}\} = \{\mathbf{a}: a_2 = 0,\quad \mathbf{a}\text{在 } G \text{ 的闭包中}\}$$

或者

$$Z = \{\mathbf{a}: a_2 = 0,\ -\sqrt{2} \leqslant a_1 \leqslant \sqrt{2}\}$$

下面找集合 $L$。由于 $\mathbf{a} = \mathbf{0}$是惟一的不变集，

$$L = \{\mathbf{a}: a_1 = 0,\quad a_2 = 0\}$$

因此，原点

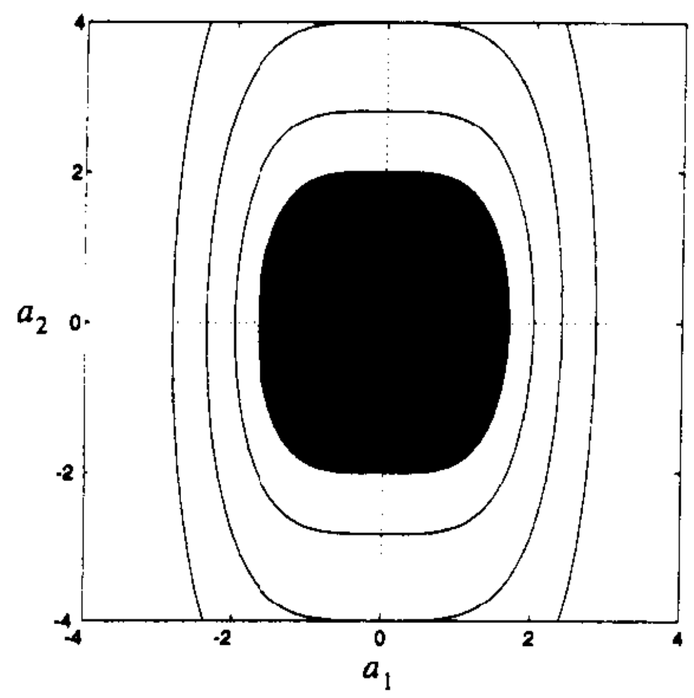



图 17-16 $V(\mathbf{a})$和 $\Omega_1$ 的等值图

$$\mathbf{a} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

是一个吸引子，且 $\Omega_1$ 即为它的吸引区。

进一步，可以增加 $\eta$ 值，使整个$\Re^2$ 都是原点的吸引区。

图 17-17 展示了弹簧 - 物块 - 阻尼器从初始位置 2、初始速度 2 开始的响应。注意，轨迹在穿过 $a_2$ 轴时与等值线平行。这与早先的结论相符，即只要 $a_2 = 0$ 时，Lyapunov 函数的导数为 0。幸运的是，$a_2$ 轴不是一个不变集(除了原点以外)；因而轨迹只被原点所吸引。

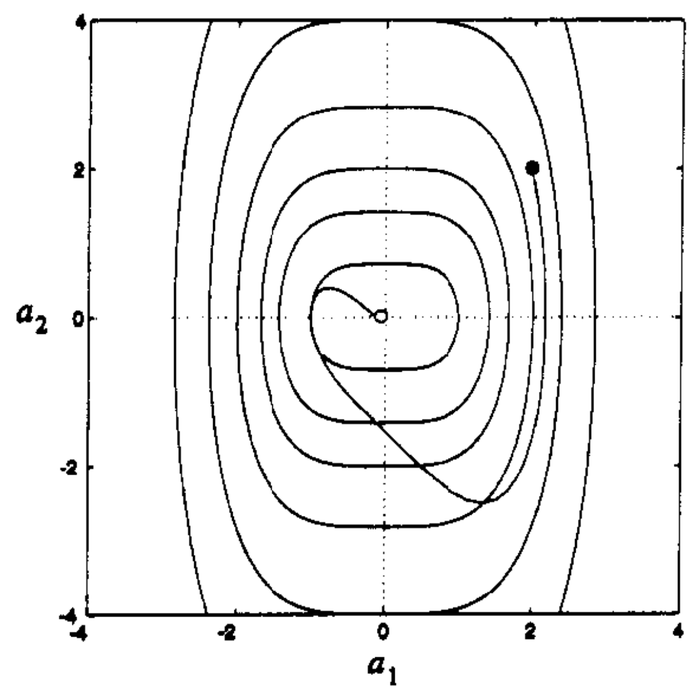



图 17-17 弹簧 - 物块 - 阻尼器响应

17-23

**P17.4** 考虑下面的非线性系统：

$$\mathrm{d}a_1/\mathrm{d}t = a_1((a_1)^2 + (a_2)^2 - 4) - a_2$$

$$\mathrm{d}a_2/\mathrm{d}t = a_1 + a_2((a_1)^2 + (a_2)^2 - 4)$$

该系统具有两个不变集，即原点

$$\{\mathbf{a} : \mathbf{a} = \mathbf{0}\}$$

和圆

$$\{\mathbf{a} : (a_1)^2 + (a_2)^2 = 4\}$$

假设候选 Lyapunov 函数

$$V(\mathbf{a}) = (a_1)^2 + (a_2)^2$$

利用 LaSalle 不变性定理，找出关于原点吸引区的尽可能多的信息。

**解**

于是，我们的工作是判断所给的不变集是否代表了一个稳定点或者一个稳定的轨迹。首先看一看 $\mathrm{d}V/\mathrm{d}t$。回想

$$\frac{\mathrm{d}V(\mathbf{a})}{\mathrm{d}t} = \frac{\partial V}{\partial a_1}\left(\frac{\mathrm{d}a_1}{\mathrm{d}t}\right) + \frac{\partial V}{\partial a_2}\left(\frac{\mathrm{d}a_2}{\mathrm{d}t}\right)$$

替换其可变项得到

$$\frac{\mathrm{d}V(\mathbf{a})}{\mathrm{d}t} = 2a_1[a_1((a_1)^2 + (a_2)^2 - 4) - a_2] + 2a_2[a_1 + a_2((a_1)^2 + (a_2)^2 - 4)]$$

化简为

$$\frac{\mathrm{d}V(\mathbf{a})}{\mathrm{d}t} = 2((a_1)^2 + (a_2)^2)((a_1)^2 + (a_2)^2 - 4)$$

这样，在 $\mathbf{a}=\mathbf{0}$处和在圆$(a_1)^2+(a_2)^2=4$ 上，$\mathrm{d}V/\mathrm{d}t$ 等于 0。

现在选取吸引区 $G$。在整个$\Re^2$ 上 $\mathrm{d}V/\mathrm{d}t$ 的符号有改变吗？有。当我们在 2 孤度处从圆的外部进入圆的内部时，$\mathrm{d}V/\mathrm{d}t$ 的符号由正变到负。因此 $\mathrm{d}V/\mathrm{d}t$ 在圆$(a_1)^2+(a_2)^2=4$ 内部是半负定的。在此圆内部选一个 $G$，使圆不被包括在其中。下面的集合满足要求：

17-24

$$G = \Omega_1 = \{\mathbf{a} : V(\mathbf{a}) \leq 1\}$$

现在考虑 $\Omega_1$。刚好有两个地方 $\mathrm{d}V/\mathrm{d}t=0$，并且 $\Omega_1$ 内仅有的一个点为 $\mathbf{a}=0$。因此

$$Z = \{\mathbf{a} : a_1 = 0, \quad a_2 = 0\}$$

且

$$L^\circ = L = Z$$

原点为吸引子，$\Omega_1$ 是它的吸引区。可以用同样的理由说明原点的吸引区包括圆$(a_1)^2+(a_2)^2=4$ 内部的所有点。

图 17-18 画出了该系统的两条轨迹，一条开始于圆$(a_1)^2+(a_2)^2=4$ 的内部，另一条开始于该圆的外部。虽然该圆是一个不变集，但它不是一个吸引子。这个系统惟一的吸引子是原点。

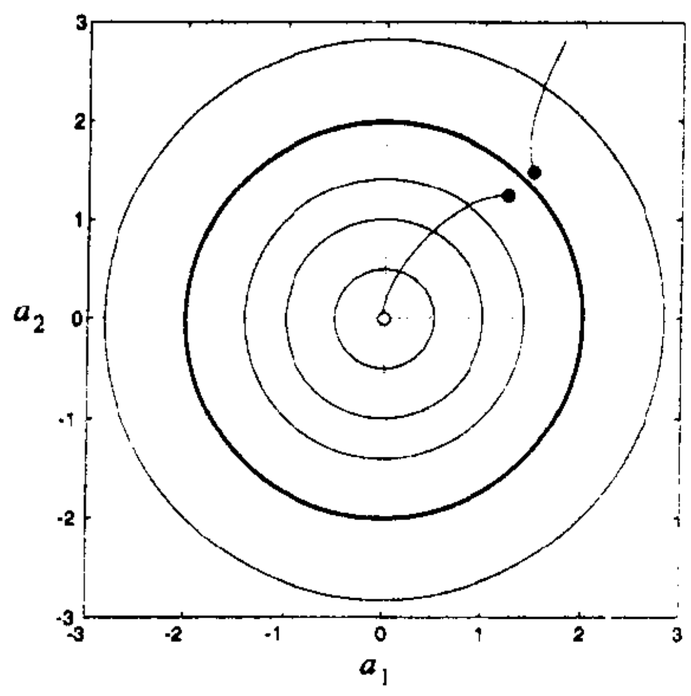



图 17-18 例题 P17.4 的样本轨迹

**P17.5** 考虑下面的非线性系统：

$$\mathrm{d}a(t)/(\mathrm{d}t) = -(a(t)-1)(a(t)-2)$$

（i）找出所有该系统的平衡点。

（ii）利用候选 Lyapunov 函数

$$V(\mathbf{a}) = (a-2)^2$$

获取你能从(i)小题中找到的有关平衡点吸引区的所有信息。（提示：利用 LaSalle 不变性定理的推论。）

17-25

**解**

（i）为找出平衡点，令 $\mathrm{d}a(t)/\mathrm{d}t=0$。

$$0 = -(a-1)(a-2) \quad \Rightarrow \quad a = 1, a = 2 \text{ 为平衡点}$$

（ii）为利用 LaSalle 定理的推论，需要求 $\mathrm{d}V/\mathrm{d}t$ 。

$$\frac{\mathrm{d}V}{\mathrm{d}t} = \frac{\partial V}{\partial a}\left(\frac{\mathrm{d}a}{\mathrm{d}t}\right) = 2(a-2)[-(a-1)(a-2)] = -2(a-1)(a-2)^2$$

现在令

$$G = \Omega_\eta = \{a : V(a) < \eta\}$$

例如，取 $\eta=0.5$，于是

$$G = \Omega_{0.5} = \{a : (a-2)^2 < 0.5\}$$

注意，求解 $(a-2)^2<0.5$ 得

$$\pm(a-2) < \sqrt{0.5} \quad \text{或者 } 1.3 < a < 2.7$$

这样，$\mathrm{d}V/\mathrm{d}t$ 在 $G$ 上是负定的。

接下来要找出集合 $Z$，它包含了 $G$ 中那些使 $\mathrm{d}V/\mathrm{d}t$ 等于零的点。有两个点使 $\mathrm{d}V/\mathrm{d}t$ 等于 0，即 $a=1$ 和 $a=2$。其中只有一个落在 $G$ 中，因此

$$Z = \{a : a = 2\}$$

现在需要找到 $L$，即 $Z$ 中的最大不变集。$Z$ 中只有一个点，而且是一个平衡点，于是

$$L^\circ = L = Z$$

这意味着 $G$ 处于 $Z$ 的吸引区中。

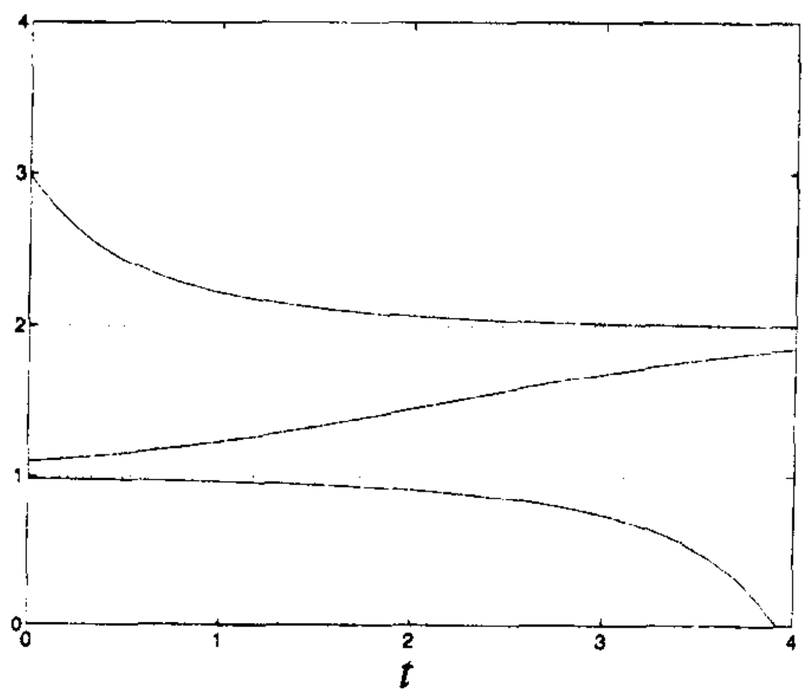
可以用同样的方法将 $\eta$ 增为 1.0 重来一次。于是可以说对于 $a=2$ 的吸引区至少应包括

$$\{a : 1 < a < 3\}$$

如果考虑那些 $\eta>1$ 的区域会怎样呢？这样 $Z$ 包括 1 和 2 在内，$\mathrm{d}V/\mathrm{d}t$ 在 $G$ 上将改变符号。因此使用这个 Lyapunov 函数和 LaSalle 不变性定理的推论，就不能说出任何关于 $a=1$ 的吸引区的内容。

17-26

图 17-19 展示了这个系统某些典型的响应。这里可以看出平衡点 $a=1$ 其实是不稳定的。任何超过 $a=1$ 的初始条件都会收敛于 $a=2$。任何小于 $a=1$ 的初始条件都会趋于负无穷大。




17-27 图 17-19 例题 P17.5 的稳定和不稳定响应

## 17.5 结束语

在这一章里，作为动力系统的应用，提出了稳定性的概念。对于非线性的动力系统，如像递归神经网络，我们不讨论有关系统稳定性的问题，而是讨论某些系统轨迹的稳定性，特别是在平衡点位置。

本章主要讨论了两个稳定性定理。第一个是 Lyapunov 稳定性定理，它介绍了广义的能量——Lyapunov 函数的概念。这个定理背后的思想是：如果一个系统的“能量”总在减小，那么它最终将稳定于最小“能量”点上。

提出的第二个定理是 LaSalle 不变性定理，它是 Lyapunov 稳定性定理的一种加强。LaSalle 作出了两个主要改进。第一是阐明这样一种现象，Lyapunov 函数在整个状态空间不减小，但是在某些区域保持常数值。LaSalle 定理引入不变集的概念来确定那些区域，它们可以捕捉系统轨迹。LaSalle 定理作出的第二个改进是，它不仅指明了平衡点的稳定性，而且也给出了关于每个稳定点的吸引区的信息。

这一章提出的一些思想是分析递归神经网络的重要工具，如对于第 15 章和第 16 章的 Grossberg 网络的分析。(LaSalle 不变性定理在递归神经网络中的应用参见[CoGr83]。)在第 18 章，将利用 LaSalle 定理解释 Hopfield 网络的操作。 17-28

## 参考文献

[Brog91] W. L. Brogan, *Modern Control Theory*, 3rd Ed., Englewood Cliffs, NJ: Prentice-Hall, 1991.

这是一本关于线性系统的很好的书。该书的前半部分讲线性代数。书中有几章讨论线性微分方程解法和线性及非线性系统的稳定性，写得很好。本书有很多例题。

[CoGR83] M. A. Cohen and S. Grossberg, “Absolute stability of global pattern formation and parallel memory storage by competitive neural networks,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 13, no. 5, pp. 815 - 826, 1983.

Cohen 和 Grossberg 用 LaSalle 的不变性定理来分析竞争型神经网络的稳定性。网络的描述是非常一般化的，作者们展示了如何将他们的分析运用到很多不同类型的递归

神经网络上。

[Lasa67] J. P. LaSalle, "An invariance principle in the theory of stability," in *Differential Equations and Dynamic Systems*, J. K. Hale and J. P. LaSalle, eds., New York: Academic Press, pp. 277 - 286, 1967.

这篇文章提供了对 Lyapuov 稳定性理论及其的几种扩展的统一表示。文中介绍了 LaSalle 不变性定理和若干推论。

[SlLi91] J. - J. E. Slotine and W. Li, *Applied Nonlinear Control*, Englewood Cliffs, NJ: Prentice-Hall, 1991.

主要介绍非线性控制系统，本书的很大一部分内容集中在动态非线性系统的分析上。书中还提出和展示了一些稳定性定理。

17-29

## 习题

**E17.1** 利用 Lyapunov 稳定性定理测试下面系统中原点的稳定性。

(i)
$$da_1/dt = -(a_1)^3 + a_2$$
$$da_2/dt = -a_1 - a_2$$

(ii)
$$da_1/dt = -a_1 + (a_2)^2$$
$$da_2/dt = -a_2(a_1 + 1)$$

**E17.2** 考虑下面的非线性系统：

$$da_1/dt = a_2 - 2a_1((a_1)^2 + (a_2)^2)$$
$$da_2/dt = -a_1 - 2a_2((a_1)^2 + (a_2)^2)$$

(i) 利用 Lyapunov 稳定性定理和下面所示的候选 Lyapunov 函数考察原点的稳定性：

$$V(\mathbf{a}) = \alpha(a_1)^2 + \beta(a_2)^2$$

(ii) 通过写出 MATLAB M-文件来模拟该系统对几个不同的初始条件的响应，检查你在(i)小题得到稳定性结果。利用 **ode45** 例行程序。画出响应图。

**E17.3** 对于非线性系统 $da/dt = \sin(a)$

(i) 找出所有不变集。

(ii) 找到一个 Lyapunov 函数，并指出吸引子及其吸引区。

17-30

**E17.4** 考虑下面的非线性系统：

$$da_1/dt = a_2$$
$$da_2/dt = -a_1 - (a_2)^3$$

(i) 找出所有平衡点。

(ii) 找出尽可能多的关于这些平衡点稳定性的信息，利用 LaSalle 定理的推论和候选 Lyapunov 函数

$$V(\mathbf{a}) = (a_1)^2 + (a_2)^2$$

(iii) 通过写出 MATLAB M-文件模拟该系统对几个不同的初始条件的响应，检查(i)、(ii)小题的结果。利用 **ode45** 例行程序。画出响应图。

**E17.5** 考虑下面的非线性系统：

$$da/dt = (1 - a)(1 + a) = 1 - a^2$$

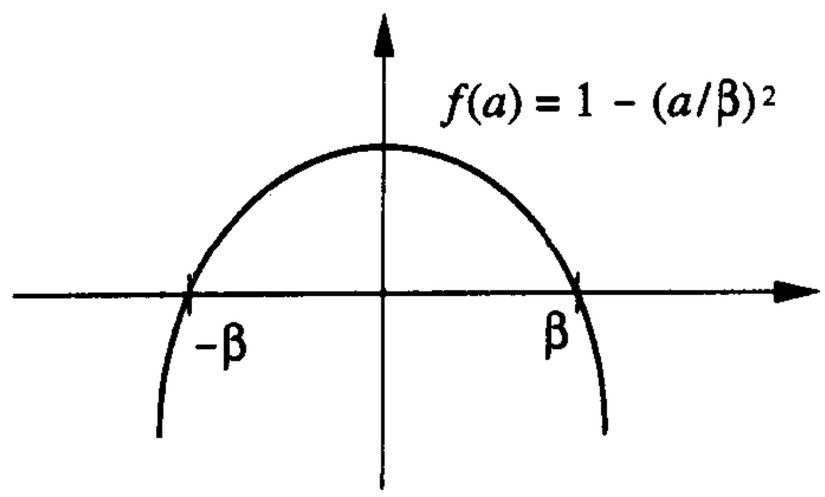
(i) 找出所有平衡点。

(ii) 找到一个合适的 Lyapunov 函数。(提示：从 $dV/dt$ 形式入手，反向推导找出 $V$。)

(iii) 画出 Lyapunov 函数简图。

(iv) 利用 LaSalle 定理的推论和(ii)小题的 Lyapunov 函数找出关于吸引区的尽可能多的信息。可能的话使用图形。

(提示：图 17-20 中所示图形可能会有帮助。)




17-31

图 17-20 对习题 E17.5 有帮助的函数

**E17.6 考虑系统**

$$da_1/dt = a_2 - a_1((a_1)^4 + 2(a_2)^2 - 10)$$
$$da_2/dt = -(a_1)^3 - 3(a_2)^5((a_1)^4 + 2(a_2)^2 - 10)$$

(i) 求所有不变集。(为有助于确定不变集，可以用 MATLAB M-文件模拟此系统。)

(ii) 利用候选 Lyapunov 函数

$$V(\mathbf{a}) = ((a_1)^4 + 2(a_2)^2 - 10)^2$$

17-32

和 LaSalle 定理的推论，考查你从(i)题求出的不变集的稳定性。

# 第 18 章　Hopfield 网络

## 18.1　目的

本章将要讨论 Hopfield 递归神经网络。这种网络对 20 世纪 80 年代初神经网络研究的重新兴起有重大影响。我们首先给出这个网络的描述，然后将展示如何用 Lyapunov 稳定性原理来分析网络的运算。最后，我们将说明如何设计网络使其具有联想存储器性能。

本章把前面各章讨论的主题结合起来：离散型 Hopfieid 网络(第 3 章)；特征值与特征向量(第 6 章)；联想存储器和 Hebb 规则(第 7 章)；赫森矩阵、最优条件、二次函数和曲面以及轮廓图(第 8 章)；最速下降法和状态平面轨迹(第 9 章)；连续的递归网络(第 15 章)；Lyapunov 稳定性定理和 LaSalle 不变性定理(第 17 章)。这一章在某些方面是前面各章工作的最终结果。



## 18.2　理论和实例

在 20 世纪 80 年代初期，神经网络研究的重新兴起可归功于 John Hopfield 的工作。作为一个著名的物理学家，Hopfield 的名声和科学资历使人们对神经网络的研究恢复了信心。在 60 年代中期，由于误解人们对神经网络的研究前景很不乐观。在 Hopfield 早期学术活动中，他曾研究光和固体间的相互作用。后来，他集中精力研究生物分子间的电子转移机制。可以想像，他在数学和物理学上的学术研究和他后来在生物学上的经验的结合，为他在神经网络提出的概念和所作的贡献奠定了基础。

Hopfield 分别在 1982 年和 1984 年写了两篇非常有影响的论文[Hopf82]、[Hopf84]。这两篇文章集中了前人的许多观点，如 McCulloch 和 Pitts 的神经模型[McPi43]，Grossberg 的改进模型[Gros67]，Anderson 和 Kohonen 的线性联想器模型[Ande72]、[Koho72]以及 Anderson、Silverstein、Ritz 和 Jones 的盒中脑状态模型[AnSi77]。Hopfield 的论文可读性好，他把一些重要思想结合起来并进行了简明的数学分析(包括 Lyapunov 稳定性定理的应用)。

还有一些原因使 Hopfield 的论文显得如此重要。首先，他指出了神经网络与统计物理学中磁性材料的 Ising 模型的相似之处。这就使许多已存在的理论可用来对神经网络进行分析，同时也鼓舞了很多物理学家以及其他科学家和工程师开始注意对神经网络的研究。

Hopfield 也与 VLSI 芯片的 设计者们有接触，因为他长期与 AT&T 贝尔实验室保持联系。早在 1987 年，贝尔实验室就已成功的在 Hopfield 网络基础上开发了神经网络芯片。神经网络的一个主要应用前景在于 VLSI 和光学设备的并行实现。Hopfield 发表了他的网络实现思想也就使他与先前的神经网络研究者区分开来。

Hopfield 强调实践，不仅体现在他的网络的实现上，同时也体现在这些网络所解决的问题上。他早期的论文描述的应用包括按内容寻址存储器(后文将要讨论)，模数转换[TaHo86]及优化问题[HoTa85](如货郎担问题)。

下一节将要提出 Hopfield 模型。我们使用 Hopfield 1984 年的论文[Hopf84]中的连续型

18-2 模型。然后，用 Lyapunov 的稳定性原理和 LaSalle 不变性定理来分析 Hopfield 模型。最后一节我们将展示如何使用 Hebb 规则把 Hopfield 网络设计成按内容寻址的存储器。

### 18.2.1 Hopfield 模型

**Hopfield 模型** 与他的实践观点一致，Hopfield 以电路的形式提出了他的模型。基本的模型(见[Hopf84])如图 18-1 所示。

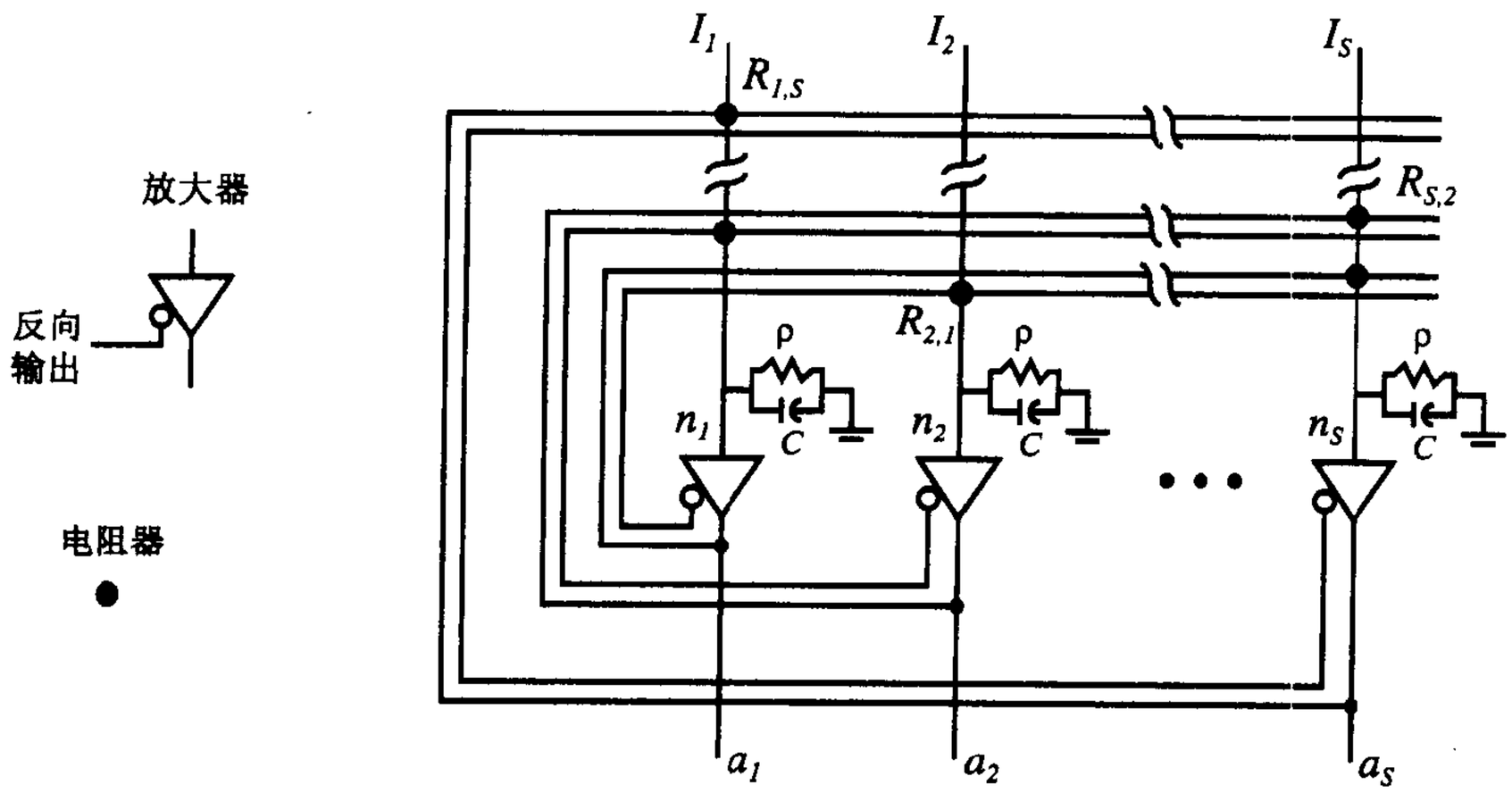



图 18-1 Hopfield 模型

每个运算放大器及其相关的电阻/电容网络代表一个神经元。神经元有两组输入。第一组是恒定的外部输入，用电流 $I_1$，$I_2$，……表示。第二组来自其他运算放大器的反馈连接。例如，第 2 个输出 $a_2$ 反馈到电阻 $R_{S,2}$上，而它又连到放大器 $S$ 的输入上。电阻只能是正的，但可通过使某个放大器的输出反相而使一个神经元获得负的输入。(在图 18-1 中，第一个放大器倒向的输出通过电阻 $R_{2,1}$连到第二个放大器的输入上。)

从 Kirchhoff 的电流定律可推导出 Hopfield 模型的运算方程

18-3

$$C\frac{\mathrm{d}n_i(t)}{\mathrm{d}t} = \sum_{j=1}^{S} T_{i,j} \quad a_j(t) - \frac{n_i(t)}{R_i} + I_i \tag{18.1}$$

式中 $n_i$ 表示第 $i$ 个放大器的输入电压，$a_i$ 是第 $i$ 个放大器的输出电压，C 表示放大器的输入电容，$I_i$ 是第 $i$ 个放大器的固定输入电流。同时有：

$$|T_{i,j}| = \frac{1}{R_{i,j}}, \quad \frac{1}{R_i} = \frac{1}{\rho} + \sum_{j=1}^{S}\frac{1}{R_{i,j}}, \quad n_i = f^{-1}(a_i)(\text{或 } a_i = f(n_i)) \tag{18.2}$$

$f(n)$是放大器的特性函数。在此处和下文中我们将假设电路是对称的，因此 $T_{i,j} = T_{j,i}$。

放大器的传输函数 $a_i = f(n_i)$通常是一个 S 形函数。S 形函数及其反函数我们都假设为增函数。在本章稍后，我们将给一个合适的传输函数。

在方程(18.1)两边乘以 $R_i$，可得到

$$R_iC\frac{\mathrm{d}n_i(t)}{\mathrm{d}t} = \sum_{j=1}^{S} R_iT_{i,j}\,a_i(t) - n_i(t) + R_iI_i \tag{18.3}$$

这可以转化为标准的神经网络表示法，如果定义

$$\varepsilon = R_i C, \quad w_{i,j} = R_i T_{i,j} \text{ 和 } b_i = R_i I_i \tag{18.4}$$

现在(18.3)式可改写为

$$\varepsilon \frac{\mathrm{d}n_i(t)}{\mathrm{d}t} = -n_i(t) + \sum_{j=1}^{S} w_{i,j} a_j(t) + b_i \tag{18.5}$$

写成向量形式就是

$$\varepsilon \frac{\mathrm{d}\mathbf{n}(t)}{\mathrm{d}t} = -\mathbf{n}(t) + \mathbf{W}\mathbf{a}(t) + \mathbf{b} \tag{18.6}$$

$$\mathbf{a}(t) = \mathbf{f}(\mathbf{n}(t)) \tag{18.7}$$

相应的 Hopfield 神经网络显示在图 18-2 中。

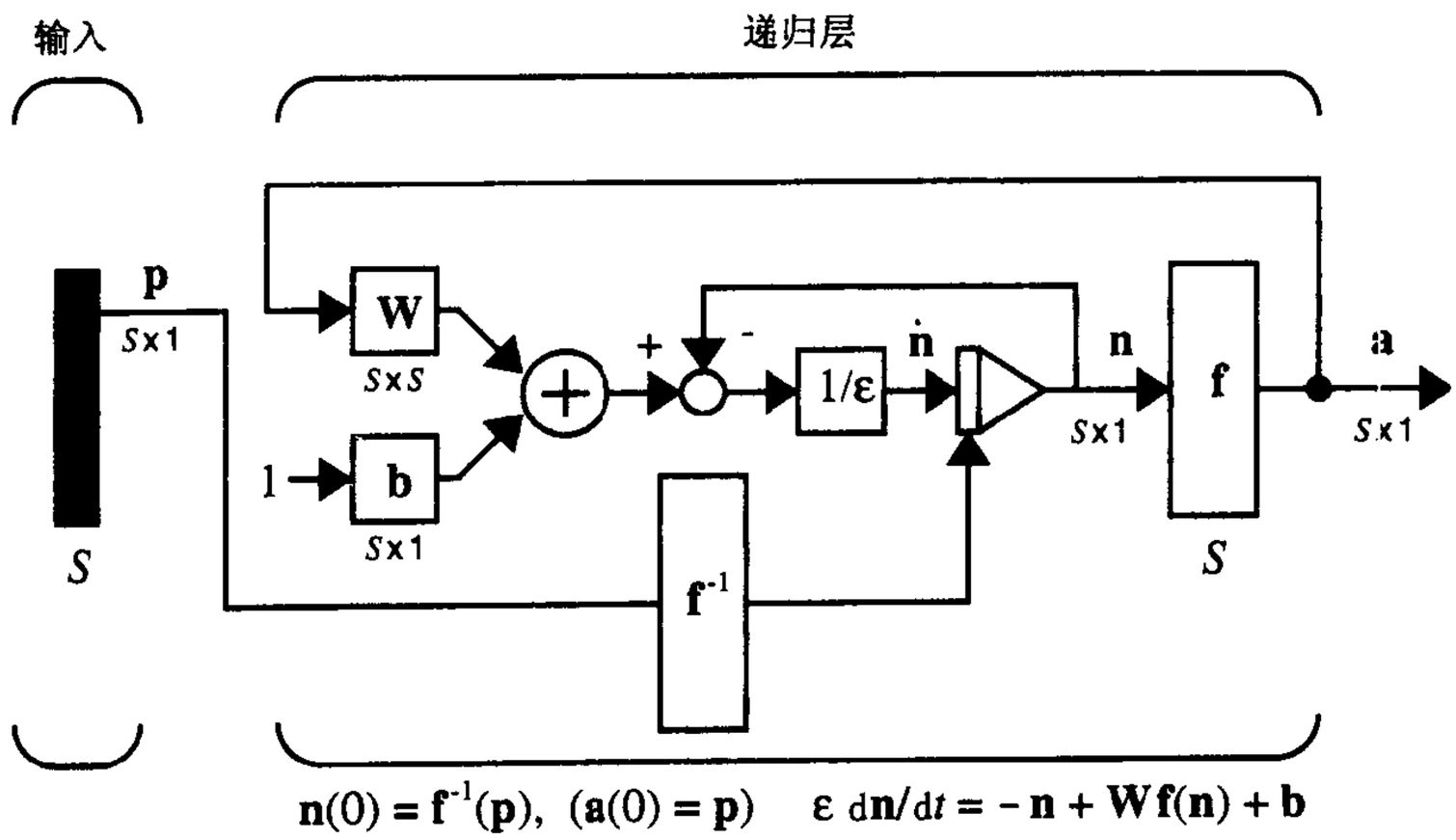



图 18-2 Hopfield 网络

因此，Hopfield 的起初的 S 型运算放大器电路可方便地用标准神经网络表示法表示。注意输入向量 **p** 决定着网络的初始输出。正如本章最后将要讨论的那样，这种形式的 Hopfield 网络可用来作联想存储器网络。 18-4

### 18.2.2 Lyapunov 函数

用 Lyapunov 稳定性定理分析递归网络是 Hopfield 的一个主要贡献。(Cohen 和 Grossberg 同期也用 Lyapunov 原理来分析竞争性的网络[CoGr83])。在这一节，我们将演示如何在 Hopfield 网络中使用第 17 章提出的 LaSalle 不变性定理。用 LaSalle 定理的第一步是选择一个 Lyapunov 函数。Hopfield 建议采用下面的函数：

$$V(\mathbf{a}) = -\frac{1}{2}\mathbf{a}^T\mathbf{W}\mathbf{a} + \sum_{i=1}^{S}\left\{\int_0^{a_i} f^{-1}(u)\mathrm{d}u\right\} - \mathbf{b}^T\mathbf{a} \tag{18.8}$$

Hopfield 选择这个特殊的 Lyapunov 的函数也是他的主要贡献之一。注意第一和第三项组成了一个二次函数。在本章后面有一节，将用先前关于二次函数的结果来分析这个 Lyapunov 函数。

为了使用 LaSalle 定理，我们将需要估计 $V(\mathbf{a})$的导数。为清晰起见，我们分别考虑 $V(\mathbf{a})$的三项。用等式(8.37)，第一项的导数为

$$\frac{\mathrm{d}}{\mathrm{d}t}\left\{-\frac{1}{2}\mathbf{a}^T\mathbf{W}\mathbf{a}\right\} = -\frac{1}{2}\nabla[\mathbf{a}^T\mathbf{W}\mathbf{a}]^T\frac{\mathrm{d}\mathbf{a}}{\mathrm{d}t} = -[\mathbf{W}\mathbf{a}]^T\frac{\mathrm{d}\mathbf{a}}{\mathrm{d}t} = -\mathbf{a}^T\mathbf{W}\frac{\mathrm{d}\mathbf{a}}{\mathrm{d}t} \tag{18.9}$$ 18-5

$V(\mathbf{a})$中的第二项由积分和组成。如果分别考虑每一项积分，我们可以得到

$$\frac{\mathrm{d}}{\mathrm{d}t}\left\{\int_0^{a_i} f^{-1}(u)\mathrm{d}u\right\} = \frac{\mathrm{d}}{\mathrm{d}a_i}\left\{\int_0^{a_i} f^{-1}(u)\mathrm{d}u\right\}\frac{\mathrm{d}a_i}{\mathrm{d}t} = f^{-1}(a_i)\frac{\mathrm{d}a_i}{\mathrm{d}t} = n_i\frac{\mathrm{d}a_i}{\mathrm{d}t} \tag{18.10}$$

这样 $V(\mathbf{a})$中第二项的总导数就可以表示为

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[\sum_{i=1}^{S}\left\{\int_0^{a_i} f^{-1}(u)\mathrm{d}u\right\}\right] = \mathbf{n}^T\frac{\mathrm{d}\mathbf{a}}{\mathrm{d}t} \tag{18.11}$$

由式(8.36)，我们可得到 $V(\mathbf{a})$第三项的导数

$$\frac{\mathrm{d}}{\mathrm{d}t}\{-\mathbf{b}^T\mathbf{a}\} = -\nabla[\mathbf{b}^T\mathbf{a}]^T\frac{\mathrm{d}\mathbf{a}}{\mathrm{d}t} = -\mathbf{b}^T\frac{\mathrm{d}\mathbf{a}}{\mathrm{d}t} \tag{18.12}$$

因此 $V(\mathbf{a})$的总导数可改写为

$$\frac{\mathrm{d}}{\mathrm{d}t}V(\mathbf{a}) = -\mathbf{a}^T\mathbf{W}\frac{\mathrm{d}\mathbf{a}}{\mathrm{d}t} + \mathbf{n}^T\frac{\mathrm{d}\mathbf{a}}{\mathrm{d}t} - \mathbf{b}^T\frac{\mathrm{d}\mathbf{a}}{\mathrm{d}t} = [-\mathbf{a}^T\mathbf{W} + \mathbf{n}^T - \mathbf{b}^T]\frac{\mathrm{d}\mathbf{a}}{\mathrm{d}t} \tag{18.13}$$

由式(18.6)，我们知道

$$[-\mathbf{a}^T\mathbf{W} + \mathbf{n}^T - \mathbf{b}^T] = -\varepsilon\left[\frac{\mathrm{d}\mathbf{n}(t)}{\mathrm{d}t}\right]^T \tag{18.14}$$

因此式(18.13)又可写作

$$\frac{\mathrm{d}}{\mathrm{d}t}V(\mathbf{a}) = -\varepsilon\left[\frac{\mathrm{d}\mathbf{n}(t)}{\mathrm{d}t}\right]^T\frac{\mathrm{d}\mathbf{a}}{\mathrm{d}t} = -\varepsilon\sum_{i=1}^{S}\left(\frac{\mathrm{d}n_i}{\mathrm{d}t}\right)\left(\frac{\mathrm{d}a_i}{\mathrm{d}t}\right) \tag{18.15}$$

因为 $n_i = f^{-1}(a_i)$，所以 $n_i$ 的导数可展开成：

$$\frac{\mathrm{d}n_i}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}[f^{-1}(a_i)] = \frac{\mathrm{d}}{\mathrm{d}a_i}[f^{-1}(a_i)]\frac{\mathrm{d}a_i}{\mathrm{d}t} \tag{18.16}$$

现在式(18.15)可重写为

$$\frac{\mathrm{d}}{\mathrm{d}t}V(\mathbf{a}) = -\varepsilon\sum_{i=1}^{S}\left(\frac{\mathrm{d}n_i}{\mathrm{d}t}\right)\left(\frac{\mathrm{d}a_i}{\mathrm{d}t}\right) = -\varepsilon\sum_{i=1}^{S}\left(\frac{\mathrm{d}}{\mathrm{d}a_i}[f^{-1}(a_i)]\right)\left(\frac{\mathrm{d}a_i}{\mathrm{d}t}\right)^2 \tag{18.17}$$

18-6 如果 $f^{-1}(a_i)$是一个增函数，对每个运算放大器来说有

$$\frac{\mathrm{d}}{\mathrm{d}a_i}[f^{-1}(a_i)] > 0 \tag{18.18}$$

由式(18.17)，可得

$$\frac{\mathrm{d}}{\mathrm{d}t}V(\mathbf{a}) \leqslant 0 \tag{18.19}$$

如果 $f^{-1}(a_i)$是增函数的话，那么 $\mathrm{d}V(\mathbf{a})/\mathrm{d}t$ 是一个半负定函数。所以 $V(\mathbf{a})$是一个有效的 Lyapunov 函数。

1. 不变集

现在我们用 LaSalle 的不变性定理来求 Hopfield 网络的平衡点。第一步先求集合 $Z$(式(17.19))：

$$Z = \{\mathbf{a} : \mathrm{d}V(\mathbf{a})/\mathrm{d}t = 0,\quad \mathbf{a}\text{属于 } G \text{ 的闭包}\} \tag{18.20}$$

这个集合包含了 Lyapunov 函数所有导数为 0 的点。现在假设 $G$ 是$\mathfrak{R}^S$ 的全体。

从式(18.17)可知，如果每个神经元输出的导数值为零，则这样的导数为零。

$$\frac{\mathrm{d}\mathbf{a}}{\mathrm{d}t} = \mathbf{0} \tag{18.21}$$

当输出的导数值为 0 时，电路处于平衡状态。因此，这些系统“能量”不再变化的点也就是电路的平衡点。

这就意味着 $Z$ 中最大的不变集合 $L$，恰恰就是集合 $Z$：

$$L = Z \tag{18.22}$$

因此，$Z$ 中所有点都是潜在的吸引子。

其他的一些特征我们在下面例子中解释。

2. 实例

下面这个例子选自 Hopfield 的论文[Hopf84]。我们考虑有这样一个放大器特性的系统：

$$a = f(n) = \frac{2}{\pi}\tan^{-1}\left(\frac{\gamma\pi n}{2}\right) \tag{18.23}$$

18-7

这个式子又可写作

$$n = \frac{2}{\gamma\pi}\tan\left(\frac{\pi}{2}a\right) \tag{18.24}$$

假设有两个放大器，其中一个的输出通过电阻单元连入另一个放大器的输入，因此

$$R_{1,2} = R_{2,1} = 1, \quad T_{1,2} = T_{2,1} = 1 \tag{18.25}$$

所以我们有加权矩阵

$$\mathbf{W} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{18.26}$$

如果放大器的输入电容也设置为 1，我们有

$$\epsilon = R_i C = 1 \tag{18.27}$$

另假设 $\gamma = 1.4$ 且 $I_1 = I_2 = 0$。因此

$$\mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{18.28}$$

回忆等式(18.8)，Lyapunov 函数为

$$V(\mathbf{a}) = -\frac{1}{2}\mathbf{a}^T\mathbf{W}\mathbf{a} + \sum_{i=1}^{S}\left\{\int_0^{a_i} f^{-1}(u)\mathrm{d}u\right\} - \mathbf{b}^T\mathbf{a} \tag{18.29}$$

对于本例，Lyapunov 函数的第一项为

$$-\frac{1}{2}\mathbf{a}^T\mathbf{W}\mathbf{a} = -\frac{1}{2}[a_1 \quad a_2]\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = -a_1 a_2 \tag{18.30}$$

第三项是 0，因为 $\mathbf{b}$ 为 $\mathbf{0}$。第二项的第 $i$ 部分为

$$\int_0^{a_i} f^{-1}(u)\mathrm{d}u = \frac{2}{\gamma\pi}\int_0^{a_i}\tan\left(\frac{\pi}{2}u\right)\mathrm{d}u = \frac{2}{\gamma\pi}\left[-\log\left[\cos\left(\frac{\pi}{2}u\right)\right]\frac{2}{\pi}\right]_0^{a_i} \tag{18.31}$$

此式可简化为

$$\int_0^{a_i} f^{-1}(u)\mathrm{d}u = -\frac{4}{\gamma\pi^2}\log\left[\cos\left(\frac{\pi}{2}a_i\right)\right] \tag{18.32}$$

18-8

最后，把所有三项都代入式(18.29)，我们有 Lypunov 函数：

$$V(\mathbf{a}) = -a_1 a_2 - \frac{4}{1.4\pi^2}\left[\log\left\{\cos\left(\frac{\pi}{2}a_1\right)\right\} + \log\left\{\cos\left(\frac{\pi}{2}a_2\right)\right\}\right] \tag{18.33}$$

现在可以写出网络的方程(式(18.6))。设 $\varepsilon=1$ 且 $b=0$，即为

$$\frac{\mathrm{d}\mathbf{n}}{\mathrm{d}t}=-\mathbf{n}+\mathbf{W}\mathbf{f}(n)=-\mathbf{n}+\mathbf{W}\mathbf{a} \tag{18.34}$$

如果代入式(18.26)的加权矩阵，这个表达式又可写为下面两个方程：

$$\mathrm{d}n_1/\mathrm{d}t=a_2-n_1 \tag{18.35}$$

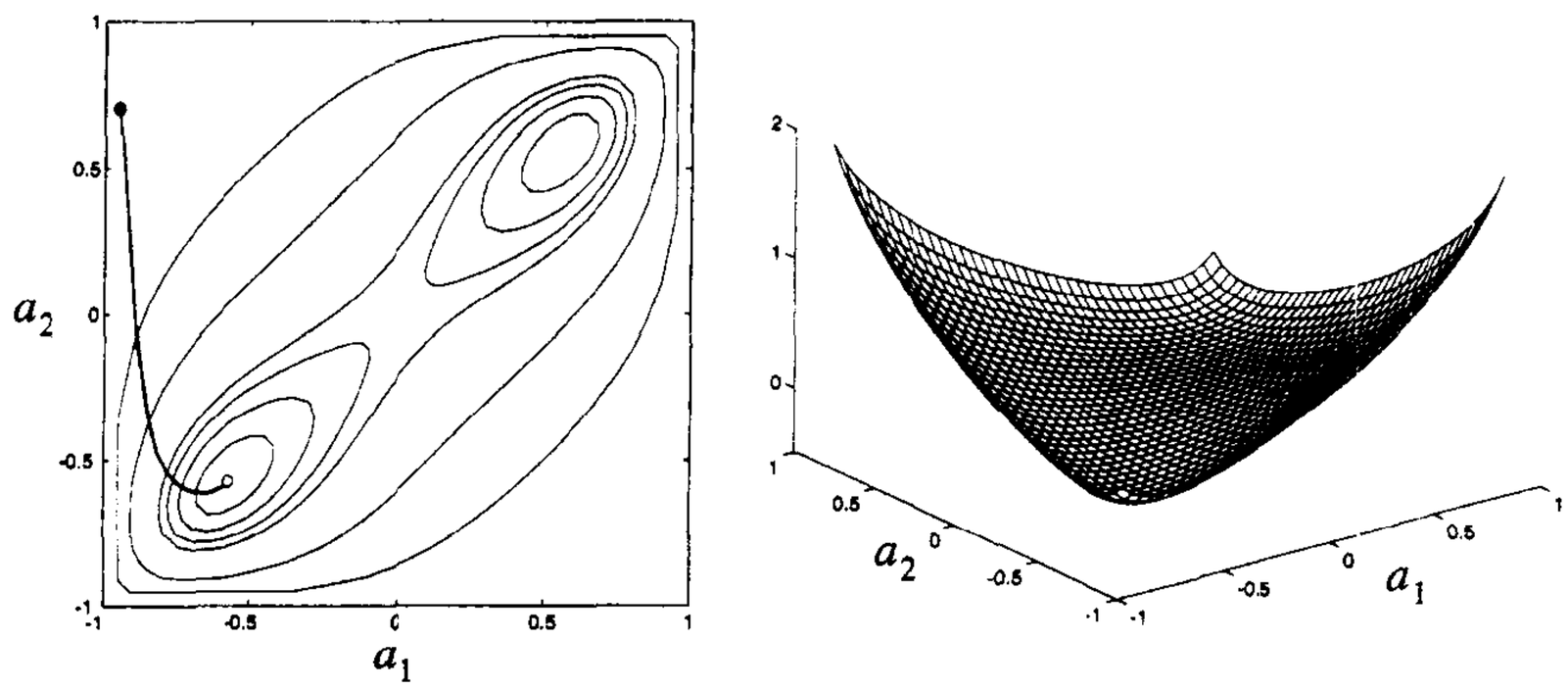
$$\mathrm{d}n_2/\mathrm{d}t=a_1-n_2 \tag{18.36}$$

神经元的输出为

$$a_1=\frac{2}{\pi}\tan^{-1}\left(\frac{1.4\pi}{2}n_1\right) \tag{18.37}$$

$$a_2=\frac{2}{\pi}\tan^{-1}\left(\frac{1.4\pi}{2}n_2\right) \tag{18.38}$$

至此我们已找到 Lyapunov 函数的表达式和网络的运算方程。让我们看看网络的特性。Lyapunov 函数图和样本轨迹如图 18-3 所示。




18-9

图 18-3 Hopfield 实例的 Lyapunov 函数和轨迹

图中的轮廓线表示 Lyapunov 函数的常数值。系统有两个吸引子，一个在图中的左下方，另一个在右上方。系统从左上方开始收敛于左下方的稳定点，如粗线所示。

图 18-4 显示了两个神经元输出的时间响应曲线。

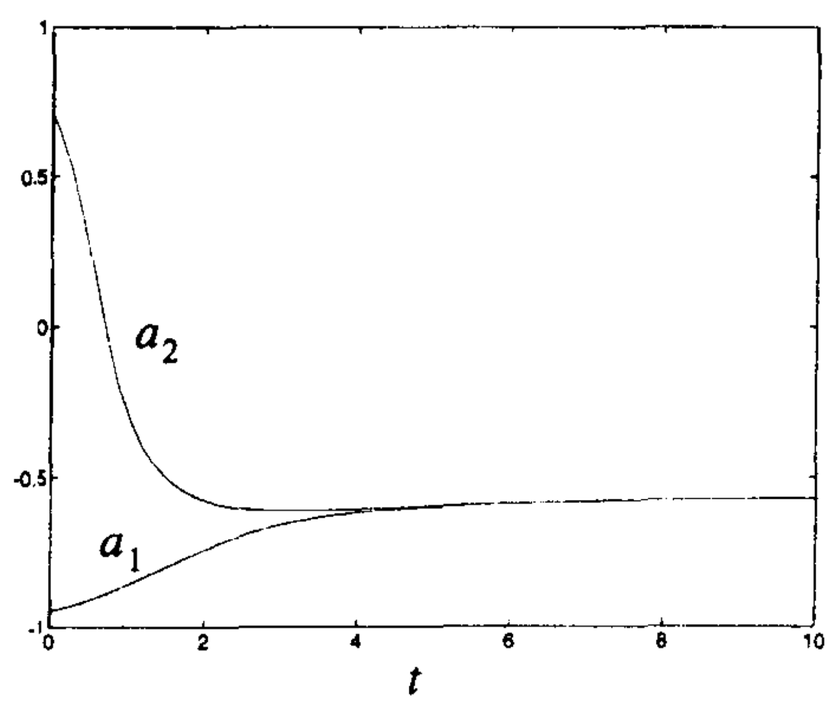



图 18-4 Hopfield 实例的时间响应图

图 18-5 显示了 Lyapunov 函数的时间响应。像预料那样，它逐渐下降趋于平衡点。

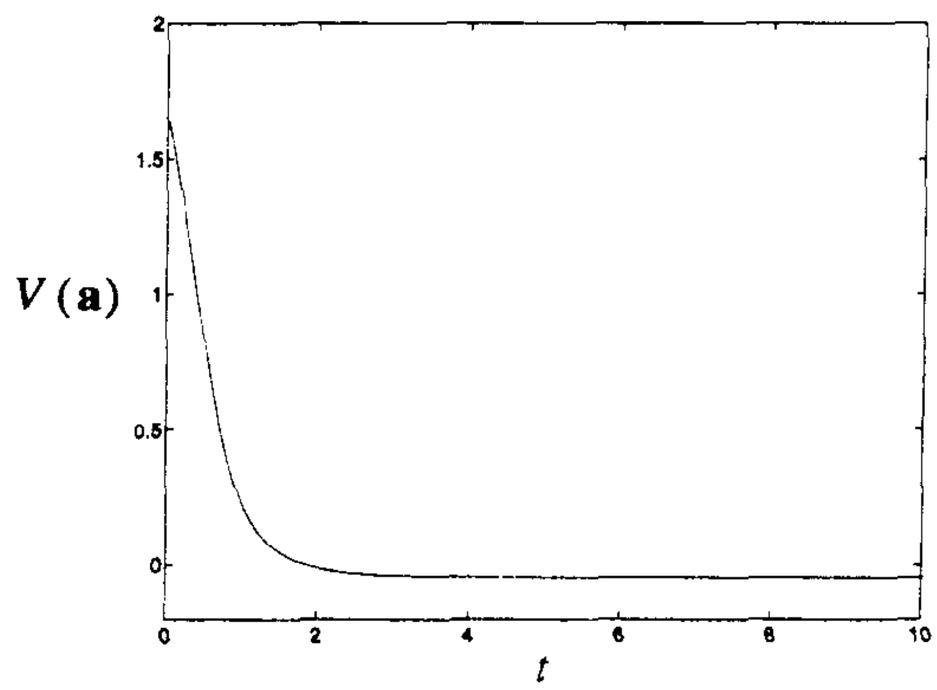



图 18-5 Lyapunov 函数响应图

系统在原点还有一个平衡点。如果把网络初始化在任何从左上角到右下角的对角线上，则解收敛于原点。但是，如果任何初始条件不落在这条对角线上，最终将收敛到左下角或右上角的解。在原点的解是 Lyapunov 函数的一个鞍点，不是局部极小值。这个问题在下一小节讨论。图 18-6 显示了收敛于鞍点的轨迹。 18-10

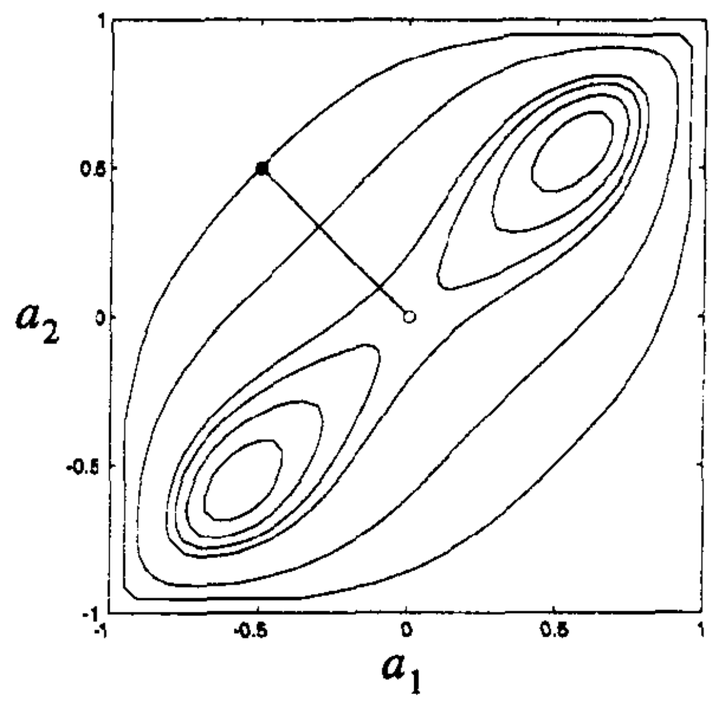



图 18-6 Hopfield 网络收敛于鞍点的情形

试验 Hopfield 网络请用 *Neural Network Design Demonstration* **Hopfield Network** (**nnd18hn**)。

这个例子给我们提供了一些 Hopfield 网络吸引子的情况。在下一小节，我们将做进一步分析。

3. Hopfield 网络吸引子

在前一小节的例子中，我们发现 Hopfield 网络的吸引子是 Lyapunov 函数的稳定点。现在我们要证明在一般情形下也是如此。回忆式(18.21)，Hopfield 网络的潜在吸引子应满足

$$\frac{\mathrm{d}\mathbf{a}}{\mathrm{d}t} = \mathbf{0} \tag{18.39}$$

这些点与 Lyapunov 函数的极小值有什么关系？在第 8 章(式(8.27))中，我们知道一个函数

的极小值一定是稳定点(梯度为 0)。$V(\mathbf{a})$的稳定点应满足

$$\nabla V = \left[\frac{\partial V}{\partial a_1} \quad \frac{\partial V}{\partial a_2} \quad \cdots \quad \frac{\partial V}{\partial a_S}\right]^T = \mathbf{0} \tag{18.40}$$

18-11 其中

$$V(\mathbf{a}) = -\frac{1}{2}\mathbf{a}^T\mathbf{W}\mathbf{a} + \sum_{i=1}^{S}\left\{\int_0^{a_i} f^{-1}(u)\mathrm{d}u\right\} - \mathbf{b}^T\mathbf{a} \tag{18.41}$$

仿照推导式(18.13)的过程，可得 $V(\mathbf{a})$梯度的表达式

$$\nabla V(\mathbf{a}) = [-\mathbf{W}\mathbf{a} + \mathbf{n} - \mathbf{b}] = -\varepsilon\left[\frac{\mathrm{d}\mathbf{n}(t)}{\mathrm{d}t}\right] \tag{18.42}$$

因此梯度的第 $i$ 个元素为

$$\frac{\partial}{\partial a_i}V(\mathbf{a}) = -\varepsilon\frac{\mathrm{d}n_i}{\mathrm{d}t} = -\varepsilon\frac{\mathrm{d}}{\mathrm{d}t}([f^{-1}(a_i)]) = -\varepsilon\frac{\mathrm{d}}{\mathrm{d}a_i}[f^{-1}(a_i)]\frac{\mathrm{d}a_i}{\mathrm{d}t} \tag{18.43}$$

附带说一下，如果 $f^{-1}(\mathbf{a})$是线性的，式(18.43)意味着

$$\frac{\mathrm{d}\mathbf{a}}{\mathrm{d}t} = -\alpha\nabla V(\mathbf{a}) \tag{18.44}$$

因此，Hopfield 网络的响应是沿着最陡的方向下降的。这样，如果在 $f^{-1}(\mathbf{a})$一个近似线性的区域内，网络的解也就近似地沿最陡方向下降。

我们已经假设传输函数和它的反函数是单调增函数。因此，

$$\frac{\mathrm{d}}{\mathrm{d}a_i}[f^{-1}(a_i)] > 0 \tag{18.45}$$

由式(18.43)，满足

$$\frac{\mathrm{d}\mathbf{a}(t)}{\mathrm{d}t} = 0 \tag{18.46}$$

的点也是满足

$$\nabla V(\mathbf{a}) = 0 \tag{18.47}$$

的点。

因此，作为集合 $L$ 中的元素且满足式(18.39)的吸引子也是 Lyapunov 函数 $V(\mathbf{a})$的稳定点。

### 18.2.3 增益效应

如果我们考虑放大器的增益系数 $\gamma$ 非常大的情况，那么 Hopfield Lyapunov 函数就可被
18-12 简化。回忆前面例子中的非线性放大器的特性

$$a = f(n) = \frac{2}{\pi}\tan^{-1}\left(\frac{\gamma\pi n}{2}\right) \tag{18.48}$$

图 18-7 显示了这个函数对于四种不同增益系数的曲线。

增益系数 $\gamma$ 决定了曲线在 $n=0$ 处的陡度。随着 $\gamma$ 的增大，曲线在原点的斜率增大。当 $\gamma$ 无限增大时，$f(n)$接近于正负号函数。

由式(18.8)，一般的 Lyapunov 函数是

$$V(\mathbf{a}) = -\frac{1}{2}\mathbf{a}^T\mathbf{W}\mathbf{a} + \sum_{i=1}^{S}\left\{\int_0^{a_i} f^{-1}(u)\mathrm{d}u\right\} - \mathbf{b}^T\mathbf{a} \tag{18.49}$$

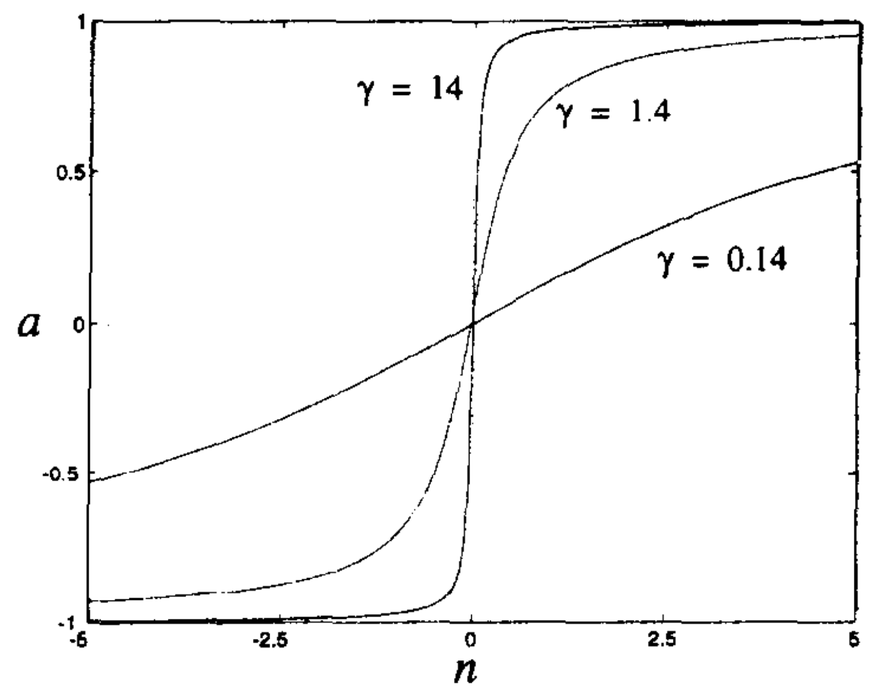



图 18-7 反正切放大器特性

对前面那个例子，

$$f^{-1}(u) = \frac{2}{\gamma\pi}\tan\left(\frac{\pi u}{2}\right) \tag{18.50}$$

因此，Lyapunov 函数的第二项便有如下形式：

$$\int_0^{a_i} f^{-1}(u)\mathrm{d}u = \frac{2}{\gamma\pi}\left[\frac{2}{\pi}\log\left(\cos\left(\frac{\pi a_i}{2}\right)\right)\right] = -\frac{4}{\gamma\pi^2}\log\left[\cos\left(\frac{\pi a_i}{2}\right)\right] \tag{18.51}$$

**高增益 Lyapunov 函数** 图 18-8 显示了三个不同增益系数的函数图形。注意，当 $\gamma$ 增大时，函数变得平坦并且在大部分地方都趋于 0。因此，当增益系数 $\gamma$ 无限增大时，在 $-1 < a_i < 1$ 范围内，Lyapunov 函数的第二项趋于 0。所以我们可以消去这一项，高增益 Lyapunov 函数便退化为 18-13

$$V(\mathbf{a}) = -\frac{1}{2}\mathbf{a}^T\mathbf{W}\mathbf{a} - \mathbf{b}^T\mathbf{a} \tag{18.52}$$

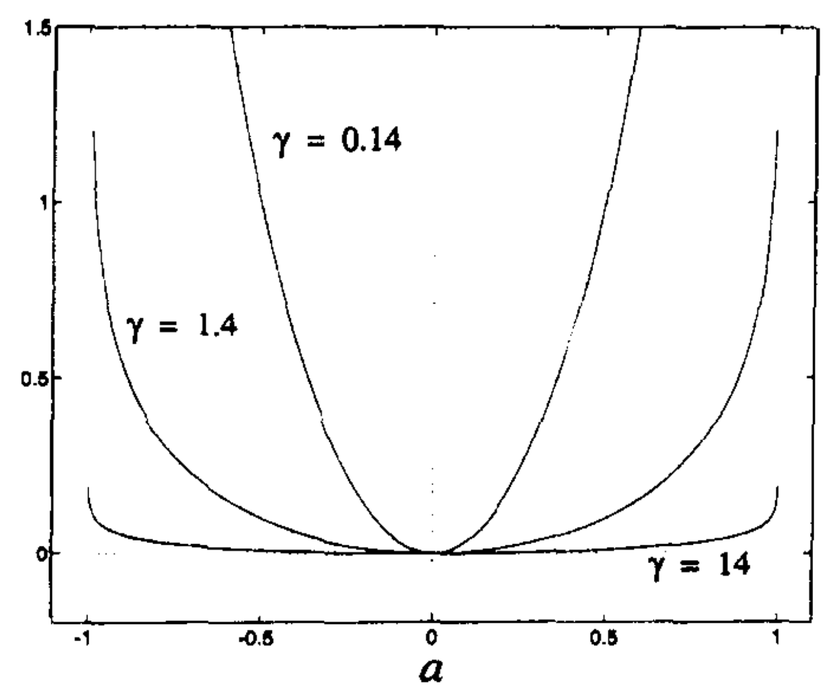



图 18-8 Lyapunov 函数的第二项

比较式(18.52)和(8.35)，我们不难发现高增益的 Lyapunov 函数实际上就是一个二次函数：

$$V(\mathbf{a}) = -\frac{1}{2}\mathbf{a}^T\mathbf{W}\mathbf{a} - \mathbf{b}^T\mathbf{a} = \frac{1}{2}\mathbf{a}^T\mathbf{A}\mathbf{a} + \mathbf{d}^T\mathbf{a} + c \tag{18.53}$$

其中

$$\nabla^2 V(\mathbf{a}) = \mathbf{A} = -\mathbf{W}, \mathbf{d} = -\mathbf{b}, c = 0 \tag{18.54}$$

这是一个重要的发展，因为现在我可以用第 8 章关于二次函数的结论来理解 Hopfield 网络的运算。

回忆二次函数的曲面形状由它的赫森矩阵的特征值和特征向量决定。对我们这个例子来说，Lyapunov 函数的赫森矩阵是

$$\nabla^2 V(\mathbf{a}) = -\mathbf{W} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \tag{18.55}$$

赫森矩阵的特征值计算如下：

$$|\nabla^2 V(\mathbf{a}) - \lambda \mathbf{I}| = \begin{vmatrix} -\lambda & -1 \\ -1 & -\lambda \end{vmatrix} = \lambda^2 - 1 = (\lambda + 1)(\lambda - 1) \tag{18.56}$$

18-14 因而，特征值是 $\lambda_1 = -1$ 和 $\lambda_2 = 1$。与之对应的特征向量是

$$\mathbf{z}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \text{ 和 } \mathbf{z}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \tag{18.57}$$

高增益 Lyapunov 函数的曲面形状是什么样子呢？我们知道这个赫森矩阵有一个正的和一个负的特征值，那么它满足鞍点条件。表面将会有一个沿着第一个特征向量的负曲率和沿着第二个特征向量的正曲率。曲面如图 18-9 所示。

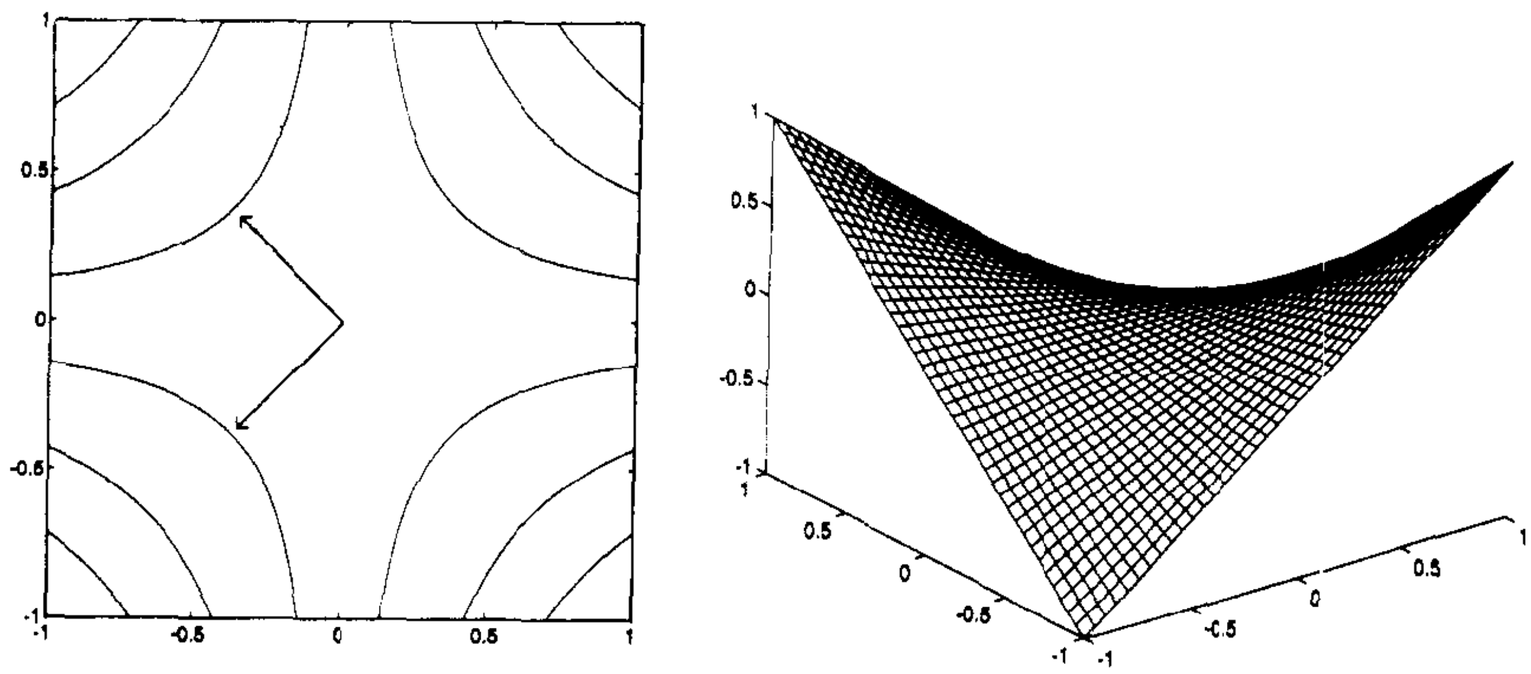



图 18-9 高增益 Lyapunov 函数的例子

这个函数没有极小值。但由于网络被放大器的传输函数限制在超立方体 $\{a: -1 < a_i < 1\}$ 内，因此，在超立方体

$$\mathbf{a} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \text{和} \quad \mathbf{a} = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \tag{18.58}$$

的两个角上被限制为极小值。当增益很小时，在原点有一个惟一的极小值(见习题 18.1)。随着增益的增加，两个极小值从原点移向由式(18.58)给出的两角。图 18-3 显示了一种中间情况，它的增益系数 $\gamma$ 为 1.4，图中极小值出现在

$$\mathbf{a} = \begin{bmatrix} 0.57 \\ 0.57 \end{bmatrix} \quad \text{和} \quad \mathbf{a} = \begin{bmatrix} -0.57 \\ -0.57 \end{bmatrix} \tag{18.59}$$

通常情况下，网络中不止有两个神经元，高增益的极小值将落入超立方体 $\{\mathbf{a}: -1 < a_i <$
18-15 $1\}$ 的某个角。在描述完 Hopfield 网络设计过程后，我们将后面几节更具体地讨论一般情形。

### 18.2.4 Hopfield 网络设计

Hopfield 网络没有与之相关的学习规则。它不被训练，也不会自己学习。它是用基于 Lyapunov 函数的设计过程来确定权值矩阵。

再次考虑高增益的 Lyapunov 函数

$$V(\mathbf{a}) = -\frac{1}{2}\mathbf{a}^T\mathbf{W}\mathbf{a} - \mathbf{b}^T\mathbf{a} \tag{18.60}$$

Hopfield 网络设计技术的关键在于选择权值矩阵 $\mathbf{W}$ 和偏置向量 $\mathbf{b}$ 以便使 $V$ 能得到想要的最小化函数形式。把需要求解的任何问题转化为二次函数的极小化问题。既然 Hopfield 网络可以用来最小化 $V$，那么也就可以解决原来的问题。自然，技巧在于转换，一般来说，这种转换并不是直接的。

1. 按内容寻址存储器

**按内容寻址存储器** 在这一小节，我们将描述如何用 Hopfield 网络来设计联想存储器。我们将设计的联想存储器也称为*按内容寻址的存储器*，因为它能够按照所存储内容的一部分来检索数据。这种存储器同标准的计算机存储器形成对比，后者是按照存储地址来寻找数据的。举个例子，假设我们有一个按内容寻址的数据库，它包含雇员的姓名、地址、电话号码。我们能够通过只提供雇员的名字(或部分名字)来得到一个完整的数据单元。按内容寻址存储器就像第 7 章所描述的自联想存储器一样有效(参见 7.2.4 节)，只不过在这一章我们将使用递归 Hopfield 网络而不是线性联想器。

假设我们要在 Hopfield 网络中存储一组原型模式。当向网络输入一个模式时，网络会产生一个与输入模式最相似的存储模式。对输入模式指定一个初始网络输出。网络最终输出应收敛于与输入模式最接近的原型模式。要这种情形发生，原型模式必须是 Lyapunov 函数的极小值。

假设原型模式为

$$\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_Q\} \tag{18.61}$$

每一个向量由 $S$ 个元素组成，每个元素值为 1 或 -1。假设 $Q << S$，那么状态空间就很大且原型模式在状态空间中均匀分布，彼此不接近。 18-16

为了使 Hopfield 网络能够回忆起原型模式，这些模式必须是 Lyapunov 函数的极小值。

既然高增益的 Lyapunov 函数是二次函数，我们需要使原型模式成为一个合适二次函数的限制极小值。我们建议使用下面的二次性能指数：

$$J(\mathbf{a}) = -\frac{1}{2}\sum_{q=1}^{Q}([\mathbf{p}_q]^T\mathbf{a})^2 \tag{18.62}$$

如果向量 $\mathbf{a}$ 的元素被限制为 $\pm 1$，我们将会证明这个函数在原型模式处取极小值。

假设原型模式是正交的。我们计算一个原型模式的性能指数：

$$J(\mathbf{p}_j) = -\frac{1}{2}\sum_{q=1}^{Q}([\mathbf{p}_q]^T\mathbf{p}_j)^2 = -\frac{1}{2}([\mathbf{p}_j]^T\mathbf{p}_j)^2 = -\frac{S}{2} \tag{18.63}$$

式中的第二个等号是由于原型模式的正交性。最后一个等号是由于 $\mathbf{p}_j$ 的全部元素为 $\pm 1$。

下面计算一个随机输入模式 $\mathbf{a}$ 的性能指数。我们假定输入模式与任何原型模式都不相似。式(18.62)中每一项都是一个原型模式与输入模式的内积。输入模式与原型模式越接近，

则内积越大；反之，则越小。因而，当 **a** 不接近原型模式时，$J(\mathbf{a})$将是最大的(最小负值)；反之，$J(\mathbf{a})$将是最小的(最大负值)。

我们已经找到了一个能够精确指示按内容寻址存储器性能的二次函数。下一步就是选择权值矩阵 **W** 和偏置值 **b**，这样就能使 Hopfield Lyapunov 函数 $V$ 等价于二次性能指数 $J$。

如果我们用有监督的 Hebb 规则来计算权值矩阵(用输入模式作目标模式)，便得

$$\mathbf{W} = \sum_{q=1}^{Q} \mathbf{p}_q (\mathbf{p}_q)^T \tag{18.64}$$

并设置偏置值

$$\mathbf{b} = 0 \tag{18.65}$$

18-17 这样 Lyapunov 函数为

$$V(\mathbf{a}) = -\frac{1}{2}\mathbf{a}^T\mathbf{W}\mathbf{a} = -\frac{1}{2}\mathbf{a}^T\left[\sum_{q=1}^{Q}\mathbf{p}_q(\mathbf{p}_q)^T\right]\mathbf{a} = -\frac{1}{2}\sum_{q=1}^{Q}\mathbf{a}^T\mathbf{p}_q(\mathbf{p}_q)^T\mathbf{a} \tag{18.66}$$

上式又可写为

$$V(\mathbf{a}) = -\frac{1}{2}\sum_{q=1}^{Q}[(\mathbf{p}_q)^T\mathbf{a}]^2 = J(\mathbf{a}) \tag{18.67}$$

因此，Lyapunov 函数实际上就是按内容寻存储器的二次性能指数。Hopfield 网络的输出通常都会收敛于存储的原型模式(我们在后文讨论其他可能的收敛点)。

如第 7 章指出的那样，有监督的 Hebb 规则在原型模式之间存在明显相关的情况下效果并不好。在这种情况下，可采用仿逆函数技术。另一种设计技术超出本书的范围，可参考[LiMi89]。

在最好的情况下，原型模式是正交的，每一个原型模式都是网络的一个平衡点。但是，仍有可能存在很多其他平衡点。这样，网络就可能收敛于一个不是原型模式的平衡点。通常在使用 Hebb 规则时，存储模式的数目不能超过神经元数目的 15%。参考书目[LiMi89]讨论了更复杂的设计过程，这种过程可使假平衡点数目达到最小。

下一小节，我们将进一步分析平衡点的位置。

2. Hebb 规则

让我们近一步看一看当 Hebb 规则用来计算权值矩阵并且原型模式是正交的情况下，Hopfield 网络如何运算。(下面的分析接着第 7 章例题 P7.5 的讨论)。有监督的 Hebb 规则为

$$\mathbf{W} = \sum_{q=1}^{Q}\mathbf{p}_q(\mathbf{p}_q)^T \tag{18.68}$$

如果把原型向量 $\mathbf{p}_j$ 用于网络中，那么有

18-18

$$\mathbf{W}\mathbf{p}_j = \sum_{q=1}^{Q}\mathbf{p}_q(\mathbf{p}_q)^T\mathbf{p}_j = \mathbf{p}_j(\mathbf{p}_j)^T\mathbf{p}_j = S\mathbf{p}_j \tag{18.69}$$

其中第二个等式成立是因为原型模式是正交的，第三个等式成立是由于 $\mathbf{p}_j$ 的每个元素不是 1 就是 $-1$。等式(18.69)有如下形式：

$$\mathbf{W}\mathbf{p}_j = \lambda\mathbf{p}_j \tag{18.70}$$

因此，每个原型向量等是权值矩阵的特征向量且它们有共同的特征值 $\lambda = S$。特征值 $\lambda = S$ 的特征向量空间 $X$ 为

$$X = \mathrm{span}\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_Q\} \tag{18.71}$$

这个空间包含所有能写成原型向量线性组合的向量。这就是说，一个向量 **a** 只要是原型向量的线性组合就是特征向量。

$$\begin{aligned}\mathbf{Wa} &= \mathbf{W}\{\alpha_1\mathbf{p}_1 + \alpha_2\mathbf{p}_2 + \cdots + \alpha_Q\mathbf{p}_Q\} \\ &= \{\alpha_1\mathbf{Wp}_1 + \alpha_2\mathbf{Wp}_2 + \cdots + \alpha_Q\mathbf{Wp}_Q\} \\ &= \{\alpha_1 S\mathbf{p}_1 + \alpha_2 S\mathbf{p}_2 + \cdots + \alpha_Q S\mathbf{p}_Q\} \\ &= S\{\alpha_1\mathbf{p}_1 + \alpha_2\mathbf{p}_2 + \cdots + \alpha_Q\mathbf{p}_Q\} = S\mathbf{a}\end{aligned} \tag{18.72}$$

特征值 $\lambda = S$ 的特征向量空间是一个 $Q$ 维的空间(假定原型向量线性无关)。

整个 $\Re^S$ 空间可分解为两个不相交的集合[Brog85]，

$$\Re^S = X \cup X^{\perp} \tag{18.73}$$

其中 $X^{\perp}$ 是 $X$ 的正交补集。(这对任何集合 $X$ 都成立，不仅包括此处我们所考虑的集合。) $X^{\perp}$ 中的每个向量均与 $X$ 中每一个向量正交。这就是说对于任何向量 $\mathbf{a} \in X^{\perp}$，

$$(\mathbf{p}_q)^T\mathbf{a} = 0, \quad q = 1,2,\cdots,Q \tag{18.74}$$

因此，如果 $\mathbf{a} \in X^{\perp}$，

$$\mathbf{Wa} = \sum_{q=1}^{Q}\mathbf{p}_q(\mathbf{p}_q)^T\mathbf{a} = \sum_{q=1}^{Q}(\mathbf{p}_q \cdot 0) = \mathbf{0} = 0 \cdot \mathbf{a} \tag{18.75}$$

所以 $X^{\perp}$ 定义了重特征值 $\lambda = 0$ 的一个特征向量空间。 18-19

概括起来，权值矩阵有两个特征值，$S$ 和 0。特征值 $S$ 的特征向量空间是由原型向量所决定的。特征值 0 的特征向量空间是原型向量所生成的空间的正交补集。

既然高增益的 Lyapunov 函数 $V$ 的赫森矩阵是

$$\nabla^2 V = -\mathbf{W} \tag{18.76}$$

那么 $\nabla^2 V$ 的特征值就是 $-S$ 和 0。

高增益的 Lyapunov 函数是一个二次函数。因而，赫森矩阵的特征值就决定了它的形状。因为第一个特征值是负值，$V$ 在 $X$ 中将有一负曲率。又由于第二个特征值是 0，$V$ 在 $X^{\perp}$ 中将有零曲率。

这些结果对 Hopfield 网络的响应说明了什么？因为 $V$ 在 $X$ 中有负的曲率，Hopfield 网络的轨迹将会落入包含在 $X$ 中的超立方体 $\{\mathbf{a}: -1 < a_i < 1\}$ 的角上。

注意，如果我们用 Hebb 规则计算权值矩阵，对每一个原型向量来说，Lyapunov 函数至少有两个极小值。如果 $\mathbf{p}_q$ 是个原型向量，那么 $-\mathbf{p}_q$ 将也在由原型向量所生成的空间里。因此，每个原型向量的负值将是包含在 $X$ 中的超立方体 $\{\mathbf{a}: -1 < a_i < 1\}$ 的一个角。除此之外，还有一些不与原型模式对应的 Lyapunov 函数的极小值。

**伪模式** $V$ 的极小值在包含在 $X$ 中的超立方体 $\{\mathbf{a}: -1 < a_i < 1\}$ 的角上。这些不仅包含原型模式，而且也包含它们的某些线性组合。那些不是原型模式的极小值通常称为伪模式。Hopfield 网络设计的目标就是使伪模式的数目减到最小并把每一个原型模式的吸引区尽可能地扩大。[LiMi89]中描述了一种保证达到最少伪模式的设计方法。

为了解释这些原理，再次考虑我们曾经讨论过的二阶矩阵的例子，在这个例子中连接矩阵为

$$\mathbf{W} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{18.77}$$

假设用 Hebb 规则设计并只有一个原型模式(显然是一个没有实际应用意义的例子) 18-20

$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{18.78}$$

那么

$$\mathbf{W} = \mathbf{p}_1(\mathbf{p}_1)^T = \begin{bmatrix} 1 \\ 1 \end{bmatrix} [1 \quad 1] = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \tag{18.79}$$

注意

$$\mathbf{W}' = \mathbf{W} - \mathbf{I} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{18.80}$$

与我们的初始连接矩阵对应。详细情况见下一小节。

高增益的 Lyapunov 函数是

$$V(\mathbf{a}) = -\frac{1}{2}\mathbf{a}^T\mathbf{W}\mathbf{a} = -\frac{1}{2}\mathbf{a}^T\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}\mathbf{a} \tag{18.81}$$

$V(\mathbf{a})$的赫森矩阵是

$$\nabla^2 V(\mathbf{a}) = -\mathbf{W} = \begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix} \tag{18.82}$$

它的特征值为

$$\lambda_1 = -S = -2 \quad 和 \quad \lambda_2 = 0 \tag{18.83}$$

相应的特征向量为

$$\mathbf{z}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad 和 \quad \mathbf{z}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \tag{18.84}$$

与特征值 $-S$ 对应的第一个特征向量代表着由原型向量生成的空间：

$$X = \{\mathbf{a}: a_1 = a_2\} \tag{18.85}$$

18-21 与特征值 0 对应的第二个特征向量代表第一个特征向量的正交补：

$$X^{\perp} = \{\mathbf{a}: a_1 = -a_2\} \tag{18.86}$$

Lyapunov 函数如图 18-10 所示。

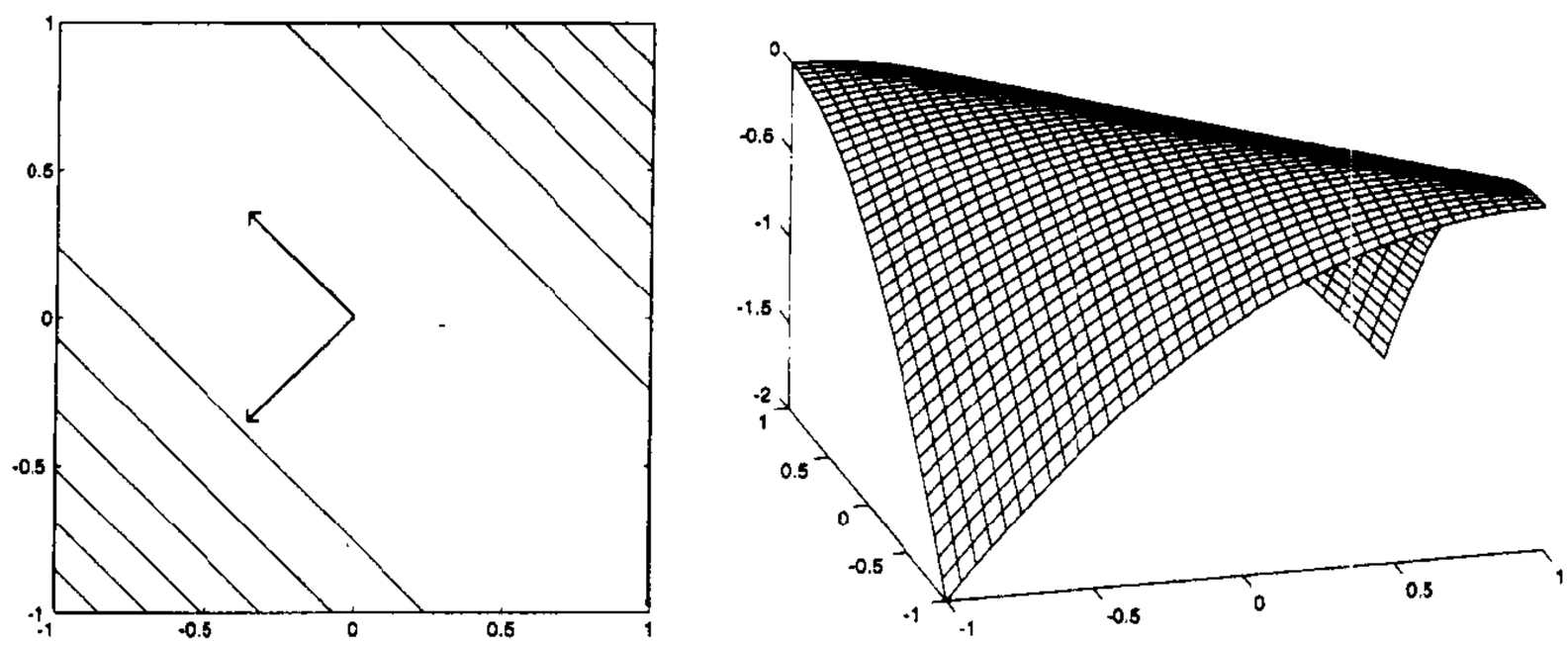



图 18-10 Lyapunov 函数实例

这个曲面有一个从左上角到右下角的直凸起。这表示 $X^{\perp}$曲率为 0 的区域。初始条件是凸起的左边还是右边决定收敛点为

$$\mathbf{a} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad 或 \quad \mathbf{a} = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \tag{18.87}$$

如果初始条件是恰在凸起上，那么网络就会稳定在开始处。情况就像我们原来的例子(参见图 18-9)，只不过在那种情况下，在斜的凸起上的初始点最终收敛于原点，而不是稳定在开始的地方(参见图 18-6)。无论初始点在凸起的左边或右边，在两个系统中都会收敛于原型设计点。因此，我们最初讨论的那个系统与用零对角线元素的系统在每一个重要方面其收敛情况是一致的。在下一小节，我们将进一步考察这一点。

3. Lyapunov 曲面

在 Hopfield 网络的很多讨论中，权值矩阵的对角线元素被设为 0。在这一小节中我们将分析这样的运算对 Lyapunov 曲面的影响。(可参见第 7 章习题 E7.5。)

对于按内容寻址的存储器网络来说，所有的权值矩阵对角线元素都为 $Q$(原型模式的个数)，因为 $\mathbf{p}_q$ 中的每个元素都为 $\pm 1$。因此，我们可以通过减去 $Q$ 与单位矩阵的乘积把对角线元素归 0: 18-22

$$\mathbf{W}' = \mathbf{W} - Q\mathbf{I} \tag{18.88}$$

让我们考查一下这种变换对 Lyapunov 函数有何影响。如果把这个新的权值矩阵乘以一个原型向量，求得

$$\mathbf{W}'\mathbf{p}_q = [\mathbf{W} - Q\mathbf{I}]\mathbf{p}_q = S\mathbf{p}_q - Q\mathbf{p}_q = (S - Q)\mathbf{p}_q \tag{18.89}$$

因此，$(S - Q)$是 $\mathbf{W}'$ 的特征值。相应的特征向量空间是由原型向量所生成的 $X$。

如果把这个新的权值矩阵乘以来自 $X$ 正交补集中的向量 $\mathbf{a}(\mathbf{a} \in X^{\perp})$，求得

$$\mathbf{W}'\mathbf{a} = [\mathbf{W} - Q\mathbf{I}]\mathbf{a} = \mathbf{0} - Q\mathbf{a} = -Q\mathbf{a} \tag{18.90}$$

因此，$-Q$ 是 $\mathbf{W}'$ 的特征值，相应的特征向量空间是 $X^{\perp}$。

概括起来说，$\mathbf{W}'$ 中的特征向量和 $\mathbf{W}$ 中的特征向量是一样的，不同的是 $\mathbf{W}'$ 的特征值为 $(S - Q)$和 $-Q$，而不是 $S$ 和 0。因此，修改后的 Lyapunov 函数$\nabla^2 V'(\mathbf{a}) = -\mathbf{W}'$ 的赫森矩阵的特征值为 $-(S - Q)$和 $Q$。

这就暗示着能量曲面在 $X$ 中有负的曲率而在 $X^{\perp}$有正的曲率。与原来的 Lyapunov 函数形成对照，它在 $X$ 中有负的曲率，在 $X^{\perp}$中曲率为 0。

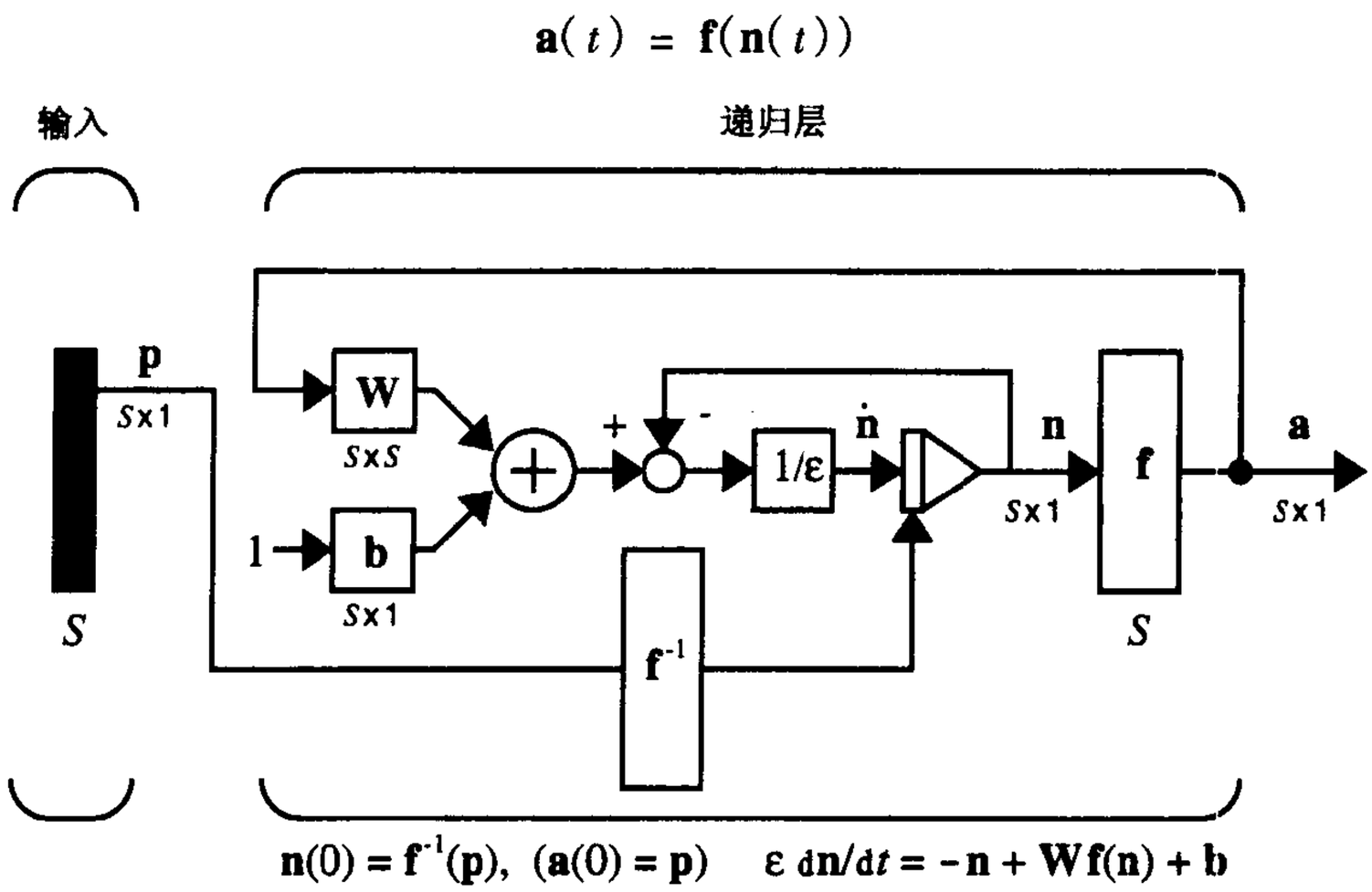
比较图 18-9 和图 18-10，可以发现把权值矩阵对角线元素设为 0 对 Lyapunov 函数的影响。在系统性能方面，这种改变只有很小的影响。如果把 Hopfield 网络的初始条件设在离开直线 $a_1 = -a_2$ 的任何一处，在两种情况下，网络的输出最终都会收敛于超立方体$\{\mathbf{a}: -1 < a_i < 1\}$的一个角上，这个超立方体包含两个点 $\mathbf{a} = [1 \quad 1]^T$ 和 $\mathbf{a} = [-1 \quad -1]^T$。

如果初始条件恰好落在直线 $a_1 = -a_2$ 上，且使用权值矩值 $\mathbf{W}$，那么网络的输出将保持为常数。如果初始条件恰好落在直线 $a_1 = -a_2$ 上，但是使用权值矩阵 $\mathbf{W}'$，那么网络的输出收敛到在原点处的鞍点上(如图 18-16 所示)。既然网络的输出不收敛于 Lyapunov 函数的极小值，这些结果就没有用。当然只有初始条件恰好落在直线 $a_1 = -a_2$ 上，网络才可能收敛于一个鞍点，在实际上这几乎是不可能的。 18-23

## 18.3 小结

### Hopfield 模型

$$\varepsilon \frac{d\mathbf{n}(t)}{dt} = -\mathbf{n}(t) + \mathbf{W}\mathbf{a}(t) + \mathbf{b}$$




Lyapunov 函数

$$V(\mathbf{a}) = -\frac{1}{2}\mathbf{a}^T\mathbf{W}\mathbf{a} + \sum_{i=1}^{S}\left\{\int_0^{a_i} f^{-1}(u)\mathrm{d}u\right\} - \mathbf{b}^T\mathbf{a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}V(\mathbf{a}) = -\varepsilon\sum_{i=1}^{S}\left(\frac{\mathrm{d}}{\mathrm{d}a_i}[f^{-1}(a_i)]\right)\left(\frac{\mathrm{d}a_i}{\mathrm{d}t}\right)^2$$

$$\text{如果}\frac{\mathrm{d}}{\mathrm{d}a_i}[f^{-1}(a_i)] > 0, \quad \text{则}\frac{\mathrm{d}}{\mathrm{d}t}V(\mathbf{a}) \leqslant 0$$

**不变集**

不变集由平衡点组成：

18-24

$$L = Z = \{\mathbf{a}: \mathrm{d}\mathbf{a}/\mathrm{d}t = \mathbf{0}, \quad \mathbf{a}\text{属于 } G \text{ 的闭包}\}$$

**Hopfield 吸引子**

平衡点是静止点：

$$\text{如果}\frac{\mathrm{d}\mathbf{a}(t)}{\mathrm{d}t} = \mathbf{0}, \quad \text{则}\nabla V(\mathbf{a}) = \mathbf{0}$$

$$\nabla V(\mathbf{a}) = [-\mathbf{W}\mathbf{a} + \mathbf{n} - \mathbf{b}] = -\varepsilon\left[\frac{\mathrm{d}\mathbf{n}(t)}{\mathrm{d}t}\right]$$

**高增益的 Lyapunov 函数**

$$V(\mathbf{a}) = -\frac{1}{2}\mathbf{a}^T\mathbf{W}\mathbf{a} - \mathbf{b}^T\mathbf{a}$$

$$\nabla^2 V(\mathbf{a}) = -\mathbf{W}$$

**按内容寻址存储器**

$$\mathbf{W} = \sum_{q=1}^{Q}\mathbf{p}_q(\mathbf{p}_q)^T \text{ 和 } \mathbf{b} = \mathbf{0}$$

**能量曲面（正交原型模式）**

$\nabla^2 V(\mathbf{a}) = -\mathbf{W}$ 的特征向量是：

$\lambda_1 = -S$， 对应特征向量空间 $X = \text{span}\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_Q\}$

$\lambda_2 = 0$， 对应特征向量空间 $X^{\perp}$

（$X^{\perp}$ 定义为，对任意向量 $\mathbf{a} \in X^{\perp}$，$(\mathbf{p}_q)^T\mathbf{a} = 0, q = 1, 2, \cdots, Q$）

**轨迹(正交原型模式)**

因为第一个特征值是负的，$V(\mathbf{a})$在 $X$ 中有负曲率。又因为第二个特征值为 0，$V(\mathbf{a})$在 $X^{\perp}$中有 0 曲率。因为 $V(\mathbf{a})$在 $X$ 中有负曲率，Hopfield 网络的轨迹会落入包含在 $X$ 中的超立方体$\{\mathbf{a}: -1 < a_i < 1\}$的角上。 18-25

## 18.4 例题

**P18.1** 假设有二进制原型向量

$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$$

（i）定义一个连续型的 Hopfield 网络(指定连接权值)来识别这些模式，使用 Hebb 规则。

（ii）求这个网络的高增益 Lyapunov 函数的赫森矩阵，其特征值和特征向量是什么?

（iii）假设增益很大，Hopfield 网络的平衡点是什么?

**解**

（i）首先使用有监督的 Hebb 规则，从参考向量中计算权值矩阵。

$$\mathbf{W} = \mathbf{p}_1(\mathbf{p}_1)^T + \mathbf{p}_2(\mathbf{p}_2)^T = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix}$$

化简得

$$\mathbf{W} = \begin{bmatrix} 2 & 0 & 0 & 2 \\ 0 & 2 & -2 & 0 \\ 0 & -2 & 2 & 0 \\ -2 & 0 & 0 & 2 \end{bmatrix}$$

（ii）由式(18.54)高增益的 Lyapunov 函数的赫森矩阵是权值矩阵的负值：

$$\nabla^2 V(\mathbf{a}) = \begin{bmatrix} -2 & 0 & 0 & -2 \\ 0 & -2 & 2 & 0 \\ 0 & 2 & -2 & 0 \\ 2 & 0 & 0 & -2 \end{bmatrix}$$ 18-26

原型模式是正交的($[\mathbf{p}_1]^T\mathbf{p}_2 = 0$)，所以特征值为 $\lambda_1 = -S = -4$ 和 $\lambda_2 = 0$。对应$\lambda_1 = -4$ 的特征向量空间为

$$X = \text{span}\{\mathbf{p}_1, \mathbf{p}_2\}$$

对应 $\lambda_2 = 0$ 的特征空间是 $X$ 的正交补集：

$$X^{\perp} = \text{span}\left\{\begin{bmatrix}1\\1\\1\\1\end{bmatrix}, \begin{bmatrix}1\\-1\\-1\\1\end{bmatrix}\right\}$$

其中我们选了两个既垂直于 $\mathbf{p}_1$ 又垂直于 $\mathbf{p}_2$ 的向量。

(iii) 稳定点分别是 $\mathbf{p}_1$，$\mathbf{p}_2$，$-\mathbf{p}_1$，$-\mathbf{p}_2$，因为原型模式的负值也是平衡点。也可能还有其他平衡点，如果超立方体其他的角 $\text{span}\{\mathbf{p}_1, \mathbf{p}_2\}$ 中。超立方体总共有 $2^4 = 16$ 个角，四个角落入 $X$ 中，四个角落入 $X^{\perp}$ 中，其他的角部分在 $X$ 中部分在 $X^{\perp}$ 中。

**P18.2** 考虑一个具有如下权值矩阵和偏置值的高增益 Hopfield 网络：

$$\mathbf{W} = \begin{bmatrix}-1 & -1\\-1 & -1\end{bmatrix} \quad 和 \quad \mathbf{b} = \begin{bmatrix}1\\-1\end{bmatrix}$$

(i) 描绘这个网络的高增益 Lyapunov 函数的轮廓图。

(ii) 如果网络的初始条件为 $[1 \quad 1]^T$，网络将收敛于何处？

**解**

(i) 首先考虑高增益的 Lyapunov 函数

$$V(\mathbf{a}) = -\frac{1}{2}\mathbf{a}^T\mathbf{W}\mathbf{a} - \mathbf{b}^T\mathbf{a}$$

赫森矩阵是

$$\nabla^2 V(\mathbf{a}) = -\mathbf{W} = \begin{bmatrix}1 & 1\\1 & 1\end{bmatrix}$$

18-27 下一步，我们需要计算特征值和特征向量：

$$|\nabla^2 V(\mathbf{a}) - \lambda\mathbf{I}| = \left|\begin{bmatrix}1-\lambda & 1\\1 & 1-\lambda\end{bmatrix}\right| = \lambda^2 - 2\lambda + 1 - 1 = \lambda(\lambda - 2)$$

特征值为 $\lambda_1 = 0$ 和 $\lambda_2 = 2$。

现在来求特征向量。对于 $\lambda_1 = 0$，

$$[\nabla^2 V(\mathbf{a}) - \lambda_1\mathbf{I}]\mathbf{z}_1 = \mathbf{0}$$

因而

$$\begin{bmatrix}1 & 1\\1 & 1\end{bmatrix}\mathbf{z}_1 = \mathbf{0} \quad 或 \quad \mathbf{z}_1 = \begin{bmatrix}1\\-1\end{bmatrix}$$

同理，对 $\lambda_2 = 2$，

$$[\nabla^2 V(\mathbf{a}) - \lambda_2\mathbf{I}]\mathbf{z}_2 = \mathbf{0}$$

因而

$$\begin{bmatrix}-1 & 1\\1 & -1\end{bmatrix}\mathbf{z}_2 = \mathbf{0} \quad 或 \quad \mathbf{z}_2 = \begin{bmatrix}1\\1\end{bmatrix}$$

所以项

$$-\frac{1}{2}\mathbf{a}^T\mathbf{W}\mathbf{a}$$

在 $\mathbf{z}_1$ 方向上曲率为 0，在 $\mathbf{z}_2$ 方向上曲率为负。

现在来说明线性项。首先画出没有线性项的轮廓图，如图 18-11 所示。

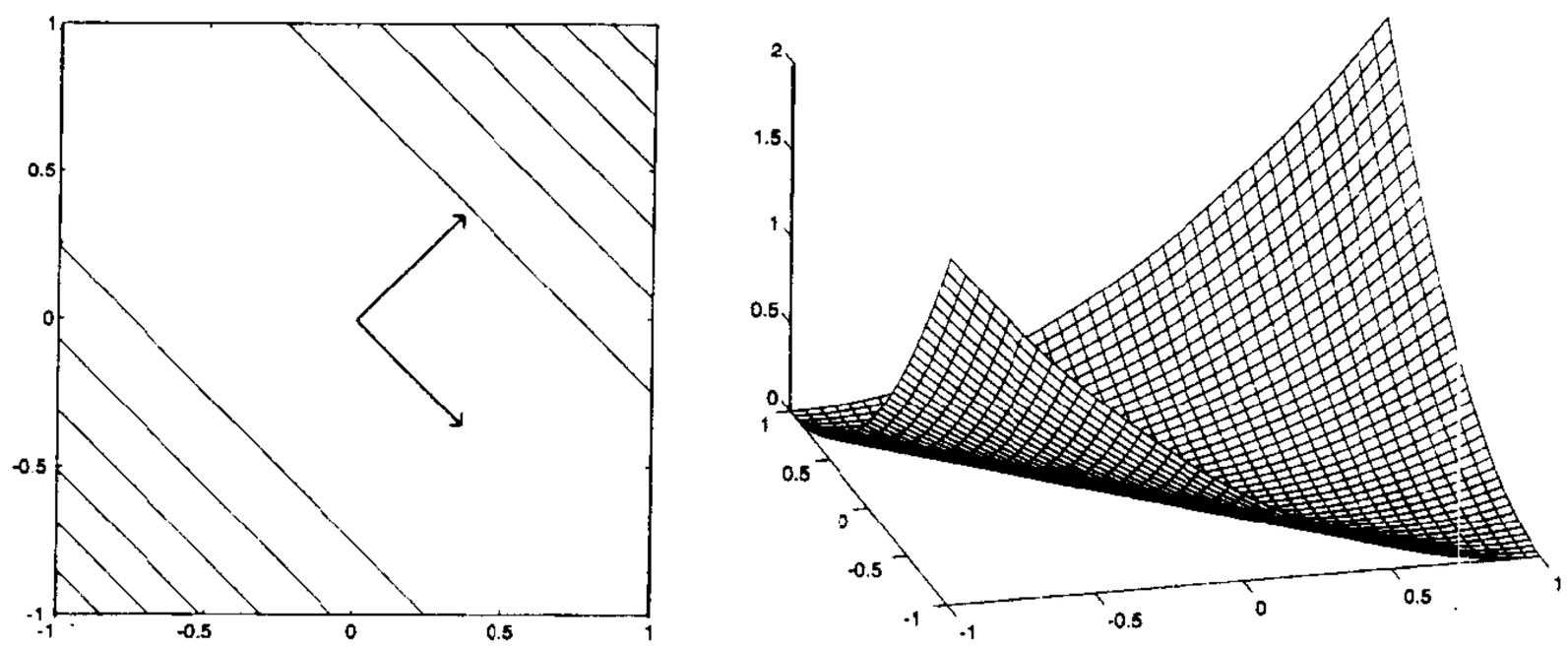


图 18-11 没有线性项的轮廓图

线性项会在方向

$$\mathbf{b} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

上引起一个负的斜率。因而所有的地方都会向$[1 \quad -1]^T$弯曲，如图 18-12 所示。 18-28

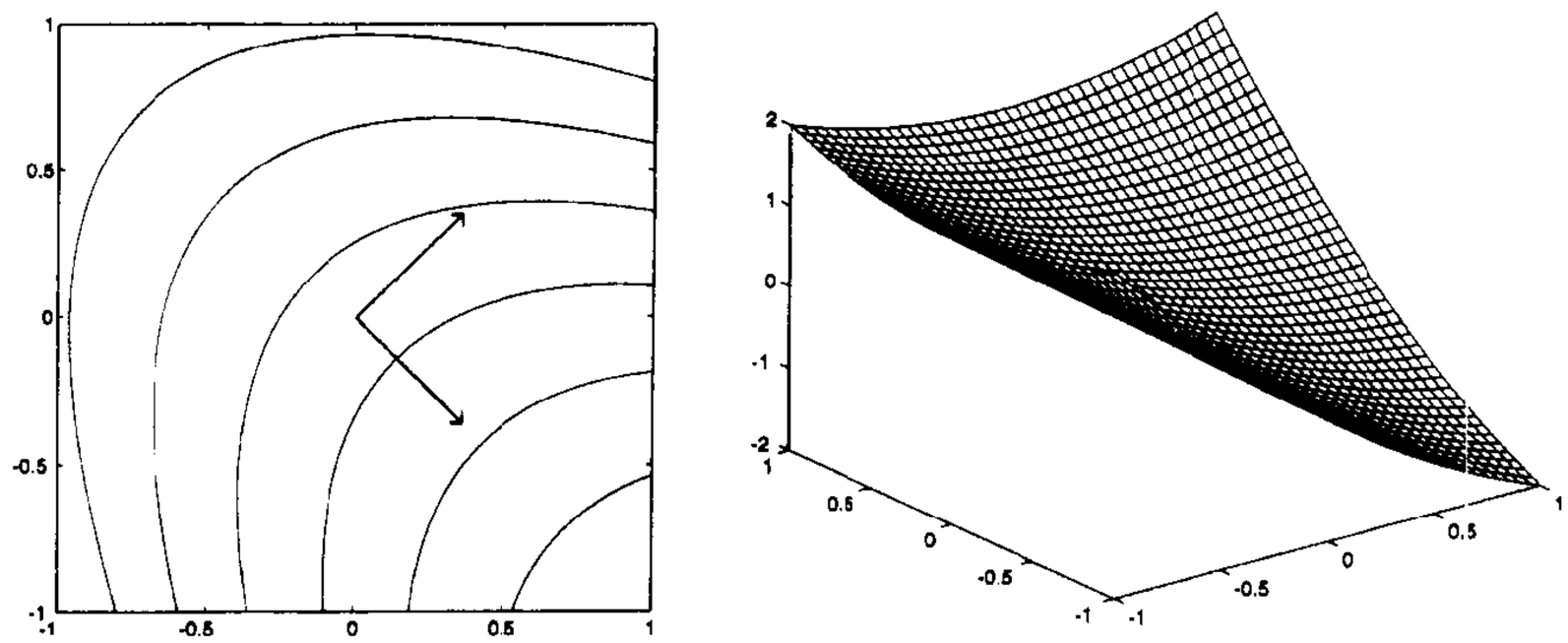


图 18-12 包含线性项的轮廓图

（ii）不论初使条件如何，所有的轨迹都会收敛于$[1 \quad -1]^T$。在图 18-12 中我们可以看出能量函数仅有一个极小值点，它位于$[1 \quad -1]^T$处。(记住网络的输出被限制在超立方体$\{\mathbf{a}: -1 < a_i < 1\}$内。)

**P18.3** 考虑下面的原型向量：

$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

（i）设计一个 Hopfield 网络来识别这两个模式。

（ii）求高增益的 Lyapunov 函数的赫森矩阵。它的特征值和特征向量是什么？ 18-29

（iii）Hopfield 网络的稳定点是什么(假定网络具有很大的增益)？吸引区是什么？

（iv）网络对模式识别的效果如何？

**解**

（i）我们使用 Hebb 规则来求权值矩阵：

$$\mathbf{W} = \mathbf{p}_1(\mathbf{p}_1)^T + \mathbf{p}_2(\mathbf{p}_2)^T = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

偏置值设置为 0：

$$\mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

（ii）高增益的 Lyapunov 函数的赫森矩阵是权值矩阵的负值：

$$\nabla^2 V(\mathbf{a}) = -\mathbf{W} = \begin{bmatrix} -2 & 0 \\ 0 & -2 \end{bmatrix}$$

凭观察，知道有重特征值

$$\lambda_1 = \lambda_2 = -S = -2$$

特征向量为

$$\mathbf{z}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad 和 \quad \mathbf{z}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

或者任意线性组合。（整个$\Re^2$就是特征值 $\lambda = -2$ 的特征向量空间。）

（iii）在第 8 章我们知道当赫森矩阵的特性值相等时，轮廓线将是环形的。因为特征值为负，所以函数将在原点有惟一的一个极大值。在超立方体$\{\mathbf{a}: -1 < a_i < 1\}$的四个角上有
18-30 4 个极小值。高增益的 Lyapunov 函数如图 18-13 示。

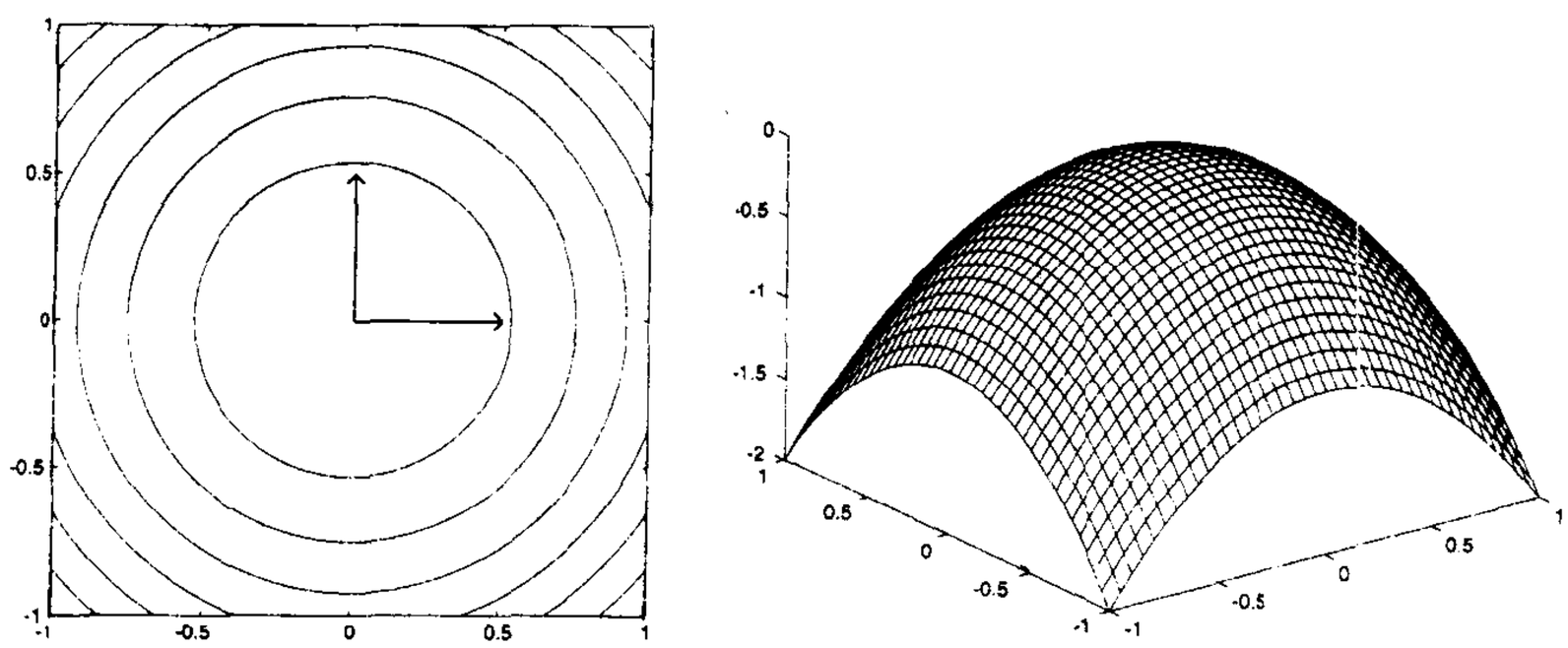



图 18-13 例题 P18.3 的高增益 Lyapunov 函数

总共有 9 个静止点。我们可用 LaSalle 不变性定理的推论来证明原点处的极大值吸引区只包含原点自己。因此它不是一个稳定的平衡点。鞍点的吸引区为直线。（例如，在$[-1 \quad 0]^T$处的鞍点吸引区为$a_1$的负轴线。）超立方体的 4 个角是仅有二维吸引区的吸引子。每一个角的吸引区是超立方体的相应象限。图 18-14 显示了低增益的 Lyapunov 函数（增益系数 $\gamma = 1.4$）并说明收敛到一个鞍点和一个极小值的情况。

（iv）网络在模式识别这个问题上做得并不十分好。它不仅识别两个原型模式，也“识
18-31 别”出超立方体的其他两个角。网络将会收敛到距输入模式最近的一个角，尽管我们只想让

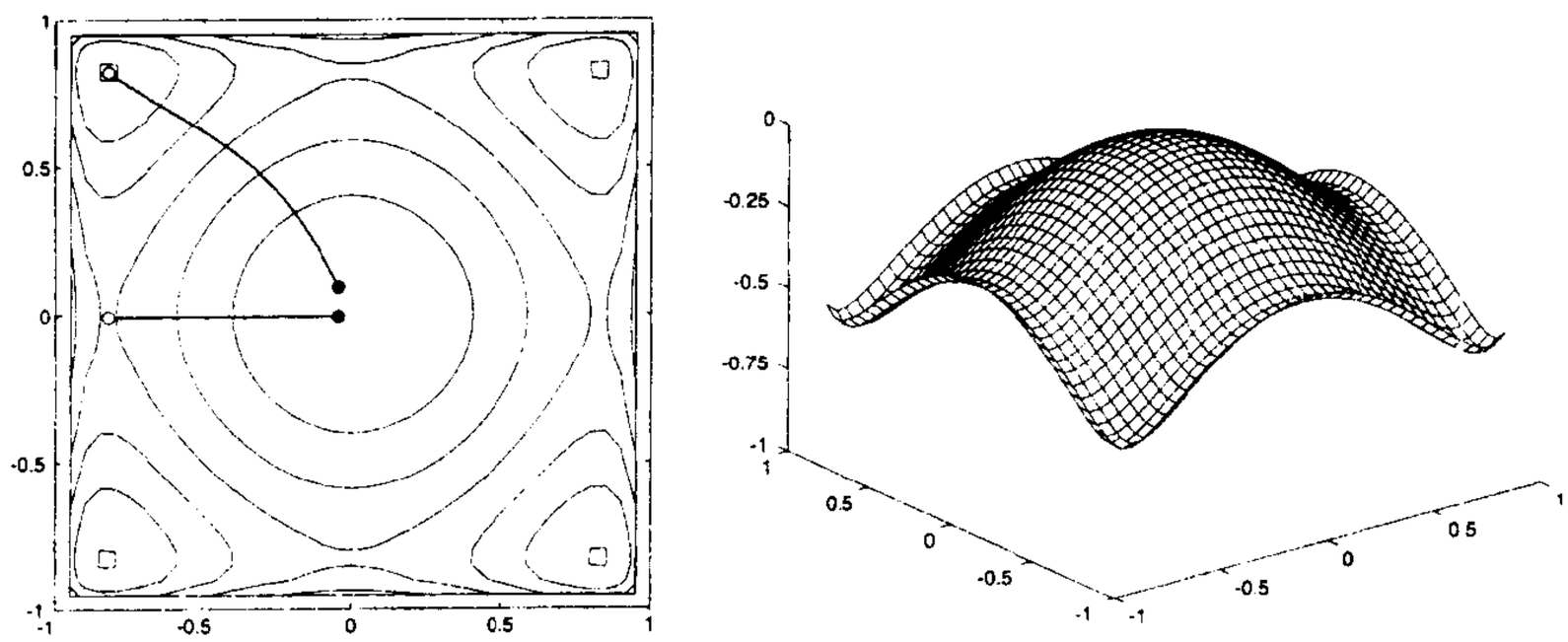



图 18-14 例题 P18.3 的 Lyapunov 函数

它存储两个原型模式。因为每一种可能的两位模式都被存储了，所以网络并非十分有用。这是我们不希望出现的，因为根据 Hebb 规则，期望的存储模式只是神经元数目的 15%。既然我们只有两个神经元，所以不希望存储许多模式。习题 E18.2 提供了一个更好的网络。

**P18.4** 一个 Hopfield 网络具有下面的高增益 Lyapunov 函数：

$$V(\mathbf{a}) = -\frac{1}{2}(7(a_1)^2 + 12a_1a_2 - 2(a_2)^2)$$

(i) 求权值矩阵。

(ii) 求 Lyapunov 函数的梯度向量。

(iii) 求 Lyapunov 函数的赫森矩阵。

(iv) 画出 Lyapunov 函数的轮廓图。

(v) 画出 $V(\mathbf{a})$在初始条件为$[0.25 \quad 0.25]^T$ 时使用最速下降算法所经过的路径。

**解**

(i) $V(\mathbf{a})$是一个二次函数，可重写成

$$V(\mathbf{a}) = -\frac{1}{2}(7(a_1)^2 + 12a_1a_2 - 2(a_2)^2) = -\frac{1}{2}\mathbf{a}^T\begin{bmatrix} 7 & 6 \\ 6 & -2 \end{bmatrix}\mathbf{a}$$

因此权值矩阵是

$$\mathbf{W} = \begin{bmatrix} 7 & 6 \\ 6 & -2 \end{bmatrix}$$

(ii) 因为 $V(\mathbf{a})$是一个二次函数，我们可用式(8.38)来求梯度：

$$\nabla V(\mathbf{a}) = -\begin{bmatrix} 7 & 6 \\ 6 & -2 \end{bmatrix}\mathbf{a}$$

(iii) 由式(8.39)，赫森矩阵为

$$\nabla^2 V(\mathbf{a}) = -\begin{bmatrix} 7 & 6 \\ 6 & -2 \end{bmatrix} = \begin{bmatrix} -7 & -6 \\ -6 & 2 \end{bmatrix} \qquad \boxed{18\text{-}32}$$

(iv) 下面计算特征值：

$$|\nabla^2 V(\mathbf{a}) - \lambda\mathbf{I}| = \left|\begin{bmatrix} -7-\lambda & -6 \\ -6 & 2-\lambda \end{bmatrix}\right| = \lambda^2 + 5\lambda - 50 = (\lambda + 10)(\lambda - 5)$$

特征值为 $\lambda_1 = -10$ 和 $\lambda_2 = 5$。

现在求特征向量。对 $\lambda_1 = -10$,

$$[\nabla^2 V(\mathbf{a}) - \lambda_1 \mathbf{I}]\mathbf{z}_1 = \mathbf{0}$$

因此

$$\begin{bmatrix} 3 & -6 \\ -6 & 12 \end{bmatrix}\mathbf{z}_1 = \mathbf{0} \quad 或 \quad \mathbf{z}_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

类似地，对 $\lambda_2 = 5$,

$$|\nabla^2 V(\mathbf{a}) - \lambda_2 \mathbf{I}|\mathbf{z}_2 = \mathbf{0}$$

因而

$$\begin{bmatrix} -12 & -6 \\ -6 & -3 \end{bmatrix}\mathbf{z}_2 = \mathbf{0} \quad 或 \quad \mathbf{z}_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

注意，这是一个有鞍点的例子，因为 $\lambda_1 < 0 < \lambda_2$。沿着 $\mathbf{z}_1$ 曲率为负，沿着 $\mathbf{z}_2$ 曲率为正。高增益的 Lyapunov 函数图见图 18-15。

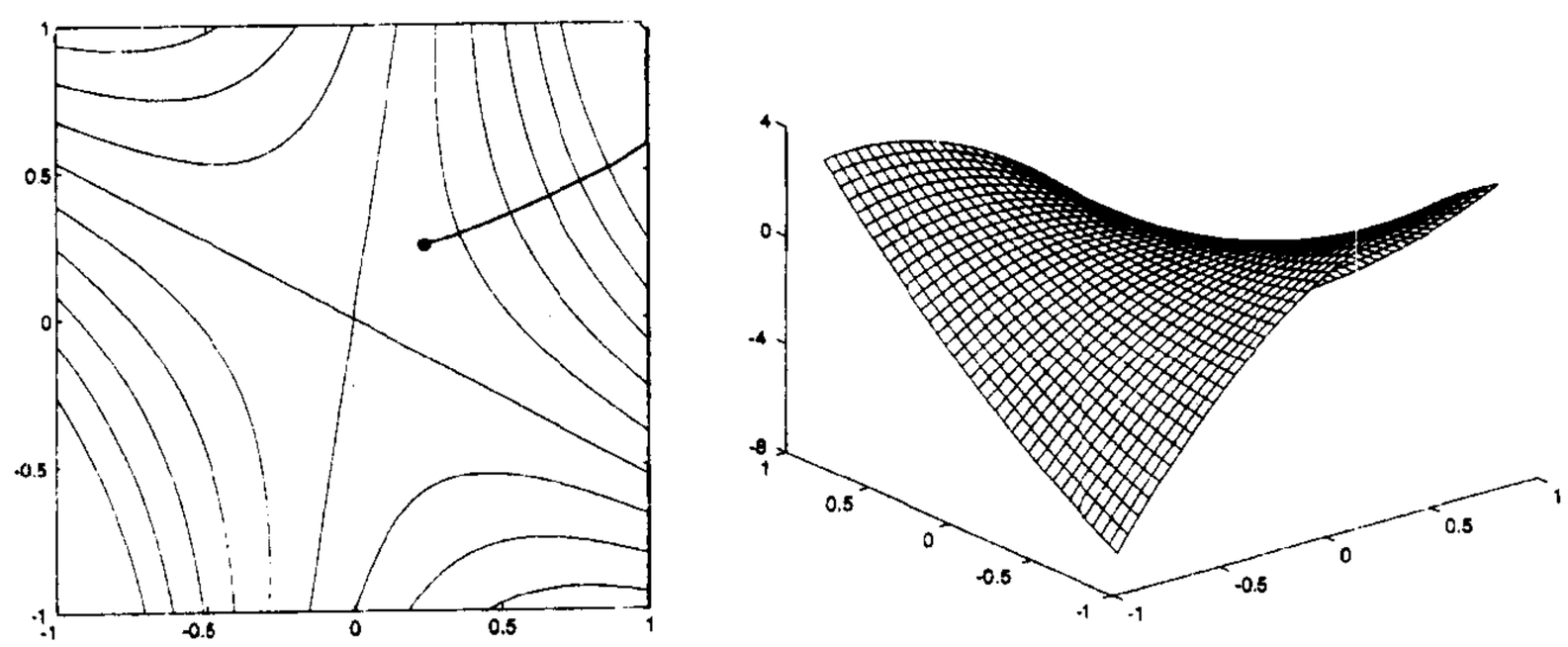


图 18-15 高增益 Lyapunov 函数和最速下降轨迹

18-33 （v）最速下降路径沿着梯度为负的方向并与轮廓线垂直，就像在第 9 章所见到的一样。当轨迹到达超立方体的边缘时，它将沿着边线下落到极小值。最后结果见图 18-15。

高增益的 Lyapunov 函数只是一个近似，因为它假设有无限大的增益。作为比较，图 18-16 画出了增益系数为 0.5 时的 Lyapunov 函数和 Hopfield 轨迹。

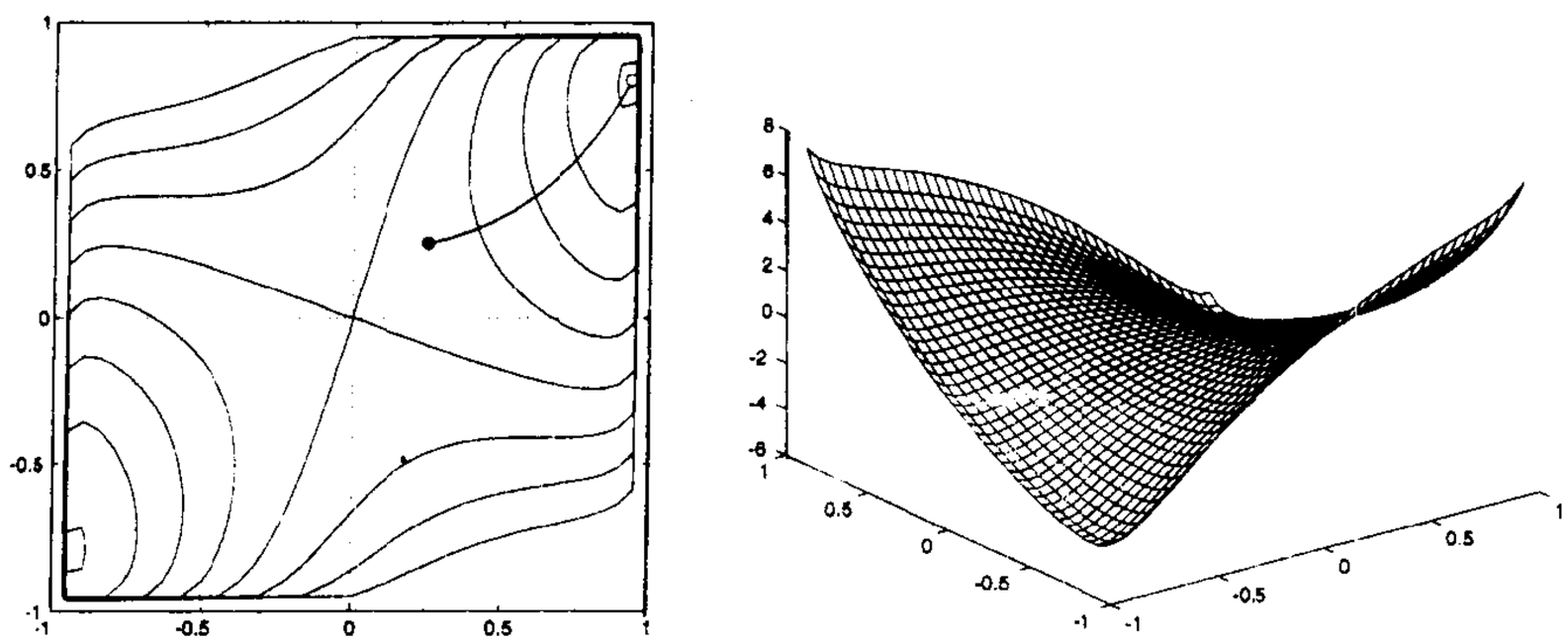


图 18-16 Lyapunov 函数与 Hopfield 轨迹

**P18.5** 除了作按内容寻址存储器外，Hopfield 网络还用在其他应用中。其一是用于模数(A/D)转换[HoTa86]。模数转换器的功能是把一个模拟信号 $y$ 转换为二进制数串(0 和 1)。例如，一个两位的模数转换器能把模拟信号 $y$ 近似为

$$y \cong \sum_{i=1}^{2} a_i 2^{(i-1)} = a_1 + a_2 2$$

其中 $a_1$ 和 $a_2$ 的值为 0 或 1。(这个 A/D 转换器把模拟信号近似在 0 到 3 的范围内，分辨单位为 1。)Tank 和 Hopfield 建议采用下面的性能指数来进行 A/D 转换：

$$J(\mathbf{a}) = \frac{1}{2}\left(y - \sum_{i=1}^{2} a_i 2^{(i-1)}\right)^2 - \frac{1}{2}\left(\sum_{i=1}^{2} 2^{2(i-1)} a_i (a_i - 1)\right)$$

其中第一项表示 A/D 转换误差，第二项迫使 $a_1$，$a_2$ 取值为 0 或 1。

证明这个性能指数可改写为 Hopfield 网络的 Lyapunov 函数，并定义适当的权值矩阵和偏置值向量。 18-34

**解**

第一步是对性能指数的项进行展开：

$$\left(y - \sum_{i=1}^{2} a_i 2^{(i-1)}\right)^2 = y^2 - 2y\sum_{i=1}^{2} a_i 2^{(i-1)} + \sum_{j=1}^{2}\sum_{i=1}^{2} a_i a_j 2^{(i-1)+(i-1)}$$

$$\left(\sum_{i=1}^{2} 2^{2(i-1)} a_i (a_i - 1)\right) = \sum_{i=1}^{2} (a_i)^2 2^{2(i-1)} - \sum_{i=1}^{2} a_i 2^{2(i-1)}$$

如果把这些项代回到性能指数中，求出

$$J(\mathbf{a}) = \frac{1}{2}\left(y^2 + \sum_{j=1}^{2}\sum_{\substack{i=1\\ i\neq j}}^{2} a_i a_j 2^{(i-1)+(i-1)} + \sum_{i=1}^{2} a_i (2^{2(i-1)} - 2^i y)\right)$$

第一项不是 $\mathbf{a}$ 的函数，因此不影响极小值出现的位置，可予以忽略。

我们要证明这个性能指数采用高增益的 Lyapunov 函数形式：

$$V(\mathbf{a}) = -\frac{1}{2}\mathbf{a}^T\mathbf{W}\mathbf{a} - \mathbf{b}^T\mathbf{a}$$

如果

$$\mathbf{W} = \begin{bmatrix} 0 & -2 \\ -2 & 0 \end{bmatrix} \quad 和 \quad \mathbf{b} = \begin{bmatrix} y - \frac{1}{2} \\ 2y - 2 \end{bmatrix}$$

即为所求。

在这个 Hopfield 网络中，不像按内容寻址存储器，网络的输入是标量 $y$，它被用来计算偏置值向量。在按内容寻址存储器中，网络的输入是向量模式，它成为网络输出的初始条件。

注意，在网络中传输函数必须把输出限制在 $0 < a < 1$ 的范围内。一个可使用的传输函数是

$$f(n) = \frac{1}{(1 - e^{-\gamma n})}$$ 18-35

## 18.5 结束语

本章我们介绍了 Hopfield 模型，这是一种最有影响的神经网络结构。Hopfield 之所以有

重要影响的一个原因是他强调网络的实际应用。他说明如何用电路的形式实现网络。在早期曾建立过用 VLSI 实现的 Hopfield 型网络。

Hopfield 还解释了如何用这种网络来求解模式识别和最优化问题。Hopfield 对他的网络提出的其他一些应用有：按内容寻址存储器[Hopf82]，A/D 转换[TaHo86]以及线性规划和最优化，如货郎担问题[HoTa85]。

Hopfield 的一个主要贡献是用 Lyapunov 稳定原理来分析他的网络。他同时证明，对于高增益的放大器，他的网络的 Lyapunov 函数是一个可由网络最小化的二次函数。这就导至了一些设计过程。设计的思想是把给定的问题转化为一个可由网络求解的二次函数最小值问题。

Hopfield 网络是本书中讨论的最后一种网络。然而，我们并没有讨论完所有重要的神经网络结构。在下一章中我们将对你下一步应继续研究什么主题提出一些看法。 18-36

## 参考文献

[Ande72] J. Anderson, "A simple neural network generating an interactive memory ," *Mathematical Biosciences*, vol. 14, pp. 197 - 220, 1972.

Anderson 提出了联想存储器的一种"线性联想器"模型。该模型使用一种推广的 Hebb 规则，进行训练，学习在输入/输出向量之间建立关联。他主要强调了网络的生理学拟真。在同一时期 Kohonen 发表了与此密切相关的论文[Koho72]。他们两人是独立进行工作的。

[AnSi77] J. A. Anderson, J. W. Silverstein, S. A. Ritz and R. S. Jones, "Distinctive features, Categorical perception, and probability learning: Some applications of a neural model," *Psychological Review*, vol. 84, pp. 413 - 451, 1977.

这篇文章介绍"盒中脑状态"神经网络模型，它将线性联想器和递归连接结合起来形成一种更强有力的自联想系统。它使用非线性传输函数将网络的输出限制在超立方体中。

[CoGr83] M. A. Cohen and S. Grossberg, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 13, no. 5, pp. 815 - 826, 1983.

Cohen 和 Grossberg 用 LaSalle 的不变性定理分析竞争型神经网络的稳定性。作者对网络的描述是非常一般化的，并展示如何将他们的分析运用到很多不同类型的递归神经网络上。

[Gros67] S. Grossberg, "Nonlinear difference - differential equations in prediction and learning theory," *Proceedings of the National Academy of Sciences*, vol. 58, pp. 1329 - 1334, 1967.

18-37 这是 Grossberg 的一项早期工作，讨论了在动态稳定配置中信息的存储。

[Hopf82] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational properties," *Proceedings of the National Academy of Sciences*, vol. 79, pp. 2554 - 2558, 1982.

这是最初提出 Hopfield 神经网络的论文，它标志着神经网络领域研究工作的重新

兴起。文章描述一种具有按内容寻址存储器性能的断续型网络。Hopfield 阐明网络涉及对特定 Lyapunov 函数的最小化。

[Hopf84] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two - state neurons," *Proceedings of the National Academy of Sciences*, vol. 81, pp. 3088 - 3092, 1984.

Hopfield 展示了一个模拟电路可以作为一个具有分级响应的大型神经网络的一个功能模型。推导了这个网络的 Lyapunov 函数并用于设计按内容寻址联想存储器的网络。

[HoTa85] J. J. Hopfield and D. W. Tank, " 'Neural' computation of decisions in optimization problems ," *Biological Cybernetics*, vol. 52, pp. 141 - 154, 1985.

这篇文章描述用 Hopfield 网络解决最优化问题。货郎担问题(货郎旅行于若干城市之间而每个城市只去一次的总路程达到最短)被映射到 Hopfield 网络。

[Koho72] T. Kohonen, "Correlation matrix memories, " *IEEE Transactions on Computers*, vol. 21 , pp. 353 - 359, 1972.

Kohonen 提出了一种联想存储器的关联矩阵模型。该模型使用外积规则(和 Hebb 规则同样有名的一个规则)进行训练，学习输入/输出向量的关联。他主要强调网络的数学结构。Anderson 也同时独立发表了类似的论文[Ande72]。 18-38

[LiMi89] J. Li, A. N. Michel and W. Porod, "Analysis and synthesis of a class of neural networks: Linear systems operating on a closed hypercube," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 11 , pp. 1405 - 1422, November 1989.

这篇文章描述可在一个闭合的超立方体(类似 Hopfield 网络)中由一阶线性微分方程定义的神经网络。预期的和无用的平衡点落在超立方体的角上。作者们讨论了使伪平衡点数目达到最小的设计过程。

[McPi43] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*., vol. 5, pp. 115 - 133, 1943.

这篇文章引入了神经元第一个数学模型。在这个模型中，将输入信号的加权和与某个阈值比较，从而确定神经元是否激发。

[TaHo86] D. W. Tank and J. J. Hopfiled, "Simple 'neural' optimization networks: An A/D converter, signal decision circuit and a linear programming circuit," *IEEE Transactions on Circuits and Systems*, vol. 33, no. 5, pp. 533 - 541, 1986.

作者描述了如何设计 Hopfield 网络使其可以解决特定的最优化问题。其中可以看到一个将 Hopfield 网络应用到模/数转换的例子。 18-39

## 习题

**E18.1** 在 18.2.2 节我们举的例子中，增益系数 $\gamma = 1.4$。图 18-3 显示了那个例子的 Lyapunov 函数。高增益的 Lyapunov 函数见图 18-9。

(i) 证明这个例子中 Lyapunov 函数的极小值位于满足 $n_1 = n_2 = f(n_1) = f(n_2)$ 的那些点。(用式(18.42)并把 $V(\mathbf{a})$的梯度设为 0。)

(ii) 研究从 $\gamma = 0.1$ 到 $\gamma = 10$ 时极小值位置的变化。

(iii) 对这个区间内的几个不同增益系数画出轮廓图。你可能需要使用 MATLAB。

**E18.2** 在例题 P18.3 中，我们使用有监督的 Hebb 规则设计 Hopfield 网络来识别下面的模式：

$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

如果我们使用另外一个设计规则[LiMing]，将得到下面的权值矩阵和偏置值：

$$\mathbf{W} = \begin{bmatrix} 1 & 0 \\ 0 & -10 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ 11 \end{bmatrix}$$

(i) 假设使用这个权值矩阵和偏置值，画出高增益的 Lyapunov 函数轮廓线图。

(ii) 讨论这个 Hopfield 网络与例题 P18.3 中设计的网络之间的性能差别。

(iii) 写一个 MATLAB M-文件来刺激 Hopfield 网络。使用 **ode45** 例行程序。画出这个网络对于几个不同初始条件的响应图。

**E18.3** 一个 Hopfield 网络具有如下高增益 Lyapunov 函数：

$$V(\mathbf{a}) = -\frac{1}{2}((a_1)^2 + 2a_1a_2 + 4(a_2)^2 + 6a_1 + 10a_2)$$

(i) 求权值矩阵与偏置值向量。

18-40 (ii) 求 $V(\mathbf{a})$的梯度与赫森矩阵。

(iii) 画出 $V(\mathbf{a})$的轮廓线图。

(iv) 求 $V(\mathbf{a})$的静止点。使用 LaSalle 不变性定理的推论获取任一稳定点吸引区的尽可能多的信息。

**E18.4** 在例题 P18.2 中我们展示了 Hopfield 网络可用来作 A/D 转换器。

(i) 设输入值 $y = 0.5$，画出 2 位 A/D 转换器网络的高增益 Lyapunov 函数的轮廓线图。

(ii) 设 $y = 2.5$，重复第(i)小题。

(iii) 用(i)和(ii)小题的结果来解释网络是如何运算的。网络可以正确地进行 A/D 转换吗?

**E18.5** 假设二进制的原型向量为

$$\mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ 1 \\ -1 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$$

(i) 设计一个连续的 Hopfield 网络(只指定连接权值与偏置值)来识别这些模式。使用 Hebb 规则。

(ii) 求高增益 Lyapunov 函数的赫森矩阵。其特征值和特征向量是什么?

(iii) 假设增益系数很大，网络的稳定平衡点是什么?

**E18.6** 在习题 E7.7 中我们曾经问过这样一个问题：一个权值矩阵可以存储多少原型模式? 在 Hopfield 网络中有同样的问题。开始时有两个数字“0”和“1”。(数字如图 18-17 所示。)每次增加一个数字直到“6”。在每次随机改变 2 个、4 个和 6 个象素后，测试一下网络对重构数字的正确识别率。

(i) 首先使用 Hebb 规则为数字“0”和“1”建立一个权值矩阵。然后每个数字随

机更改两个象素点，并加入噪声数字。重复这个过程 10 次并记录正确模式(不含噪声)的出现率。每个数字改变 4 个象素和 6 个象素后重复上述过程。然后使用数字“0”，“1”和“2”重复整个过程。每次一个数字，继续下去直至数字“0”到“6”都被使用过。当你完成了整个测试后，画出错误次数对存储数字数目百分比的三条曲线，对于 2 个、4 个和 6 个象素错误各有一条曲线。 18-41

(ii) 使用伪逆规则(见第 7 章)重复( i )小题，并比较两种规则的结果。

(iii) 为了佐证使用[LiMi89]中描述的方法，重复(i)小题。在那篇论文中，它被称为合成过程 5.1。

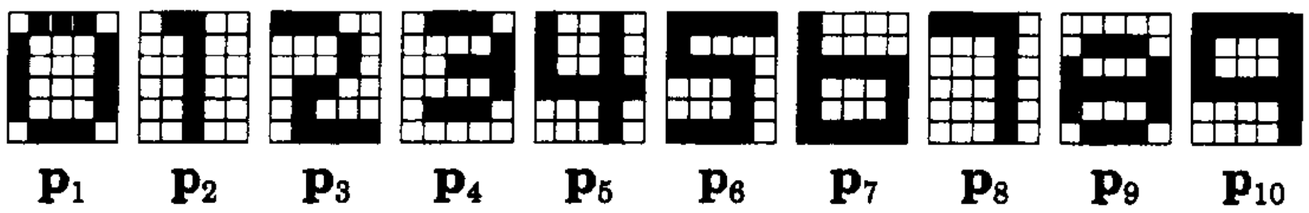



图 18-17

18-42

# 第19章 结 束 语

## 19.1 目的

我们已经讨论了多种重要的神经网络结构和学习规则，也解释了如何将它们应用于模式识别，函数逼近，自适应滤波等应用方面。当然，在一本书中不可能对每一种重要的神经网络都进行很深地讨论。神经网络是一个相当广阔而又发展很快的领域。

本章将告诉你下一步还需研究一些什么。我们将讨论本书中没有具体介绍的一些网络，同时也为你进一步阅读提供了一些参考文献。

19-1

## 19.2 理论和实例

第 3 章粗略地介绍了本书中所讲述的主要网络。回忆一下，第 3 章提供了三种神经网络并把它们应用到一个简单的模式识别问题上。这三种网络分别是感知机、Hamming 网络和 Hopfield 网络。感知机是一种前馈网络，后来我们又把它推广到多层感知机网络。在第 4 ~ 7 章及第 10 ~ 12 章中我们讨论了前馈网络(感知机，线性联想器，自适应线性神经元，多层感知机)。Hamming 网络是一种竞争网络。在第 14 ~ 16 章我们提供了几种竞争网络(Kohonen 层，自组织特征图，学习向量量化器，Grossberg 网络，ART 网络)。Hopfield 网络是动态联想存储器网络的一个例子。第 18 章讲述了连续型的 Hopfield 网络。

本章中我们讨论一下前几章没有详细讲述的一些其他神经网络。这些网络和我们所讲过的网络有关，它们也归入第 3 章提出的三类网络—前馈网络、竞争网络和动态联想存储器网络。我们将提供一些属于这三种范畴的其他网络。

除了讲述当前神经网络的研究，我们还将探讨神经网络的经典基础。在前几章，我们描述了在线性代数、最优化和稳定性理论中对神经网络有贡献的一些原理。本章我们要指出对这一领域提供了概念和算法的某些其他学科。

本章最后一节列出了当前一些神经网络杂志和书籍，以供进一步深入学习使用。

本章其余部分讨论的网络深奥难懂，而且即使它们完成了，很快也会过时。然而我们希望这些网络能使你对这个领域的广度有所了解，并为你继续深入研究网络提供一个起点。

### 19.2.1 前馈和联想网络

1. 径向基本网络

首先在求解多实变量插值问题时引入径向基本网络。径向基本函数(RBF)网络由两层组成。这是典型的多层网络，第一层的神经元不使用输入值的加权和及 S 型传输函数。相反，第一层神经元的输出(每一个表示基本函数)由网络的输入与基函数“中心”的距离决定。随着输入离中心的距离的增大，神经元的输出迅速减至零。RBF 网络的第二层是线性的，它产生第一层输出的加权和。RBF 网络具有局部化接收域，因为神经元只对接近中心的输入有响应。这同标准多层网络形成对照。标准多层网络使用 S 型函数建立全局响应。RBF 比多

19-2

层感知机的训练要快，但是对高维输入空间来说，需要很多神经元[Power87]、[BrLo88]、[MoDa89]、[PoGi90]。

2.CMAC（小脑模型连接控制器）

与RBF一样，CMAC网络使用具有局部化接收域的隐含单元。这使学习更有效。CMAC由Albus(1971)从小脑模型发展来的。他把网络应用于机械手的控制。CMAC的原始过程是由表查找过程实现的[Albu71]、[Albu75]。

3. 多项式网络

在第3、4章和第10章我们讨论了单层网络的局限。它们只能用来识别线性可分的模式。在第11章我们可用多层网络来克服这种局限。多层网络可实现任一形状区域的识别。这个问题的另一种解决方案是只用一层但采用不同的神经元。神经元不仅可计算输入值的线性组合，并可计算更复杂的函数如多项式。下面就是多项式网络的一个例子。

**函数链接网络** 函数链接网络的神经元接收标准输入元素的线性组合加上一个高次项。高次项包括不同输入元素乘积的各种组合。

**数据处理的成组方法(GMDH)** 数据处理的成组方法(GMDH)由A. G. Ivakhnenko于1968提出。网络中每个神经元只有两个输入。每个神经元的输出是两个输入的二次多项组合。网络的每一层增加网络创建的多项式次数[Ivak71]。 19-3

**Sigma-Pi网络** 这个网络是多层感知机网络的推广。它将乘积项结合到每一个神经元的净输入上。每一个净输入是加在那个神经元上所有信号的加权和，以及选定的这些信号积的加权和[RuMc86]、[HeNo95]。

4. 模块化网络

这个网络是具有局部接收域网络(如RBF与CMAC)和全局网络(如多层感知机)之间的一种折衷。它由一系列专家网络构成，其中每一个都可以是多层网络，再加一个门控网络，后者把专家网络的输出汇总成整体输出[JaJo91a]、[JaJo91b]。

5. 自适应评价网络

基本的自适应评价系统通常用于控制系统。它由两个网络构成：评价网络和行动网络。评价网络的目的是用来估计缺乏真实错误测量情况下系统的性能。行动网络用来更新来自评价网络的信息。系统使用介于有监督和无监督之间的一种强化学习规则来训练。系统虽不使用目标输出，但是接收增强信号，如“成功”或“失败”[BaSu83]、[Sutt84]。

6. 反传网络的变形

除第12章谈到的外，反传网络还有许多变形。这可能是自1986年以来神经网络研究最活跃的一个领域。下面我们讨论一些较成功的反传网络的变形。

**Quickprop过程** Quickprop是反传的一种启发式修改。通过假设错误曲面是二次的和一个权值的导数独立于其他权值的导数决定步长。

**Rprop过程** 当净输入的大小太大时，S型函数的导数太小了。设计Rprop过程就是为了克服这个问题。这可能引起性能指数梯度变小，即使离极小值点很远。最速下降法产生很小的步长。在Rprop中，步长不是梯度大小的函数。如果一个给定权的导数符号在几次迭代中都不变，则步长增大。如果导数符号不断地摆动，则步长减小。 19-4

**级联相关** 级联相关学习结构(Fahlman和Lebiere，1990)是网络增长过程的一个例子。开始时没有隐藏结点，并可用LMS算法训练。网络一次增加一个隐藏结点。每个隐藏结点

都与输入结点和先前的隐藏结点有连接。每个隐藏结点都连接到每个输出结点[FaLe90]。

**网络修剪** 如第 11 章所述，神经网络训练的一个问题是缺乏一般化。如果网络有过多的参数，它可能会对数据适合过度了。当数据在训练集合中时，错误可能变得很小；反之，当数据在训练集合之外时，则错误很大。网络一般化的方法之一是减少参数。网络修剪是在网络训练后，去掉一些权值。例子有最佳脑损坏[LeDe90]和最佳脑手术[HaSt93]。

**规则化** 另一种解决网络过度适合的方法是对性能指数增加一项，加大复杂性。换句话说，修改后的性能指数包含两部分，一部分是误差平方的函数，而另一部分是网络参数数目(或它们的大小)的函数。训练过程试图用最不复杂的网络来使误差平方达到最小值。有两个规则化的例子：权值减小过程[Hinto89]和权值消除方法[WeRu91]。

**停止训练** 这个过程和规则化一样是用来使训练网络更一般化。其思想是把数据分为三个部分：训练集、确认集和测试集。训练集用来计算梯度和决定权值更新。确认集用来判断什么时候应停止训练。测试集是用来比较不同网络的性能。当确认集中误差开始增加时，训练即停止。这使网络不会在训练集中过度适合[Sarl95]。

19-5

7. 概率神经网络

概率神经网络(PNN)是一种标准的贝叶斯分类器的一种并行实现。它是一个可用来把模式分类的三层网络。概率神经网络的标准形式是不进行训练的。与 Hamming 网络的方式相似，训练向量仅变为第一层的权值向量。这种网络的优势在于不用训练。但也有不利之处，当训练集中的向量很多时，权值矩阵可能非常大。如果训练集太大，就要进行一个聚类运算来减少大小[Spec90]。

8. 广义回归神经网络

与 PNN 一样，广义回归神经网络(GRNN)也不需要重复训练过程。PNN 用于分类问题，而 GRNN 则用于连续变量的估计，就如标准回归技术一样。它与径向基本函数网络和 CMAC 有关系。它建立在称为核心回归的标准统计技术上的。

9. 具有时间延迟的多层网络

多层前馈网络可以逼近任何 Borel 可积函数，但不能加入时间独立性。为此，一些研究人员提出了把多层感知机和时间延迟结合起来的网络，其中有些包括反馈连接。

**时间延时神经网络** 时间延时神经网络(TDNN)是一个多层前馈网络。每一层的输出分几步存入缓冲区，然后再整个连入下一层。它主要应用于语音识别[LaHi88]、[WaHa89]。

**有限刺激响应多层感知机** 有限刺激响应(FIR)多层感知机是 TDNN 的推广。FIR 网络是一个多层网络，每一个权值被一个有限刺激响应滤波器代替。这个网络首先被应用于时间序列的预测[Wan90a]、[Wan90b]、[Wan94]。

**管道式递归神经网络** 管道式递归神经网络(PPRN)由一组模块构成。每一个模块接收适当延迟的输入信号。每一个模块是一个全连结的递归神经网络，具有一个单输出神经元。这些模块按顺序运算，一个模块的输出馈给后面的模块。PPRN 比 TDNN 和 FIR 网络更复杂。因为它同时有前馈和反馈(递归)连接，因而有无限大的存储空间。然而，网络的模块化能使训练更有效。PPRN 用于非静态信号的自适应预测[HaLi95]。

19-6

**非线性自回归移动平均网络** 非线性自回归移动平均(NARMA)网络是建立在使用时序分析和系统识别的 ARMA 模型基础上的。它包含带两输入集合的多层网络。第一个集合包括输入信号和输入信号的延迟值。第二个集合包括网络输出的延迟值。这个系统用于动态系

统的识别和控制以及时间序列的预测[NaPa90]。

**Elman网络** Elman网络是一个两层的网络，具有从隐藏层输出到它的输入的反馈连接。反馈路径使Elman网络能学习识别和产生瞬时模式和空间模式[Elma90]。

**实时递归网络** 实时递归网络(RTRN)的结构与离散Hopfield网络相似，只是它含有隐藏神经元。RTRN有两层：隐藏层和输出层。每层都接收两组输入。第一组是所有神经元(包含隐含层和输出层神经元)输出的延迟值。第二组是外部输入信号。RTRN加上相应的学习规则可以连续地运行和进行实时学习。不过这也有不利之处，因为它是全连接的，所以需要很多神经元和过多的计算[WiZi89]。

10. 带延迟的多层网络训练

前一小节所述的多层网络及其他动态网络，由于时间依赖性而不能用标准反传算法进行正常训练。它们需要用动态反传算法。动态反传有两种基本结构。一种是沿时间前进，另一种则沿时间后退。

**沿时间反传** 动态网络的沿时间反传(BTT)算法是静态网络反传算法的扩展。它是通过时间方向前展开网络而导出的多层反馈网络，每一个时间步产生一层。反传过程能有效地沿时间后移。BTT算法的特征是较低的计算代价和较高的存储需求。标准BTT算法不适合实时运算。因为在梯度计算出来前(通过整个时间序列的反传)，每一个时间步的网络输出都必须计算出来。(BTT概念的例子请见习题E11.5。)[RuMc86]、[Werb90]。 19-7

**前向扰动算法** 前向扰动算法(也称为实时递归学习算法、灵敏方法或循环反传算法)是用于实时运算的。

梯度每一个向前时间步更新一次。算法的特点是较高的计算代价和较低的存储需求。(前向扰动概念的例子见例题P11.4和P11.9。)[WiZi89]、[NaPa91]。

### 19.2.2 竞争网络

1. 对传网络

对传网络(CPN)把instar竞争层与outstar层结合起来。CPN可用于数据压缩、函数逼近或模式联想。它把有监督和无监督的训练结合起来[Hech87]、[Hech88]。

2. 新认知机

新认知机是一种层次结构的网络，也是目前最复杂的网络之一。网络每一层的神经元仅接收来自前一层神经元的局部子集的连接。新认知机用于模式识别，尤其是手写字符的识别。它对模式的大小和形变不敏感[FuMi83]、[Fuku88]。

3. ART网络

除了第16章讨论的ART1外，ART网络还有许多变形。ART1用于二进制模式的无监督的分类。后来的网络被修改用于模拟模式的识别，有些也包含有监督的学习[CaGr87]、[CaGr90]、[CaGrMa92]、[CaGrRo91]、[CaGrRe91]、[CaRo95]。 19-8

### 19.2.3 动态联想存储器网络

Hopfield网络是本书中惟一的动态联想存储器网络。下面介绍文献中提出的一些相关的网络。

1. Li - Michel 网络

这类网络可描述为一个由定义在闭超立方体上的一阶线性微分方程组的系统。网络的设计过程保证了假平衡点尽可能地少和原型模式的吸引区尽可能地大。这些网络与 Hopfield 模型密切相关，设计过程能直接应用于 Hopfield 模型[LiMi89]、[MIFa90]。

2. Boltzman 机

Hopfield 网络将会收敛于 Lyapunov 函数的局部极小值，但并不能保证它会收敛于全局极小值。在 Boltzman 机中，为达到全局极小值而使用噪声。这个技术被称为模拟退火，同冶金中的退火相似。模拟退火是指一个金属体被加热到接近融化，然后按照指定的时刻表慢慢冷却。高温引起了温度搅动，这使金属不能在较高的能量状态凝固。在 Boltzman 机中，网络的轨迹被加进噪声，这样就不会陷入局部极小值。噪声的大小随时间逐渐减小，因此网络最终可以收敛[GeGe84]、[AkHi85]。

3. 双向联想存储器

双向联想存储器(BAM)和 Hopfield 网络有关，它的结构与 ART 结构有点相似。BAM 由两层组成，并使用两层之间的向前和向后信息流，执行对存储的刺激 - 响应联想信息的搜索。网络演化到能量曲面的一个局部极小值，这是两个模式共振的状态，在每一层的输出有一个模式[Kosk87]、[Kosk88]。

4. 盒中脑状态模型

盒中脑状态(BSB)是先于 Hopfield 模型的动态联想存储器模型。这种离散模型是线性联想器的扩展。为了使网络响应在超立方体内，增加了反馈和使用饱合线性传输函数。对高增益的 Hopfield 网络来说，稳定点对应于超立方体的角[AnSi77]。

19-9

### 19.2.4 神经网络的经典基础

神经网络的许多技术与其他研究领域提出的过程密切相关。这一点常常被刚从事这个领域研究的人员忽视。在这一小节，我们想回顾一下与神经网络结构或学习规则密切相关的其他学科的思想。

1. 统计学

很多种神经网络在功能上与数理统计的一些标准过程等价。例如，单层前馈网络(包括函数链接神经网络和多项式神经网络)基本上是推广的线性模型。两层的前馈网络与投影寻踪回归密切相关。概率神经网络与核判别分析相同。一般回归神经网络与 Nadaraya - Watson 核回归相同。Kohonen 竞争性网络与 k - 均值聚类分析相似。Hebb 学习与主成分分析密切相关[Smit93]、[Sarle94]、[BaCo94]、[Brid90]、[MacK92]、[Joll86]、[HwLa94]。

2. 物理学/统计力学

一些神经网络的思想来自物理学，尤其是统计力学。例如，Hopfield 模型就是模仿统计力学中磁性材料的伊辛自旋模型。Boltzman 机建立在模拟退火原理的基础上，而这个原理也是来自统计物理学文献[ShKi72]、[KiSh78]、[Pere84]、[Pere92]。

3. 生物学/心理学

神经网络与生物学和心理学中的思想之间的联系是显然的。但是，即使整个神经网络领域都受到这两门学科的影响，我们也时常跟不上这些学科的发展[Thom75]、[Gros82]、[ChSe92]、[Ande95]。

### 19.2.5 参考书目和杂志

1. 神经网络杂志

本章所提供的一些参考文献可以说只是神经网络研究与应用的冰山一角。如果需要了解当前神经网络研究的一些热门，可查阅下面一些杂志。其中一些是专门研究神经网络，而另一些则覆盖更广阔的领域，但对神经网络研究非常重视。 19-10

- 《应用光学》(*Applied Optics*)
- 《生物学控制论》(*Biological Cybernetics*)
- 《认识科学》(*Cognitive Science*)
- 《联系科学》(*Connection Science*)
- 《IEEE 电路与系统学报》(*IEEE Transactions on Circuits and Systems*)
- 《IEEE 神经网络学报》(*IEEE Transactions on Neural Networks*)
- 《IEEE 系统、人类与控制论学报》(*IEEE Transactions on Systems, Man, and Cybernetics*)
- 《神经系统国际杂志》(*International Journal of Neural Systems*)
- 《人工神经网络杂志》(*Journal of Artificial Neural Networks*)
- 《认知神经科学杂志》(*Journal of cognitive Neurosciences*)
- 《神经科学杂志》(*Journal of Neurosciences*)
- 《机器学习》(*Machine Learning*)
- 《网络：神经系统计算》(*Networks: Computation in Neural Systems*)
- 《神经计算》(*Neural Computation*)
- 《神经网络》(*Neural Networks*)
- 《美国科学院进展》(*Proceedings of the National Academy of Sciences*)

2. 神经网络教科书

我们在下面列出了一些神经网络参考书目。虽然我们希望你能对本书满意，但是要想深入了解一个主题，最好能从不同的角度考察。下面每一本书都有一些自己的特色。

- 《*Self-Organization and Associative Memory*》, 3rd Edition, T. Kohonen, Springer-Verlag, 1989.
- 《*Adaptive Pattern Recognition and Neural Networks*》, Y. -H. Pao, Addison-Wesley, 1989.
- 《*Neurocomputing*》, R. Hecht-Nielsen, Addison-Wesley, 1990.
- 《*Introduction to the Theory of Neural Computation*》, J. Hertz, A. Krogh and R. G. Palmer , Addison-Wesley, 1991. 19-11
- 《*Neural Networks: Algorithms, Applications, and Programming Techniques*》, J. A. Freeman and D. M. Skapura, Addison-Wesley, 1991.
- 《*Neural Computing: An Introduction*》, 2nd Edition, R. Beale and T. Jackson, Adam Hilger, 1991.
- 《*Introduction to Artificial Neural Systems*》, J. Zurada, West Publishing, 1992.
- 《*An Introduction to the Modeling of Neural Networks*》, P. Peretto, Cambridge Uni-

versity Press, 1992.

- 《*Neural Networks and Fuzzy Systems*》, B. Kosko, Prentice-Hall, 1992.
- 《*Neural Networks for Pattern Recognition*》, A. Nigrin, MIT Press, 1993.
- 《*Digital Neural Networks*》, S. Y. Kung, Prentice-Hall, 1993.
- 《*Neural Networks for Statistical Modeling*》, M. Smith, Van Nostrand Reinhold, 1993.
- 《*Advanced Methods in Neural Computing*》, P. D. Wasserman, Van Nostrand Reinhold, 1993.
- 《*Neural Networks: A Tutorial*》, M. Chester, Prentice-Hall, 1993.
- 《*Neural Networks for Optimization and Signal Processing*》, A. Cichocki and R. Unbehauen, John Wiley & Sons, 1993.
- 《*Neural Networks: A Comprehensive Foundation*》, S. Haykin, Macmillan, 1994.
- 《*Neural Network Principles*》, R. L. Harvey, Prentice-Hall, 1994.
- 《*Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*》, L. Fausett, Prentice-Hall, 1994.
- 《*Fundamentals of Artificial Neural Networks*》, M. H. Hassoun, MIT Press, 1995.
- 《*An Introduction to Neural Networks*》, J.A. Anderson, MIT Press, 1995.

19-12

- 《*Self-Organizing Maps*》, T. Kohonen, Springer-Verlag, 1995.

## 19.3 结束语

我们希望本书有助于传播神经网络研究领域的一些观点，鼓舞你继续探索下去。这个领域博大精深，并且发展迅速。在今后几年里，神经网络肯定会有许多新的发展。本书中所讨论的一些概念已为你继续探索打下了一个坚实的基础。在这一章里，我们为你继续研究神经网络提供了一些方向。

19-13

## 参考文献

1. 径向基本网络

[BrLo88] D. S. Broomhead and D. Lowe, "Multivariavble functional interpolation and adaptive networks," *Complex Systems*, vol. 2, pp. 321 - 355, 1988.

[MoDa89] J. E. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, pp. 281 - 294, 1989.

[PoGi90] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proceedings of the IEEE*, vol. 78, pp. 1481 - 1497, 1990.

[Powe87] M. J. D. Powell, "Radial basis functions for multivariable interpolation: A review," in *Algorithms for the Approximation of Functions and Data*, J. C. Mason and M. G. Cox, eds., Oxford, England: Clarendon Press, pp. 143 - 167, 1987.

2. CMAC (小脑模型连接控制器)

[Albu71] J. S. Albus, "A theory of cerebellar function," *Mathematical Biosciences*, vol. 10, pp. 25 - 61, 1971.

[Albu75] J. S. Albus, "A new approach to manipulator control: The cerebellar model articula-

tion controller (CMAC)," *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, vol. 97, pp. 220-227, 1975.

3. 多项式网络

**函数链接网络**

[Pao 89] Y.-H. Pao, *Adaptive Pattern Recognition and Neural Networks*, Reading, MA: Addison-Wesley, 1989.

**数据处理的成组方法(GMDH)**

[Ivak71] A. G. Ivakhnenko, "Polynomial theory of complex systems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 12, pp. 364-378, 1971.

**Sigma-Pi 网络**

[HeNo95] M. Heywood and P. Noakes, "A framework for improved training of sigma-pi networks," *IEEE Transactions on Neural Networks*, vol. 6, no. 4, pp. 893-903, 1995. 19-14

[RuMc86] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing, Explorations in the Macrostructure of Cognition*, vol. 1, Cambridge, MA: MIT Press, 1986.

4. 模块化网络

[JaJo91a] R. A. Jacobs and M. I. Jordan, "A competitive modular connectionist architecture," in *Advances in Neural Information Processing Systems* 3, R. P. Lippman, J. E. Moody and D. J. Touretzky, eds., pp. 767-773, San Mateo, CA: Morgan Kaufmann, 1991.

[JaJo91b] R. A. Jacobs, M. I. Jordan, S. J. Nowlan and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, pp. 79-87, 1991.

5. 自适应评价网络

[BaSu83] A. R. Barto, R. S. Sutton and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 13, pp. 834-846, 1983.

[Sutt84] R. S. Sutton, "Temporal credit assignment in reinforcement learning," Ph. D, Dissertation, University of Massachusetts, Amherst, MA, 1984.

6. 反传学习的变形

**Quickprop 过程**

[Fahl89] S. E. Fahlman, "Fast learning variations on back-propagation: An empirical study," in *Proceedings of the* 1988 *Connectionist Models Summer School* (Pittsburgh 1988), D. Touretzky, G. Hinton and T. Sejnowski, eds., pp. 38-51, San Mateo, CA: Morgan Kaufmann, 1989.

**Rprop 过程**

[RiBr93] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm", *Proceedings of the IEEEInternational Conference on Neural Networks*, San Francisco: IEEE, 1993.

**级联相关**

[FaLe90] A. E. Fahlman and C. Lebiere,: "The cascade-correlation learning architecture," in *Advances in Neural Information Processing Systems* 2, D. Touretzky, ed., San Mateo,

19-15 CA: Morgan Kaufmann, pp. 524 - 532, 1990.

**网络修剪**

[HaSt93] B. Hassibi, D. G. Stork and G. J. Wolff, "Optimal brain surgeon and general network pruning," *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 441 - 444, 1992.

[LeDe90] Y. Le Cun, J. S. Denker and S. A. Solla, "Optimal Brain Damage," in *Advances in Neural Information Processing Systems* 2, D. Touretzky, ed., San Mateo, CA: Morgan Kaufmann, pp. 598 - 604, 1990.

**规则化**

(i)权值衰减

[Hinto89] G. E. Hinton, "Connectionist learning procedures," *Artificial Intelligence*, vol. 40, pp. 185 - 234, 1989.

(ii)权值消除

[WeRu91] A. S. Weigand, D. E. Rumelhart and B. A. Huberman, "Generalization by weight elimination with application to forecasting," *in Advances in Neural Information Processing Systems* 3, R. Lippman, J. Moody and D. Touretzky, eds., San Mateo, CA: Morgan Kaufmann, pp. 575 - 582, 1991.

**停止的训练**

[Sarle95] W. S. Sarle, "Stopped training and other remedies for overfitting," to appear in *Proceedings of the 27th Symposium on the Interface*, 1995.

7. 概率神经网络(PNN)

[Spec90] D. F. Specht, "Probabilistic neural networks," *Neural Networks*, vol. 3, no. 1, pp. 109 - 118, 1990.

8. 广义回归神经网络(GRNN)

[Spec91] D. F. Specht, "A general regression neural network," *IEEE Transactions on Neural*
19-16 *Networks*, vol. 2, no. 6, pp. 568 - 576, 1991.

9. 带时间延迟的多层网络

**时间延时神经网络**

[LaHi88] K. J. Lang and G. E. Hinton, "The development of the time-delay neural network architecture for speech recognition," Technical Report CMU-CS-88-152, Carnegie-Mellon University, Pittsburgh, PA, 1988.

[WaHa89] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, pp. 328 - 339, 1989.

**有限刺激响应(FIR)网络**

[Wan 90a] E. A. Wan, "Temporal backpropagation for FIR neural networks," *IEEE International Joint Conference on Neural Networks*, vol. 1, San Diego, CA, pp. 575 - 580, 1990.

[Wan90b] E. A. Wan, "Temporal backpropagation: An efficient algorithm for finite impulse response neural networks," in *Proceedings of the 1990 Connectionist Models Summer*

*School*, D. S. Touretzky, J. L. Elman, T. J. Sejnowski and G. E. Hinton, eds., San Mateo, CA: Morgan Kaufmann, pp. 131-140.1990.

[Wan94] E. A. Wan, "Time series prediction by using a connectionist network with internal delay lines," in *Time Series Prediction: Forecasting the Future and Understanding the Past*, A. S. Weigend and N. A. Gershenfeld, eds., Reading, MA: Addison Wesley, pp. 195-217, 1994.

**管道式递归神经网络(DPRN)**

[HaLi95] S. Haykin and L. Li, "Nonlinear adaptive prediction of nonstationary signals," *IEEE Transactions on Signal processing*, vol. 43, no. 2, pp. 526-535, 1995.

**非线性自回归移动平均(NARMA)网络**

[NaPa90] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4-27, 1990.

**Elman 网络**

[Elma90] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, pp. 179-211, 1990.

19-17

**实时递归网络(RTRN)**

[WiZi89] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, pp. 270-280, 1989.

10. 带时间延迟的多层网络训练

**沿时间反传(BTT)**

[RuMc86] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing, Explorations in the Macrostructure of Cognition*, vol. 1, Cambridge, MA: MIT Press, 1986.

[Werb90] P. J. Werbos, "Backpropagation through time: What it is and how to do it," *Proceedings of the IEEE*, vol. 78, pp. 1550-1560, October 1990.

**前向扰动算法**

[NaPa91] K. S. Narendra and K. Parthasarathy, "Gradient methods for the optimization of dynamical systems containing neural networks," *IEEE Transactions on Neural Networks*, vol. 2, pp. 252-262, 1991.

[WiZi89] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, pp. 270-280, 1989.

**竞争性网络**

1. 对传网络

[Hech87] R. Hecht-Nielsen, "Counterpropagation Networks," *Applied Optics*, vol. 26, pp. 4979-4984, December 1987.

[Hech88] R. Hecht-Nielsen, "Applications of Counterpropagation Networks," *Neural Networks*, vol. 1, no. 2, pp. 131-139, 1988.

2. 新认知机

[FuMi83] K. Fukushmia, S. Miyake and T. Ito, "Neocognitron: A neural network model for a mechanism of visual pattern recognition ," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol.13, no. 5, pp. 826 - 834, 1983.

[Fuku88] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural Networks*, vol. 1, pp. 119 - 130, 1988.

19-18

3. ART 网络

[CaGr87b] G. A. Carpenter and S. Grossberg, "ART2: Self-organization of stable category recognition codes for analog input patterns," *Applied Optics*, vol. 26, no. 23, pp. 4919 - 4930, 1987.

[CaGr90] G. A. Carpenter and S. Grossberg, "ART3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architecfures," *Neural Networks*, vol. 3, no. 23, pp. 129 - 152, 1990.

[CaGrMa92] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds and D. B. Rosen, " Fuzzy ARTMAP: A neural network architecture for incremental learning of analog multidimensional maps," *IEEE Transactions of Neural Networks*, vol. 3, pp. 698 - 713, 1992.

[CaGrRo91] G. A. Carpenter, S. Grossberg and D. B. Rosen, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, vol. 4, pp. 759 - 771, 1991.

[CaGrRe91] G. A. Carpenter, S. Grossberg and J. Reynolds, "ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural networks" *Neural Networks*, vol. 4, pp. 565 - 588, 1991.

[CaRo95] G. A. Carpenter and W. D. Ross, "ART-EMAP": Aneural network architecture for object recognition by evidence accumulation, "*IEEE Transactions on Neural Networks*, vol. 6, no. 4, pp. 805 - 818, July 1995.

## 联想存储器递归网络

1. Li-Michel 网络

[LiMi89] J. Li, A. N. Michel and W. Porod, "Analysis and synthesis of a class of neural networks: Linear systems operating on a closed hypercube," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 11, pp. 1405 - 1422, November 1989.

[MiFa90] A. N. Michel and J. A. Farrell, "Associative memories via artificial neural networks," *IEEE Control Systems Magazine*, April, pp. 6 - 17, 1990.

19-19

2. Boltzman 机

[AcHi85] D. H. Ackley, G. F. Hinton and T. J. Sejnowski, "A learning algorithm for Boltzman machines, "*Cognitive Science* , vol. 9, pp. 147 - 169, 1985.

[GeGe84] S. Geman and D. Geman, "Stochastic relaxation , Gibbs distributions , and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine In-*

*telligence*, vol. 6, pp. 721 - 741, 1984.

3. 双向联想存储器(BAM)

[Kosk87] B. Kosko, "Adaptive bidirectional associative memories," *Applied Optics*, vol. 26, pp. 4910 - 4918, 1987.

[Kosk88] B. Kosko, "Bidirectional associative memories," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 18, no. 1, pp. 49 - 60, 1988.

4. 盒中脑状态(BSB)

[AnSi77] J. A. Anderson, J. W. Silverstein, S. A. Ritz and R. S. Jones, "Distinctve features, categorical perception, and probability learning: some applications of a neural model," *Psychological Review*, vol. 84, pp. 313 - 351, 1977.

## 神经网络的经典基础

1. 统计学

[BaCo94] P. V. Balakrishnan, M. C. Cooper, V. S. Jacob, and P. A. Lewis, "A study of the classification capabilities of neural networks using unsupervised learning: A comparison with k-means clustering," *Psychometrika*, vol. 59, pp. 509 - 525, 1994.

[Brid90] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neurocomputing: Algorithms, Architectures and Applications*, F. Fogleman-Soulie and J. Herault, eds., Berlin: Springer-Verlag, pp. 227 - 236, 1990.

[HwLa94] J.-N. Hwang, S.-R. Lay, M. Maechler, R. D. Martin and J. Schimert, "Regression modeling in back-propagation and projection pursuit learning," *IEEE Transactions on Neural Networks*, vol. 5, no. 3, pp. 342 - 353, 1994.

[Joll86] I. T. Jolliffe, *Principal Component Analysis*, New York: Springer-Verlag, 1986. 19-20

[MacK92] D. J. C. MacKay, "A practical Bayesian framework for backpropagation networks," *Neural Computation*, vol. 4, pp. 448 - 472, 1992.

[Sarle94] W. S. Sarle, "Neural networks and statistical models," *Proceedings of the Nineteenth Annual SAS Users Group International Conference, Cary*, NC: SAS Institute, pp. 1538 - 1550, 1994.

[Smit93] M. Smith, *Neural Networks for Statistical Modeling*, New York: Van Nostrand Reinhold, 1993.

2. 物理学/统计力学

[KiSh78] S. Kirkpatrick and D. Sherrington, "Infinite-range models of spin-glasses," *Physical Review, Series B*, vol. 17, pp. 4384 - 4403, 1978.

[Pere84] P. Peretto, "Collective properties of neural networks: A statistical physics approach," *Biological Cybernetics*, vol. 50, pp. 51 - 62, 1984.

[Pere92] P. Peretto, *An Introduction to the Modeling of Neural Networks*, Cambridge, England: Cambridge University Press, 1992.

[ShKi72] D. Sherrington and S. Kirkpatrick, "Spin-glasses," *Physical Review Letters*, vol. 35, 1972.

3. 生物学/心理学

[Ande95] J. A. Anderson, *An Introduction to Neural Networks*, Cambridge, MA: MIT Press, 1995.

[ChSe92] P. S. Churchland and T. J. Sejnowski, *The Computational Brain*, Cambridge, MA: MIT Press, 1992.

[Gros82] S. Grossberg, *Studies of Mind and Brain*, Boston: D. Reidel Publishing Co., 1982.

[Thom75] R. F. Thompson, *Introduction to Physiological Psychology*, New York: Harper and Row, 1975.

19-21

# 附录A 文献目录


[AcHi85] D. H. Ackley, G. F. Hinton and T. J. Sejnowski, "A learning algorithm for Boltzman machines," *Cognitive Science*, vol. 9, pp. 147-169, 1985. (Chapter 19)

[Albe72] A. Albert, *Regression and the Moore-Penrose Pseudoinverse*, New York: Academic Press, 1972. (Chapter 7)

[Albu71] J. S. Albus, "A theory of cerebellar function," *Mathematical Biosciences*, vol. 10, pp. 25-61, 1971. (Chapter 19)

[Albu75] J. S. Albus, "A new approach to manipulator control: The cerebellar model articulation controller (CMAC)," *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, vol. 97, pp. 220-227, 1975. (Chapter 19)

[Ande72] J. A. Anderson, "A simple neural network generating an interactive memory," *Mathematical Biosciences*, vol. 14, pp. 197-220, 1972. (Chapter 1, 13, 18)

[Ande95] J. A. Anderson, *An Introduction to Neural Networks*, Cambridge, MA: MIT Press, 1995. (Chapter 19)

[AnRo88] J. A. Anderson and E. Rosenfeld, *Neurocomputing: Foundations of Research*, Cambridge, MA: MIT Press, 1989. (Chapters 1, 10)

[AnSi77] J. A. Anderson, J. W. Silverstein, S. A. Ritz and R. S. Jones, "Distinctive features, categorical perception, and probability learning: Some applications of a neural model," *Psychological Review*, vol. 84, pp. 413-451, 1977. (Chapters 18, 19)

[BaCo94] P. V. Balakrishnan, M. C. Cooper, V. S. Jacob, and P. A. Lewis, "A study of the classification capabilities of neural networks using unsupervised learning: A comparison with k-means clustering," *Psychometrika*, vol. 59, pp. 509-525, 1994. (Chapter 19)

[Barn92] E. Barnard, "Optimization for training neural nets," *IEEE Transactions on Neural Networks*, vol. 3, no. 2, pp. 232-240, 1992. (Chapter 12)

[BaSu83] A. R. Barto, R. S. Sutton and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 13, pp. 834-846, 1983. (Chapters 4, 19)

[Batt92] R. Battiti, "First and second order methods for learning: Between steepest descent and Newton's method," *Neural Computation*, vol. 4, no. 2, pp. 141-166, 1992. (Chapters 9, 12)

[Brid90] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neurocomputing: Algorithms, Architectures and Applications*, F. Fogleman-Soulie and J. Herault, eds., Berlin: Springer-Verlag, pp. 227-236, 1990. (Chapter 19)

[Brog91] W. L. Brogan, *Modern Control Theory*, 3rd Ed., Englewood Cliffs, NJ: Prentice-Hall, 1991. (Chapters 4, 5, 6, 8, 9, 17)

[BrLo88] D. S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems*, vol. 2, pp. 321-355, 1988. (Chapter 19)

[CaGr87a] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision, Graphics, and Image Processing*, vol. 37, pp. 54-115, 1987. (Chapter 16)

[CaGr87b] G. A. Carpenter and S. Grossberg, "ART2: Self-organization of stable category recognition codes for analog input patterns," *Applied Optics*, vol. 26, no. 23, pp. 4919-4930, 1987. (Chapters 16, 19)

[CaGr90] G. A. Carpenter and S. Grossberg, "ART3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures," *Neural Networks*, vol. 3, no. 23, pp. 129-152, 1990. (Chapters 16, 19)

[CaGrMa92] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds and D. B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental learning of analog multidimensional maps," *IEEE Transactions on Neural Networks*, vol. 3, pp. 698-713, 1992. (Chapters 16, 19)

[CaGrRe91] G. A. Carpenter, S. Grossberg and J. Reynolds, "ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network," *Neural Networks*, vol. 4, pp. 565-588, 1991. (Chapters 16, 19)

A-2

[CaGrRo91] G. A. Carpenter, S. Grossberg and D. B. Rosen, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, vol. 4, pp. 759-771, 1991. (Chapter 19)

[CaRo95] G. A. Carpenter and W. D. Ross, "ART-EMAP: A neural network architecture for object recognition by evidence accumulation," *IEEE Transactions on Neural Networks*, vol. 6, no. 4, pp. 805-818, July 1995. (Chapter 19)

[Char92] C. Charalambous, "Conjugate gradient algorithm for efficient training of artificial neural networks," *IEEE Proceedings*, vol. 139, no. 3, pp. 301-310, 1992. (Chapter 12)

[ChSe92] P. S. Churchland and T. J. Sejnowski, *The Computational Brain*, Cambridge, MA: MIT Press, 1992. (Chapter 19)

[CoGr83] M. A. Cohen and S. Grossberg, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 13, no. 5, pp. 815-826, 1983. (Chapters 17, 18)

[DARP88] *DARPA Neural Network Study*, Lexington, MA: MIT Lincoln Laboratory, 1988. (Chapter 1)

[Elma90] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, pp. 179-211, 1990. (Chapter 19)

[Fahl89] S. E. Fahlman, "Fast learning variations on back-propagation: An empirical study," in *Proceedings of the* 1988 *Connectionist Models Summer School*, D. Touretzky, G. Hinton and T. Sejnowski, eds., San Mateo, CA: Morgan Kaufmann, pp. 38-51, 1989. (Chapters 12-19)

[FaLe90] A. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Advances in Neural Information Processing Systems* 2, D. Touretzky, ed., San Mateo, CA: Morgan Kaufmann, pp. 524-532, 1990. (Chapter 19)

[FrSk91] J. Freeman and D. Skapura, *Neural Networks: Algorithms, Applications, and Programming Techniques*, Reading, MA: Addison-Wesley, 1991. (Chapter 14)

[FuMi83] K. Fukushima, S. Miyake and T. Ito, "Neocognitron: A neural network model for a mechanism of visual pattern recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 13, no. 5, pp. 826-834, 1983. (Chapter 19) A-3

[Fuku88] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural Networks*, vol. 1, pp. 119-130, 1988. (Chapter 19)

[GeGe84] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721-741, 1984. (Chapter 19)

[Gill81] P. E. Gill, W. Murray and M. H. Wright, *Practical Optimization*, New York: Academic Press, 1981. (Chapters 8, 9)

[GrMi89] S. Grossberg, E. Mingolla and D. Todorovic, "A neural network architecture for preattentive vision," *IEEE Transactions on Biomedical Engineering*, vol. 36, no. 1, pp. 65-84, Jan. 1989. (Chapter 15)

[Gros67] S. Grossberg, "Nonlinear difference-differential equations in prediction and learning theory," *Proceedings of the National Academy of Sciences*, vol. 58, pp. 1329-1334, 1967. (Chapter 18)

[Gros68] S. Grossberg, "Some physiological and biochemical consequences of psychological postulates," *Proceedings of the National Academy of Sciences*, vol. 60, pp. 758-765, 1968. (Chapter 13)

[Gros76] S. Grossberg, "Adaptive pattern classification and universal recoding: 1. Parallel development and coding of neural feature detectors," *Biological Cybernetics*, vol. 23, pp. 121-134, 1976. (Chapters 1, 15, 16)

[Gros80] S. Grossberg, "How does the brain build a cognitive code?" *Psychological Review*, vol. 88, pp. 375-407, 1980. (Chapter 1)

[Gros82] S. Grossberg, *Studies of Mind and Brain*, Boston: D. Reidel Publishing Co., 1982. (Chapters 13, 15, 16, 19)

[Gros90] S. Grossberg, "Neural networks: From foundations to applications," Short-Course Notes, Boston University, Boston, MA, May 6-11, 1990. (Chapter 15)

[HaMe94] M. T. Hagan and M. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 989-993, 1994. (Chapter 12)

[HaLi95] S. Haykin and L. Li, "Nonlinear adaptive prediction of nonstationary signals," *IEEE Transactions on Signal Processing*, vol. 43, no. 2, pp. 526-535. (Chapter 19) A-4

[HaSt93] B. Hassibi, D. G. Stork and G. J. Wolff, "Optimal brain surgeon and general network

pruning," *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 441-444, 1992. (Chapter 19)

[Hebb49] D. O. Hebb, *The Organization of Behavior*, New York: Wiley, 1949. (Chapters 1, 7, 13)

[Hech87] R. Hecht-Nielsen, "Counterpropagation networks," *Applied Optics*, vol. 26, pp. 4979-4984, December 1987. (Chapter 19)

[Hech88] R. Hecht-Nielsen, "Applications of counterpropagation networks," *Neural Networks*, vol. 1, no. 2, pp. 131 - 139, 1988. (Chapter 19)

[Hech90] R. Hecht-Nielsen, *Neurocomputing*, Reading, MA: Addison-Wesley, 1990. (Chapter 14)

[HeNo95] M. Heywood and P. Noakes, "A framework for improved training of sigma-pi networks," *IEEE Transactions on Neural Networks*, vol. 6, no. 4, pp. 893-903, 1995. (Chapter 19)

[Himm72] D. M. Himmelblau, *Applied Nonlinear Programming*, New York: McGraw-Hill, 1972. (Chapters 8, 9)

[Hinto89] G. E. Hinton, "Connectionist learning procedures," *Artificial Intelligence*, vol. 40, pp. 185-234, 1989. (Chapter 19)

[Hopf82] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational properties," *Proceedings of the National Academy of Sciences*, vol. 79, pp. 2554-2558, 1982. (Chapters 1, 18)

[Hopf84] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proceedings of the National Academy of Sciences*, vol. 81, pp. 3088-3092, 1984. (Chapter 18)

[HoTa85] J. J. Hopfield and D. W. Tank, " 'Neural' computation of decisions in optimization problems," *Biological Cybernetics*, vol. 52, pp. 141-154, 1985. (Chapter 18)

[HoSt89] K. M. Hornik, M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359-366, 1989. (Chapter 11)

[Hube88] D. H. Hubel, *Eye, Brain, and Vision*, New York: Scientific American Library, 1988. (Chapter 15)

A-5

[HwLa94] J.-N. Hwang, S.-R. Lay, M. Maechler, R. D. Martin and J. Schimert, "Regression modeling in back-propagation and projection pursuit learning," *IEEE Transactions on Neural Networks*, vol. 5, no. 3, pp. 342-353, 1994. (Chapter 19)

[Ivak71] A. G. Ivakhnenko, "Polynomial theory of complex systems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 12, pp. 364-378, 1971. (Chapter 19)

[Jaco88] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks*, vol. 1, no. 4, pp 295-308, 1988. (Chapter 12)

[JaJo91a] R. A. Jacobs and M. I. Jordan, "A competitive modular connectionist architecture," in *Advances in Neural Information Processing Systems* 3, R. P. Lippman, J. E. Moody and

D. J. Touretzky, eds., pp. 767-773, San Mateo, CA: Morgan Kaufmann, 1991. (Chapter 19)

[JaJo91b] R. A. Jacobs, M. I. Jordan, S. J. Nowlan and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, pp. 79-87, 1991. (Chapter 19)

[John01] G. L. Johnson, "Contributions to the comparative anatomy of the mammalian eye, chiefly based on ophthalmoscopic examination," *Philosophical Transactions of the Royal Society of London*, Series B., vol. 194, pp. 1-82, Plate 1, 1901. (Chapter 15)

[Joll86] I. T. Jolliffe, *Principal Component Analysis*, New York: Springer Verlag, 1986. (Chapter 19)

[KiSh78] S. Kirkpatrick and D. Sherrington, "Infinite-range models of spin-glasses," *Physical Review, Series B*, vol. 17, pp. 4384-4403, 1978. (Chapter 19)

[Koho72] T. Kohonen, "Correlation matrix memories," *IEEE Transactions on Computrers*, vol. 21, pp. 353-359, 1972. (Chapters 1, 13, 18)

[Koho87] T. Kohonen, *Self-Organization and Associative Memory*, 2nd Ed., Berlin: Springer-Verlag, 1987. (Chapters 13, 14)

[Kosk87] B. Kosko, "Adaptive bidirectional associative memories," *Applied Optics*, vol. 26, pp. 4910-4918, 1987. (Chapter 19)

A-6

[Kosk88] B. Kosko, "Bidirectional associative memories," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 18, no. 1, pp. 49-60, 1988. (Chapter 19)

[LaHi88] K. J. Lang and G. E. Hinton, "The development of the timedelay neural network architecture for speech recognition," Technical Report CMU-CS-88-152, Carnegie-Mellon University, Pittsburgh, PA, 1988. (Chapter 19)

[LaSa67] J. P. LaSalle, "An invariance principle in the theory of stability," in *Differential Equations and Dynamic Systems*, J. K. Hale and J. P. Lasalle, eds., New York: Academic Press, pp. 277-286, 1967. (Chapter 17)

[LeCu85] Y. Le Cun, "Une procedure d'apprentissage pour reseau a seuil assymetrique," *Cognitiva*, vol. 85, pp. 599-604, 1985. (Chapter 11)

[LeDe90] Y. Le Cun, J. S. Denker and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems* 2, D. Touretzky, ed., San Mateo, CA: Morgan Kaufmann, pp. 598-604, 1990. (Chapter 19)

[Leib90] D. Lieberman, *Learning, Behavior and Cognition*, Belmont, CA: Wadsworth, 1990. (Chapter 13)

[LiMi89] J. Li, A. N. Michel and W. Porod, "Analysis and synthesis of a class of neural networks: Linear systems operating on a closed hypercube," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 11, pp. 1405-1422, November 1989. (Chapter 18, 19)

[MacK92] D. J. C. MacKay, "A practical bayesian framework for backpropagation networks," *Neural Computation*, vol. 4, pp. 448-472, 1992. (Chapter 19)

[McPi43] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115-133, 1943. (Chapters 1, 4, 18)

[MiFa90] A. N. Michel and J. A. Farrell, "Associative memories via artificial neural networks," *IEEE Control Systems Magazine*, April, pp. 6-17, 1990. (Chapter 19)

[MiPa69] M. Minsky and S. Papert, *Perceptrons*, Cambridge, MA: MIT Press, 1969. (Chapters 1, 4)

[MoDa89] J. E. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, pp. 281-294, 1989. (Chapter 19)

[NaPa90] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4-27, 1990. (Chapter 19)

A-7

[NaPa91] K. S. Narendra and K. Parthasarathy, "Gradient methods for the optimization of dynamical systems containing neural networks," *IEEE Transactions on Neural Networks*, vol. 2, pp. 252-262, 1991. (Chapter 19)

[NgWi90] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," *Proceedings of the IJCNN*, vol. 3, pp. 21-26, July 1990. (Chapter 12)

[Pao 89] Y.-H. Pao, *Adaptive Pattern Recognition and Neural Networks*, Reading, MA: Addison-Wesley, 1989. (Chapter 19)

[Park85] D. B. Parker, "Learning-logic: Casting the cortex of the human brain in silicon," Technical Report TR-47, Center for Computational Research in Economics and Management Science, MIT, Cambridge, MA, 1985. (Chapter 11)

[Pere84] P. Peretto, "Collective properties of neural networks: A statistical physics approach," *Biological Cybernetics*, vol. 50, pp. 51-62, 1984. (Chapter 19)

[Pere92] P. Peretto, *An Introduction to the Modeling of Neural Networks*, Cambridge, England: Cambridge University Press, 1992. (Chapter 19)

[PoGi90] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proceedings of the IEEE*, vol. 78, pp. 1481-1497, 1990. (Chapter 19)

[Powe87] M. J. D. Powell, "Radial basis functions for multivariable interpolation: A review," in *Algorithms for the Approximation of Functions and Data*, J. C. Mason and M. G. Cox, eds., Oxford, England: Clarendon Press, pp. 143-167, 1987. (Chapter 19)

[RiBr93] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco: IEEE, 1993. (Chapter 19)

[RiIr90] A. K. Rigler, J. M. Irvine and T. P. Vogl, "Rescaling of variables in back propagation learning," *Neural Networks*, vol. 3, no. 5, pp 561-573, 1990. (Chapter 12)

[Rose58] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, pp. 386-408, 1958. (Chapters 1, 4)

[Rose61] F. Rosenblatt, *Principles of Neurodynamics*, Washington DC: Spartan Press, 1961. (Chapter 4)

A-8

[RuHi86] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by

back-propagating errors," *Nature*, vol. 323, pp. 533-536, 1986. (Chapter 11)

[RuMc86] D. E. Rumelhart and J. L. McClelland, eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, Cambridge, MA: MIT Press, 1986. (Chapters 1, 11, 14, 19)

[Sarle94] W. S. Sarle, "Neural networks and statistical models," *Proceedings of the Nineteenth Annual SAS Users Group International Conference*, Cary, NC: SAS Institute, pp. 1538-1550, 1994. (Chapter 19)

[Sarle95] W. S. Sarle, "Stopped training and other remedies for overfitting," to appear in *Proceedings of the 27th Symposium on the Interface*, 1995. (Chapter 19)

[Scal85] L. E. Scales, *Introduction to Non-Linear Optimization*, New York: Springer-Verlag, 1985. (Chapters 8, 9, 12)

[Shan90] D. F. Shanno, "Recent advances in numerical techniques for large-scale optimization," in *Neural Networks for Control*, Miller, Sutton and Werbos, eds., Cambridge, MA: MIT Press, 1990. (Chapter 12)

[ShKi72] D. Sherrington and S. Kirkpatrick, "Spin-glasses," *Physical Review Letters*, vol. 35, 1972. (Chapter 19)

[SlLi91] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*, Englewood Cliffs, NJ: Prentice-Hall, 1991. (Chapter 17)

[Smit93] M. Smith, *Neural Networks for Statistical Modeling*, New York: Van Nostrand Reinhold, 1993. (Chapter 19)

[Spec90] D. F. Specht, "Probabilistic neural networks," *Neural Networks*, vol. 3, no. 1, pp. 109-118, 1990. (Chapter 19)

[Spec91] D. F. Specht, "A general regression neural network," *IEEE Transactions on Neural Networks*, vol. 2, no, 6, pp. 568-576, 1991. (Chapter 19)

[StDo84] W. D. Stanley, G. R. Dougherty and R. Dougherty, *Digital Signal Processing*, Reston VA: Reston Publishing Co., 1984. (Chapter 10)

[Stra76] G. Strang, *Linear Algebra and Its Applications*, New York: Academic Press, 1980. (Chapters 5, 6)

A-9

[Sutt84] R. S. Sutton, "Temporal credit assignment in reinforcement learning," Ph. D. Dissertation, University to Massachusetts, Amherst, MA, 1984. (Chapter 19)

[TaHo86] D. W. Tank and J. J. Hopfield, "Simple 'neural' optimization networks: An A/D converter, signal decision circuit and a linear programming circuit," *IEEE Transactions on Circuits and Systems*, vol. 33, no. 5, pp. 533-541, 1986. (Chapter 18)

[Thom75] R. F. Thompson, *Introduction to Physiological Psychology*, New York: Harper and Row, 1975. (Chapter 19)

[Toll90] T. Tollenaere, "SuperSAB: Fast adaptive back propagation with good scaling properties," *Neural Networks*, vol. 3, no. 5, pp. 561-573, 1990. (Chapter 12)

[vanT75] H. F. J. M. van Tuijl, "A new visual illusion: Neonlike color spreading and complementary color induction between subjective contours," *Acta Psychologica*, vol. 39, pp. 441-

445, 1975. (Chapter 15)

[VoMa88] T. P. Vogl, J. K. Mangis, A. K. Zigler, W. T. Zink and D. L. Alkon, "Accelerating the convergence of the backpropagation method," *Biological Cybernetics*, vol. 59, pp. 256-264, Sept. 1988. (Chapter 12)

[vond73] C. von der Malsburg, "Self-organization of orientation sensitive cells in the striate cortex," *Kybernetic*, vol. 14, pp. 85-100, 1973. (Chapter 15)

[WaHa89] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, pp. 328-339, 1989. (Chapter 19)

[Wan 90a] E. A. Wan, "Temporal backpropagation for FIR neural networks," *IEEE International Joint Conference on Neural Networks*, vol, 1, pp. 575-580, San Diego, CA, 1990. (Chapter 19)

[Wan 90b] E. A. Wan, "Temporal backpropagation: An efficient algorithm for finite impulse response neural networks," in *Proceedings of the 1990 Connectionist Models Summer School*, D. S. Touretzky, J. L. Elman, T. J. Sejnowski and G. E. Hinton, eds., San Ma-
A-10 teo, CA: Morgan Kaufmann, pp. 131-140, 1990. (Chapter 19)

[Wan 94] E. A. Wan, "Time series prediction by using a connectionist network with internal delay lines," in *Time Series Prediction: Forecasting the Future and Understanding the Past*, A. S. Weigend and N. A. Gershenfeld, eds., Reading, MA: Addison-Wesley, pp. 195-217, 1994. (Chapter 19)

[WeRu91] A. S. Weigand, D. E. Rumelhart and B. A. Huberman, "Generalization by weight elimination with application to forecasting," in *Advances in Neural Information Processing Systems 3*, R. P. Lippman, J. E. Moody and D. J. Touretzky, eds., San Mateo, CA: Morgan Kaufmann, pp. 575-582, 1991. (Chapter 19)

[Werbo74] P. J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," Ph. D. Thesis, Harvard University, Cambridge, MA, 1974. Also published as *The Roots of Backpropagation*, New York: John Wiley & Sons, 1994. (Chapter 11)

[Werb90] P. J. Werbos, "Backpropagation through time: What it is and how to do it," *Proceedings of the IEEE*, vol. 78, pp. 1550-1560, October 1990. (Chapter 19)

[WhSo92] D. White and D. Sofge, eds., *Handbook of Intelligent Control*, New York: Van Nostrand Reinhold, 1992. (Chapter 4)

[WiHo60] B. Widrow, M. E. Hoff, "Adaptive switching circuits," *1960 IRE WESCON Convention Record*, New York: IRE Part, 4, pp. 96-104, 1960. (Chapters 1, 10)

[WiSt 85] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1985. (Chapter 10)

[WiWi 88] B. Widrow and R. Winter, "Neural nets for adaptive filtering and adaptive pattern recognition," *IEEE Computer Magazine*, March 1988, pp. 25-39. (Chapter 10)

[WiZi89] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent
A-11 neural networks," *Neural Computation*, vol. 1, pp. 270-280, 1989. (Chapter 19)

# 附录B 符　　号

## 基本概念

标量：小写斜体字母..... $a$，$b$，$c$

向量：小写黑正体字母..... $\mathbf{a}$，$\mathbf{b}$，$\mathbf{c}$

矩阵：大写黑正体字母..... $\mathbf{A}$，$\mathbf{B}$，$\mathbf{C}$

## 语言

向量表示一列数。

行向量将矩阵中的一行作为一个向量(列)使用。

## 一般向量以及变换(第5章和第6章)

$$\mathcal{X} = \mathcal{A}(y)$$

## 权值矩阵

**标量元素**

$w_{i,j}^{k}(t)$

$i$－行，$j$－列，$k$－层，$t$－时间或迭代次数

**矩阵**

$\mathbf{W}^{k}(t)$

**列向量**

$\mathbf{w}_{j}^{k}(t)$

**行向量**

${}_{i}\mathbf{w}^{k}(t)$

## 偏置值向量

**标量元素**

$b_{i}^{k}(t)$

**向量**

$\mathbf{b}^{k}(t)$

## 输入向量

**标量元素**

$p_{i}(t)$

**输入向量序列中的一个向量**

$\mathbf{p}(t)$

输入向量集合中的一个向量

$\mathbf{p}_q$

## 净输入向量

**标量元素**

$n_i^k(t)$或 $n_{i,q}^k$

**向量**

$\mathbf{n}^k(t)$或 $\mathbf{n}_q^k$

## 输出向量

**标量元素**

$a_i^k(t)$或 $a_{i,q}^k$

**向量**

$\mathbf{a}^k(t)$或 $\mathbf{a}_q^k$

## 传输函数

**标量元素**

$a_i^k = f^k(n_i^k)$

**向量**

$\mathbf{a}^k = \mathbf{f}^k(\mathbf{n}^k)$

## 目标向量

**标量元素**

B-2 $t_i(t)$或 $t_{i,q}$

**向量**

$\mathbf{t}(t)$或 $\mathbf{t}_q$

## 原型输入/目标向量的集合

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \cdots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

## 误差向量

**标量元素**

$e_i(t) = t_i(t) - a_i(t)$或 $e_{i,q} = t_{i,q} - a_{i,q}$

**向量**

$\mathbf{e}(t)$或 $\mathbf{e}_q$

## 大小和维数

**层数，每层的神经元数**

$M$，$S^k$

输入向量(和目标)数，输入向量的维数

$Q$，$R$

## 参数向量(包括所有权值和偏置值)

向量

$\mathbf{x}$

在第 $k$ 次迭代

$\mathbf{x}(k)$或 $\mathbf{x}_k$

## 范数

$\|\mathbf{x}\|$

## 性能指标

$F(\mathbf{x})$

## 梯度和赫森

$\nabla F(\mathbf{x}_k)=\mathbf{g}_k$ 和$\nabla^2 F(\mathbf{x}_k)=\mathbf{A}_k$ B-3

## 参数向量的改变

$\Delta\mathbf{x}_k=\mathbf{x}_{k+1}-\mathbf{x}_k$

## 特征值和特征向量

$\lambda_i$ 和 $\mathbf{z}_i$

## 近似性能指标(单个时间步)

$\hat{F}(\mathbf{x})$

## 传输函数的导数

标量

$$\dot{f}(n)=\frac{\mathrm{d}}{\mathrm{d}n}f(n)$$

矩阵

$$\dot{\mathbf{F}}^m(\mathbf{n}^m)=\begin{bmatrix} \dot{f}^m(n_1^m) & 0 & \cdots & 0 \\ 0 & \dot{f}^m(n_2^m) & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \dot{f}^m(n_{S^m}^m) \end{bmatrix}$$

### 雅可比矩阵

$\mathbf{J}(\mathbf{x})$

### 近似赫森矩阵

$\mathbf{H}=\mathbf{J}^T\mathbf{J}$

### 灵敏度向量

**标量元素**

$$s_i^m \equiv \frac{\partial \hat{F}}{\partial n_i^m}$$

**向量**

B-4

$$\mathbf{s}^m \equiv \frac{\partial \hat{F}}{\partial \mathbf{n}^m}$$

### Marquardt 敏感度矩阵

**标量元素**

$$\tilde{s}_{i,h}^m \equiv \frac{\partial v_h}{\partial n_{i,q}^m} = \frac{\partial e_{k,q}}{\partial n_{i,q}^m}$$

**子(矩)阵(单个输入向量 $\mathbf{p}_q$)和全(矩)阵(所有输入)**

$\tilde{\mathbf{S}}_q^m$ 和 $\tilde{\mathbf{S}}^m=[\tilde{\mathbf{S}}_1^m \quad \tilde{\mathbf{S}}_2^m \quad \cdots \quad \tilde{\mathbf{S}}_Q^m]$

### 反向传播及其变形的参数

**学习速率和动量**

$\alpha$ 和 $\gamma$

**学习速率递增幅度、递减幅度以及改变的百分率**

$\eta$, $\rho$ 和 $\zeta$

**共轭梯度方向调整参数**

$\beta_k$

**Marquardt 参数**

$\mu$ 和 $\vartheta$

### 特征图术语

**神经元之间的距离**

$d_{ij}$ – 神经元 $i$ 和神经元 $j$ 之间的距离

**邻域**

$N_i(d)=\{j,\ d_{ij}\leqslant d\}$

### Grossberg 网络和 ART 网络

**加强中心和抑制周围连接矩阵**

$$^{+}\mathbf{W}^{1}=\begin{bmatrix}1 & 0 & \cdots & 0\\ 0 & 1 & \cdots & 0\\ \vdots & \vdots & & \vdots\\ 0 & 0 & \cdots & 1\end{bmatrix} \quad 和 \quad ^{-}\mathbf{W}^{1}=\begin{bmatrix}0 & 1 & \cdots & 1\\ 1 & 0 & \cdots & 1\\ \vdots & \vdots & & \vdots\\ 1 & 1 & \cdots & 0\end{bmatrix}$$

B-5

**激励和抑制偏置值**

$^{+}\mathbf{b}$ 和 $^{-}\mathbf{b}$

**时间常数**

$\varepsilon$

**相对强度**

$\bar{p}_i=\dfrac{p_i}{p}$，其中 $p=\sum_{j=1}^{s^1}p_j$

**instar 和 outstar 权值矩阵**

$\mathbf{W}^{1:2}$和 $\mathbf{W}^{2:1}$

**定向子系统参数**

$\alpha$，$\beta$ 和 $\rho=\dfrac{\alpha}{\beta}$（警戒线的值）

**ART1 学习规则参数**

$\zeta$

## Lyapunov 稳定性

**Lyapunov 函数**

$V(\mathbf{a})$

**零导数集、最大不变集和闭包**

$Z$，$L$ 和 $L^{\circ}$

**有界 Lyapunov 函数集**

$\Omega_{\eta}=\{\mathbf{a}:\mathbf{V}(\mathbf{a})>\eta\}$

## Hopfield 网络的参数

**电路参数**

$T_{i,j}$，$C$，$R_i$，$I_i$，$\rho$

**放大器增益**

$\gamma$

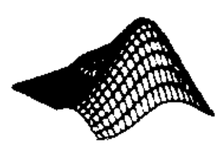
B-6

# 附录 C 软　　件

## 简介

在本书中，我们使用了数值计算和可视化软件包 MATLAB。但是需要说明，本书并不是一定要使用 MATLAB。书中的计算机练习也以采用任何编程语言实现。同样，演示软件(*Neural Network Design Demonstration*)虽然有助于理解本书的内容，但它也并不是最关键的。

到处可用的 MATLAB 软件由于其矩阵/向量表示能力以及图形输出能力，是作神经网络实验的便利环境。我们以两种不同的方式使用 MATLAB。一种是为读者提供许多在 MATLAB 上实现的练习。神经网络的许多重要特征只有在大型问题中才会体现出来，而这些问题是计算密集型的，不可能用手工计算来求解。然而，如果使用 MATLAB，不仅可以很快地实现神经网络算法，而且也可以方便地对大型问题进行测试。当然，如果没有 MATLAB，也可以使用任何其他程序设计语言来完成这些练习。



第二种方法是通过本书附带磁盘中的 *Neural Network Design Demonstration* 软件包来使用 MATLAB。这些交互式的演示说明每一章的重要概念。左边的图标表示书中对这些演示的引用。

首先应该将 MATLAB 4.0 或更新的版本，或者 MATLAB 4.0 的学生版，安装在硬盘的目录名 MATLAB(DOS 计算机)或一个文件夹(MAC 计算机)下。为了创建该目录或一个文件夹，完成整个安装过程，请根据在 MATLAB 文档中所给出的指示进行。请注意根据软件安装向导来设置路径。有一些演示需要 MathWorks 公司的 *Neural Network Toolbox* 1.0 或其更新的版本。

当这个软件装入到你的计算机的 MATLAB 目录下后，可以在 MATLAB 提示符下键入 **nnd** 进入演示程序。然后通过主菜单可以很容易访问所有的演示。



本书共有 58 个用 MATLAB 运行的演示。

## 演示文件概述

演示文件包括两个目录：NNDESIGN 和 MININNET。第一个目录 NNDESIGN 包括所有的演示以及这些演示所使用的函数。

第二个目录 MININNET 包括几个从 *Neural Network Toolbox*(NNT)软件中借用过来的关键函数。这些函数可以使多数神经网络演示在没有 NNT 的情况下运行。但是，如果你没有 NNT 的话，只需要安装这个目录即可。在机器中同时安装 NNT 和 MININNET 目录，还可能由于借用的函数在机器中存在多个版本而导致一些不可预料的结果。

## 演示程序的需求

许多演示既不需要 MININNET 目录的支持，也不需要 *Neural Network Toolbox* 的支持。有些函数需要 MININNET 目录或 *Neural Network Toolbox* 的支持，少数函数则必需 *Neural Network Toolbox* 的支持。

本附录的最后一部分列出了所有的演示及每个演示的运行要求。在安装了 NNDESIGN 目录之后，你可以在 MATLAB 内键入 **help nndesign** 看到同样的列表。

## 运行演示

你可以在 MATLAB 提示符下直接键入各个演示的名称来运行这些演示。键入 **help nndesign** 可以列出所有供你选择的演示列表。

另一种方法是运行 Neural Network Design 展示窗口(**nnd**)，然后用鼠标点击 Contents(目录)按钮，系统将会以图形的方式显示 Table of Contents(目录表)。在这里你可以使用窗口底部的按钮来选择章，根据弹出菜单来选择每一个演示。

## 声音

许多演示都使用了声音。在许多情况下加入声音是为了有助于理解演示，而另一些情况则仅仅是为了增加演示的趣味性。如果需要将声音关掉，你可以在 MATLAB 中使用下面的命令，那么所有的演示都将会静静地运行：

**nnsound off**

要把声音打开，可以使用如下命令：

**nnsound on**

你可能会注意到，演示在声音打开时比声音关闭时的运行速度要快一些。比外，除非将声音关闭，否则某些不支持声音播放的机器在声音打开时的状态下可能会出现演示运行错误。

C-2

## 演示列表

许多演示都跟随下面的两个符号来表示其运行所需要的资源：

+ 表示需要 MININNET 中的函数或 *Neural Network Toolbox* 的支持。

* 表示需要 *Neural Network Toolbox* 的支持。

**通用命令**

nnd - 展示屏幕。

nndtoc - Table of Contents(目录表)。

nnsound - 打开、关闭 Neural Network Design 演示的声音。

**第 2 章 神经元模型和网络结构**

nnd2n1 - 一个输入的神经元演示。 +

nnd2n2 - 两个输入的神经元演示。 +

**第 3 章 一个说明性实例**

nnd3pc - 感知机分类演示。 +

nnd3hamc - Hamming 分类演示。+

nnd3hopc - Hopfield 分类演示。+

**第 4 章 感知机学习规则**

nnd4db - 判定边界演示。+

nnd4pr - 感知机规则演示。+

**第 5 章 信号和权值向量空间**

nnd5gs - Gram - Schmidt 演示。

nnd5rb - 互逆基演示。

**第 6 章 神经网络中的线性变换**

nnd6lt - 线性变换演示。

nnd6eg - 特征向量游戏。

**第 7 章 有监督的 Hebb 学习**

nnd7sh - 有监督的 Hebb 演示。

**第 8 章 性能曲面和最优点**

nnd8ts1 - 泰勒级数演示 # 1。

nnd8ts2 - 泰勒级数演示 # 2。

nnd8dd - 方向导数演示。

C-3 nnd8qf - 二次函数演示。

**第 9 章 性能优化**

nnd9sdq - 二次函数最速下降法演示。

nnd9mc - 方法比较演示。

nnd9nm - 牛顿法演示。

nnd9sd - 最速下降法演示。

**第 10 章 Widrow-Hoff 学习算法**

nnd10nc - 自适应噪声消除演示。

nnd10eeg - 脑电图噪声消除演示。

nnd10lc - 线性模式分类演示。

**第 11 章 反传神经网络**

nnd11nf - 网络功能演示。+

nnd11bc - 反传计算演示。*

nnd11fa - 函数逼近演示。*

nnd11gn - 一般化演示。*

**第 12 章 反向传播算法的变形**

nnd12sd1 - 最速下降反传演示 # 1。*

nnd12sd2 - 最速下降反传演示 # 2。*

nnd12mo - 带动量的反传演示。*

nnd12vl - 可变学习速度的反传演示。*

nnd12ls - 共轭梯度线性搜索演示。*

nnd12cg - 共轭梯度反传演示。*

nnd12ms – Marquardt 步演示。*

nnd12m – Marquardt 反传演示。*

**第 13 章 联想学习**

nnd13uh – 无监督 Hebb 演示。+

nnd13hd – 带衰减的 Hebb 演示。+

nnd13edr – 衰减率影响演示。+

nnd13gis – 图形化 instar 演示。+

nnd13is – instar 演示。+

nnd13os – outstar 演示。+

**第 14 章 竞争网络**

nnd14cc – 竞争分类演示。+

nnd14cl – 竞争学习演示。+

nnd14fm1 – 一维特征图演示。*

nnd14fm2 – 二维特征图演示。*

nnd14v1 – LVQ1 演示。*

nnd14v2 – LVQ2 演示。*

C-4

**第 15 章 Grossberg 网络**

nnd15li – 漏积分器演示。

nnd15sn – 并联网络演示。

nnd15gl1 – Grossberg 层 1 演示。

nnd15gl2 – Grossberg 层 2 演示。

nnd15aw – 自适应权值演示。

**第 16 章 自适应谐振理论**

nnd16al1 – ART1 层 1 演示。

nnd16al2 – ART1 层 2 演示。

nnd16os – 定向子系统演示。

nnd16a1 – ART1 算法。

**第 17 章 稳定性**

nnd17ds – 动态系统演示。

**第 18 章 Hopfield 网络**

nnd18hn – Hopfield 网络演示。

C-5

# 索　引

索引中的页码为英文原书页码，书中页边标出原书页码。


## A

Abbreviated notation（简化符号），2－8
ADALINE network（ADALINE网络），10－2
　decision boundary（判定边界），10－4
　mean squared error（均方误差），10－4
Adaptive critic（自适应评价），19－4
Adaptive filtering（自适应滤波器），10－13
Adaptive noise cancellation（自适应噪声消除），10－15
Adaptive resonance theory（ART）（自适应谐振理论），16－2
Amacrine cell（无长突细胞），15－4
Amari, S.，15－2
AND gate（与门），4－7
Anderson, J. A.，1－2，1－3，13－2，15－2
Angle（角度），5－7
Apple and orange example（苹果和橘子实例），3－2
　Hamming network solution（Hamming 网络求解），3－8
　Hopfield solution（Hopfield 求解），3－12
　perceptron（感知机），3－3
　perceptron solution（感知机求解），3－5
　problem statement（问题描述），3－2
Application of neural network（神经网络的应用），1－5
　aerospace（航空），1－5
　automotive（汽车），1－5
　banking（银行），1－5
　defense（国防），1－6
　electronics（电子），1－6
　entertainment（娱乐），1－6
　financial（金融），1－6
　insurance（保险），1－6
　manufacturing（制造），1－6
　medical（医疗），1－6
　oil and gas（石油和天然气），1－6
　robotics（机器人），1－7
　securities（有价证券），1－7
　speech（语音），1－7
　telecommunications（电信），1－7
　transportation（运输），1－7
ART network（ART 网络），19－8
ART1
　fast learning（快速学习），16－19
　Layer 1（第一层），16－4
　Layer 2（第二层），16－10
　learning law（学习规则）
　　L1－L2，16－17
　　L2－L1，16－17
　orienting subsystem（定向子系统），16－13
　resonance（谐振），16－17
　subset/superset dilemma（子集/超集二难问题），16－17
　summary（小结），16－21
　vigilance（警戒），16－15
ART2，16－23
ART3，16－23
ARTMAP，16－23
Associative learning（联想学习）
　Hebb rule（Hebb 规则），7－4
　instar rule（instar 规则），13－11
　Kohonen rule（Kohonen 规则），13－17
　outstar rule（outstar 规则），13－17
　pseudoinverse rule（仿逆规则），7－7
　unsupervised Hebb rule（无监督 Hebb 规则），13－5
Associative memory（联想存储器），7－3
　autoassociative memory（自联想存储器），7－10
　bidirectional associative memory（BAM）（双向联想存储器），19－9
　Boltzman machine（Boltzman 机），19－9
　brain－state－in－a－box（盒中脑状态），19－9

Hopfield network（Hopfield 网络），18－5
Li－Michel network（Li－Michel 网络），19－9
linear associator（线性联想器），7－3
Associative network（联想网络），13－3
instar，13－9
outstar，13－16
Attractor（吸引子），18－11
Autoassociative memory（自联想存储器），7－10

## B

Backpropagation（反向传播），11－7
batching（批处理），12－7
CGBP，12－15
choice of network architecture（网络结构的选择），11－17
conjugate gradient（共轭梯度法），12－14
convergence（收敛性），11－19
delta－bar－delta，12－13
drawback（缺点），12－3
example（实例），11－14
generalization（推广），11－21
initial weight（初始权值），12－6
Jacobian matrix（雅可比矩阵），12－23
Levenberg－Marquardt，12－19，12－21
Jacobian calculation（雅可比计算），12－22
Marquardt sensitivity（Marquardt 灵敏性），12－24
LMBP，12－25
MOBP，12－11
performance index（性能指数），11－8
performance surface（性能曲面），12－3
Quickprop，12－14
SDBP，12－2
sensitivity（灵敏性），11－10
summary（小结），11－13
SuperSAB，12－14
variation（变形），19－4
cascade－correlation（级联相关），19－5
network pruning（网络修剪），19－5
Quickprop，19－4
regularization（规则化），19－5
Rprop，19－4
stopped training（停止训练），19－5
VLBP，11－12
Backpropagation through time（BTT）（沿时间反传），19－7
Basis set（基集），5－5
Batching（批处理），12－7
Bidirectional associative memory（BAM）（双向联想存储器），19－9
Biological inspiration of neural network（神经网络的生物学启示），1－8
Biology，psychology and neutral network（生物学、心理学和神经网络），19－10
Bipolar cell（双极细胞），15－3
Boltzman machine（Boltzman 机），19－9
Brain－state－in－a－box（盒中脑状态），19－9
Brightness constancy（亮度一致），15－8

## C

Carpenter，G，16－2
Cascade－correlation（级联相关），19－5
Cerebellar model articulation controller（CMAC）（小脑模型连接控制器），19－3
CGBP，12－15
Chain rule（链法则），11－9
Change of basis（基的变换），6－6
similarity transformation（相似变换），6－8
Choice of network architecture（网络结构选择），11－7
Circular hollow（环形空洞），8－16
Competitive learning（竞争学习），14－7
adaptive resonance theory（自适应谐振理论），16－2
ART1，16－4
ART2，16－23
ART3，16－23
ARTMAP，16－23
Fuzzy ARTMAP（模糊 ARTMAP），16－23
instar rule（instar 规则），14－7
Kohonen rule（Kohonen 规则），14－7
learning rate（学习速度），14－9
LVQ2，14－21
problem（问题），14－9
Competitive network（竞争网络），14－5
ART1，16－4

Grossberg, 15 - 13
Hamming network (Hamming网络), 14 - 3
lateral inhibition (侧向抑制), 14 - 5
learning vector quantization (学习向量的量化), 14 - 16
self - organizing feature map (自组织特征图), 14 - 12
winner - talk - all (胜者全得), 14 - 5
Conditioned stimulus (条件刺激), 13 - 3
Cone (锥体), 15 - 3
Conjugate direction (共轭方向), 9 - 16
Conjugate gradient (共轭梯度法), 9 - 15, 12 - 14
golden section search (黄金分割搜索), 12 - 17
interval location (区间定位), 12 - 16
interval reduction (区间缩小), 12 - 16
Content - addressable memory (按内容寻址存储器), 18 - 16
Contour plot (轮廓线图), 8 - 8
Contrast enhancement (对比增强), 15 - 18
Correlation matrix (相关矩阵), 10 - 6
Counterpropagation (对传), 19 - 8

## D

Decay rate (衰减速度), 13 - 7
Decision boundary (判定边界), 4 - 5, 10 - 4, 11 - 4
Delay (延时), 2 - 13
Delta rule (增量规则), 7 - 13, 10 - 7
Delta - bar - delta, 12 - 13
Descent direction (下降方向), 9 - 13
Diagonalization (对角化), 6 - 13
Directional derivative (方向导数), 8 - 5
Domain (定义域), 6 - 2

## E

Echo cancellation (回声消除), 10 - 21
EEG, 11 - 15
Eigenvalue (特征值), 6 - 10
Eigenvector (特征向量), 6 - 10
Elliptical hollow (椭圆空洞), 8 - 17
Elman network (Elman网络), 19 - 7
Emergent segmentation (应急切断), 15 - 6
Equilibrium point (平衡点), 17 - 4
Euclidean space (欧几里德空间), 5 - 3
Excitatory (激励), 15 - 10

## F

Fahlman, A. E., 12 - 14
Feature filling - in (特征填充), 15 - 6
Finite impulse response network (FIR) (有限刺激响应网络), 19 - 6
Forward perturbation algorithm (前向扰动算法), 19 - 8
Fovea (凹斑), 15 - 5
Fukushima, K., 15 - 2
Function approximation (函数逼近), 11 - 4
Functional link network (功能链网络), 19 - 3
Fuzzy ARTMAP (模糊 ARTMAP), 16 - 23

## G

Ganglion cell (神经节细胞), 15 - 4
Gauss - Newton algorithm (高斯 - 牛顿算法), 12 - 21
Jacobian matrix (雅可比矩阵), 12 - 20
Generalization (推广), 11 - 21
Generalized regression neural network (广义回归神经网络), 19 - 6
Golden section search (黄金分割搜索), 12 - 17
Gredient (梯度), 8 - 4
Gradient descent (梯度下降法), 9 - 2
Gram - Schmidt orthogonalization (Gram - Schmidt 正交化), 5 - 8
Grossberg competitive network (Grossberg竞争网络), 15 - 13
choice of transfer function (传输函数的选择), 15 - 20
Layer 1 (第一层), 15 - 13
Layer 2 (第二层), 15 - 17
learning law (学习规则), 15 - 22
relation to Kohonen law (与 Kohonen 规则的关系), 15 - 24
Grossberg, S., 1 - 3, 13 - 2, 15 - 2, 16 - 2
Group method of data handling (GMDH) (数据处理的成组方法), 19 - 3

## H

Hamming network (Hamming网络), 3 - 8, 14 - 3

feedforward layer（前馈层），3－8，14－3
recurrent layer（递归层），3－9，14－4
Hebb rule（Hebb 规则），7－4，18－18
decay rate（衰减速度），13－7
performance analysis（性能分析），7－5
supervised（有监督的），7－4，13－5
unsupervised（无监督的），7－12
with decay（带衰减的），7－12
Hebb，D. O.，1－3，7－2
Hebb's postulate（Hebb 假设），7－2
Hebbian learning（Hebb 学习），7－2
variation（变形），7－2
Hessian（赫森），8－5
eigensystem（特征系统），8－13
Hidden layer（隐含层），2－11
High－gain Lyapunov function（高增益 Lyapunov 函数），18－13
Hinton，G. E.，11－2
History of neural network（神经网络的历史），1－2
Hoff，M. E.，1－3，10－2，11－2
Hopfield model（Hopfield 模型），18－3
Hopfield network（Hopfield 网络），3－12，6－2，18－5
attractor（吸引子），18－11
design（设计），18－16
content－addressable memeoty（按内容寻址存储器），18－16
effect of gain（增益效应），18－12
example（实例），18－7
Hebb rule（Hebb 规则），18－18
high－gain Lyapunov function（高增益 Lyapunov 函数），18－13
Lasalle's invariance theorem（LaSalle 不变性定理），18－7
Lyapunov function（Lyapunov 函数），18－5
Lyapunov surface（Lyapunov 曲面），18－22
spurious pattern（伪模式），18－20
Hopfield，J. J.，1－4
Horizontal cell（水平细胞），15－4
Hubel，D. H.，14－2，15－12

## I

Illusion（幻觉），15－4
Inhibitory（抑制），15－10
Inner product（内积），5－6
Instar，13－9
Instar rule（instar 规则），13－11，14－7
Integrator（积分器），2－13
Interval location（区间定位），12－15
Interval reduction（区间缩小），12－16
Invariant set（不变集），17－13

## J

Jacobian matrix（雅可比矩阵），12－20
Jacobs，R. A.（R. A. 雅可比），12－13
Journal（杂志），19－10

## K

Kohonen rule（Kohonen 规则），13－15，14－7
graphical representation（图形表示），14－7
Kohonen，T.，1－13，13－2，15－2

## L

LaSalle's corollary（LaSalle 推论），17－14
LaSalle's Invariant Theorem（LaSalle 不变性定理），17－13
invariant set（不变集），17－13
set（集）
L，17－13
Z，17－12
Lateral inhibition（侧向抑制），14－5
Layer（层），2－9
competitive（竞争），14－5
problem（问题），14－9
hidden（隐含），2－11
output layer（输出层），2－11
superscript（上标），2－11
Le Cun，Y.，11－2
Leaky integrator（漏积分器），15－9
Learning rate（学习速度），9－3，10－8
competitive learning（竞争学习），14－9
stable（稳定的），9－6，10－10
Learning rule（学习规则），4－2
ART1，16－21
backpropagation（反向传播），11－7
competitive learning（竞争学习），14－7

delta rule（增量规则），7－13
Grossberg competitive network（Grossberg 竞争网络），15－22
Hebb rule（Hebb 规则），7－4
Hebbian learning（Hebb 学习），7－2
learning vector quantization（学习向量的量化），14－16
LMS algorithm（LMS 算法），10－7
local learning（局部学习），13－5
perceptron（感知机），4－8，4－13
proof of convergence（收敛性证明），4－15
performance learning（性能学习），8－2
pseudoinverse rule（仿逆规则），7－7
reinforcement learning（增强学习），4－3
supervised learning（有监督的学习），4－3
unsupervised learning（无监督的学习），4－3
Widrow－Hoff，7－13
Learning vector quantization（LVQ）（学习向量的量化），14－16
subclass（子类），14－17
Levenberg－Marquardt algorithm（Levenberg－Marquardt 算法），12－19，12－21
Jacobian calculation（雅可比计算），12－22
Jacobian matrix（雅可比矩阵），12－20
Li－Michel network（Li－Michel 网络），19－9
Linear associator（线性联想器），7－3
Linear independence（线性无关），5－4
Linear separability（线性可分性），4－19
Linear transformation（线性变换），6－2
change of basis（基变换），6－6
domain（定义域），6－6
matrix representation（矩阵表示），6－3
change of basis（基变换），6－6
range（值域），6－2
Linear vector space（线性向量空间），5－2
LMBP，12－25
LMS algorithm（LMS 算法），10－2，10－7
adaptive filtering（自适应滤波），10－13
adaptive noise cancellation（自适应噪声消除），10－15
analysis of convergence（收敛性分析），10－9
learning rate（学习速度），10－8
stable learning rate（稳定的学习速度），10－10
Local learning（局部学习），13－5
Long－term memory（LTM）（长期记忆），15－12，15－22
LVQ2，14－21
Lyapunov function（Lyapunov 函数），17－12
Lyapunov stability theorem（Lyapunov 稳定性定理），17－16

## M

Mach，E.，1－2
Marquardt algorithm（Marquardt 算法），12－19
Marquardt sensitivity（Marquardt 灵敏度），12－24
Matrix representation（矩阵表示），6－3
change of basis（基变换），6－6
diagonalization（对角化），6－13
McClelland，J. L.，1－4，11－2
McCulloch，W. S.，1－3，4－2
Mean squared error（均方误差），10－4，11－8
Memory（存储器）
associative（联想），7－3
autoassociative（自联想），7－10
Mexican－hat function（墨西哥帽形函数），14－11
Minima（极小点），8－7
first－order condition（一阶条件），8－10
global minimum（全局极小点），8－7
necessary condition（必要条件），8－9
second－order condition（二阶条件），8－11
strong minimum（强极小点），8－7
sufficient condition（充分条件），8－11
weak minimum（弱极小点），8－7
Minsky，M.，1－3，4－2
MOBP，12－11
Modular network（模块化网络），19－4
Momentum（动量），12－9，13－7
Multilayer perceptron（多层感知机），11－2

## N

Negative definite matrix（负定矩阵），8－11
Negative semidefinite（半负定），8－11
Neighborhood（邻域），14－12
Neocognitron（新认知机），19－8
Network architecture（网络结构），2－9
layer（层），2－9

multilayer（多层），2－10
Network pruning（网络修剪），19－5
Neural network journal（神经网络杂志），19－10
Neural network textbook（神经网络教科书），19－11
*Neural Network Toolbox* for MATLAB（MATLAB 的 *Neural Network Toolbox*），1－5
Neuron model（神经元模型），2－2
multiple－input neuron（多输入神经元），2－7
single－input neuron（单输入神经元），2－2
transfer function（传输函数），2－3
Newton's method（牛顿法），9－10
Nilsson，N.，14－2
Noise cancellation（噪声消除）
adaptive（自适应），10－15
echo cancellation（回声消除），10－21
Nonlinear autoregressive moving average（NARMA）network（非线性自回归移动平均网络），19－7
Norm（范数），5－7

## O

On－center/off－surround（加强中心/抑制周围），14－11，15－14
Optic disk（光盘），15－5
Optimality（优化）
first－order condition（一阶条件），8－10
necessary condition（必要条件），8－9
second－order condition（二阶条件），8－11
sufficient condition（充分条件），8－11
Optimization（优化）
conjugate gradient（共轭梯度法），9－15，12－14
descent direction（下降方向），9－3
Gauss－Newton（高斯－牛顿法），12－21
Levenberg－Marquardt（Levenberg－Marquardt 算法），12－19，12－21
Newton's method（牛顿法），9－10
quadratic termination（二次终结法），9－15
steepest descent（最速下降法），9－2
stable learning rate（稳定的学习速度），9－6
Oriented receptive field（定向接受区），15－20
Orienting subsystem（定向子系统），16－13
Orthogonality（正交性），5－7
Orthonormal（标准正交），5－9
Outstar，13－16
Outstar rule（outstar 规则），13－17

## P

Papert，S.，1－3，4－2
Parker，D.B.，11－2
Pattern classification（模式分类），11－3
Pavlov，I.，1－2
Perceptron（感知机），3－3
architecture（结构），4－3
constructing learning rule（构造学习规则），4－10
decision boundary（判定边界），4－5
learning rule（学习规则），4－8，4－13
proof of convergence（收敛性证明），4－15
multilayer（多层），11－2
multiple－neuron（多神经元），4－8
single－neuron（单神经元），4－5
test problem（测试问题），4－9
training multiple－neuron perceptron（训练多神经元感知机），4－13
two－input case（双输入情况），3－4
unified learning rule（统一的学习规则），4－12
Performance index（性能指数），8－2，11－8
quadratic function（二次函数），8－12
Performance learning（性能学习），8－2
Pipelined recurrent neural network（PPRN）（流水线递归神经网络），19－6
Pitts，W.H.，1－3，4－2
Polynomial network（多项式网络），19－3
functional link network（功能链网络），19－3
group method of data handling（GMDH）（数据处理的成组方法），19－3
Sigma－Pi network（Sigma－Pi 网络），19－4
Positive definite（正定），17－5
Positive definite matrix（正定矩阵），8－11
Positive semidefinite（半正定），8－11，17－5
Probabilistic neural network（概率神经网络），19－6
Projection（投影），5－8
Prototype pattern（原型模式），18－16
Pseudoinverse rule（仿逆规则），7－7

## Q

Quadratic function（二次函数），8－12
circular hollow（环状空洞），8－16

elliptical hollow（椭圆空洞），8－17
Hessian（赫森）
eigensystem（特征系统），8－13
saddle point（鞍点），8－18
stationary valley（驻谷），8－19
Quadratic termination（二次终结法），9－15
Quickprop，12－14，19－4

## R

Radial basis network（径向基本网络），19－2
Range（值域），6－2
Real－time recurrent network（RTRN）（实时递归网络），19－7
Reciprocal basis vector（互逆基向量），5－10
Recurrent network（递归网络），2－13，2－14，17－2
Regularization（规则化），19－5
Reinforcement learning（增强学习），4－3
Resonance（谐振），16－17
Retina（视网膜），15－3
Rod（杆状体），15－3
Rosenblatt，F.，1－3，4－2，10－2，11－2，14－2
Rosenfeld，E.，1－2
Rprop，19－4
Rumelhart，D.E.，1－4，11－2

## S

Saddle point（鞍点），8－8，8－18
SDBP，12－2
Self－organizing feature map（SOFM）（自组织特征图），14－12
neighborhood（邻域），14－12
Sensitivity（灵敏性），11－10
backpropagation（反向传播），11－11
Set（集合）
L，17－13
Z，17－12
Shakespeare，W.（W.莎士比亚），1－5
Short－term memory（STM）（短期记忆），15－12，15－17
Shunting model（并联模型），15－10
Sigma－pi network（Sigma－pi 网络），19－4
Similarity transform（相似变换），6－8
Spanning a space（生成一个空间），5－5
Spurious pattern（假模式），18－20
Stability（稳定性）
asymptotically stable（渐进稳定），17－3，17－5
concept（概念），17－4
equilibrium point（平衡点），17－4
in the sense of Lyapunov（Lyapunov 意义下），17－3，17－4
LaSalle's corollary（LaSalle 推论），17－14
LaSalle's Invariance Theorem（LaSalle 不变性定理），17－13
Lyapunov function（Lyapunov 函数），17－12
Lyapunov stability theorem（Lyapunov 稳定性定理），17－6
pendulum example（单摆例子），17－6
Stability/plasticity dilemma（稳定性/可塑性二难问题），16－2
Stationary point（驻点），8－10
minima（极小点），8－7
saddle point（鞍点），8－8
Stationary valley（驻谷），8－19
Statistical physics and neural network（统计物理学和神经网络），19－10
Statistics and neural network（统计学和神经网络），19－10
Steepest descent（最速下降法），9－2
learning rate（学习速度），9－3
minimizing along a line（沿直线最小化），9－8
stable learning rate（稳定的学习速度），9－6
Stimulus－response（刺激－响应），13－2
conditioned stimulus（条件刺激），13－3
unconditioned stimulus（无条件刺激），13－3
Stopped training（停止训练），19－5
Subclass（子类），14－17
Subset/superset dilemma（子集/超集二难问题），16－17
SuperTAB，12－14
Supervised learning（有监督学习），4－3
Hebb rule（Hebb 规则），7－4
performance learning（性能学习），8－2
target（目标），4－3
training set（训练集），4－3

## T

Tapped delay line（抽头延迟线），10－13
Target（目标），4－3
Taylor series expansion（泰勒级数展开），8－2
  vector case（向量情形），8－4
Textbook（教科书），19－11
Time constant（时间常数），15－9
Time delay neural network（TDNN）（时间延时神经网络），19－6
Tollenaere，T.，12－14
Training set（训练集），4－3
  sequence（序列），13－5
Transfer function（传输函数），2－3，2－6
  competitive（竞争），2－6
  hard limit（硬极限），2－3，2－6
  hyperbolic tangent sigmoid（双曲正切S形函数），2－6
  linear（线性的），2－4，2－6
  log－sigmoid（对数-S形函数），2－5，2－6
  positive linear（正线性），2－6
  saturating linear（饱和线性），2－6
  symmetric saturating linear（对称饱和线性），2－6
  symmetrical hard limit（对称硬极限），2－6
  table（表），2－6

## U

Unconditioned stimulus（无条件刺激），13－3
Unsupervised learning（无监督的学习），4－3
  Hebb rule（Hebb规则），7－4，13－5

## V

Vector expansion（向量展开），5－9
  reciprocal basis vector（互逆基向量），5－10
Vector space（向量空间），5－2
  angle（角度），5－7
  basis set（基集），5－5
  orthonormal（标准正交），5－9
  projection（投影），5－8
  spanning（生成），5－5
  vector expansion（向量展开），5－9
Vigilance（警戒），16－15
Vision（视觉），15－3
Vision normalization（视觉规格化），15－8
Visual cortex（视觉皮层），15－4
VLBP，12－12
von der Malsburg，C.，14－2，15－12

## W

Weight indiex（权值下标），2－7
Weight matrix（权值矩阵），2－7
Werbos，P. J.，11－2
Widrow，B.，1－3，10－2，11－2
Widrow－Hoff algorithm（Widrow－Hoff算法），7－13，10－7
  adaptive filtering（自适应滤波），10－13
Wiesel，T.，14－2，15－12
Williams，R. J.，11－2
Winner－talk－all（胜者全得），14－5

[General Information]
书名=神经网络设计
作者=（美）哈根（Hagan，M.T.）等著　　戴葵等译
页数=463
SS号=10896697
出版日期=2002年09月第1版

前言
目录
出版者的话
专家委员会
译者序
前言
第１章　绪论
１．１　目的
１．２　历史
１．３　应用
１．４　生物学的启示
参考文献
第２章　神经元模型和网络结构
２．１　目的
２．１　理论和实例
２．２．１　符号
２．２．２　神经元模型
２．２．３　网络结构
２．３　小结
２．４　例题
２．５　结束语
习题
第３章　一个说明性实例
３．１　目的
３．２　理论和实例
３．２．１　问题描述
３．２．２　感知机
３．２．３　Ｈａｍｍｉｎｇ网络
３．２．４　Ｈｏｐｆｉｅｌｄ网络
３．３　结束语
习题
第４章　感知机学习规则
４．１　目的
４．２　理论和实例
４．２．１　学习规则
４．２．２　感知机的结构
４．２．３　感知机学习规则
４．２．４　收敛性证明
４．３　小结
４．４　例题
４．５　结束语
参考文献
习题
第５章　信号和权值向量空间
５．１　目的
５．２　理论和实例
５．２．１　线性向量空间
５．２．２　线性无关
５．２．３　生成空间
５．２．４　内积
５．２．５　范数
５．２．６　正交性
５．２．７　向量展开式
５．３　小结
５．４　例题
５．５　结束语
参考文献
习题
第６章　神经网络中的线性变换
６．１　目的

６．２　理论和实例
６．２．１　线性变换
６．２．２　矩阵表示
６．２．３　基变换
６．２．４　特征值和特征向量
６．３　小结
６．４　例题
６．５　结束语
参考文献
习题
第７章　有监督的Ｈｅｂｂ学习
７．１　目的
７．２　理论和实例
７．２．１　线性联想器
７．２．２　Ｈｅｂｂ规则
７．２．３　仿逆规则
７．２．４　应用
７．２．５　Ｈｅｂｂ学习的变形
７．３　小结
７．４　例题
７．５　结束语
参考文献
习题
第８章　性能曲面和最优点
８．１　目的
８．２　理论和实例
８．２．１　泰勒级数
８．２．２　方向导数
８．２．３　极小点
８．２．４　优化的必要条件
８．２．５　二次函数
８．３　小结
８．４　例题
８．５　结束语
参考文献
习题
第９章　性能优化
９．１　目的
９．２　理论和实例
９．２．１　最速下降法
９．２．２　牛顿法
９．２．３　共轭梯度法
９．３　小结
９．４　例题
９．５　结束语
参考文献
习题
第１０章　Ｗｉｄｒｏｗ-Ｈｏｆｆ学习算法
１０．１　目的
１０．２　理论和实例
１０．２．１　ＡＤＡＬＩＮＥ网络
１０．２．２　均方误差
１０．２．３　ＬＭＳ算法
１０．２．４　收敛性分析
１０．２．５　自适应滤波
１０．３　小结
１０．４　例题
１０．５　结束语
参考文献
习题

第１１章　反向传播
１１．１　目的
１１．２　理论和实例
１１．２．１　多层感知机
１１．２．２　反向传播算法
１１．２．３　例子
１１．２．４　反向传播
１１．３　小结
１１．４　例题
１１．５　结束语
参考文献
习题
第１２章　反向传播算法的变形
１２．１　目的
１２．２　理论和实例
１２．２．１　ＢＰ算法的缺点
１２．２．２　ＢＰ算法的启发式改进
１２．２．３　数值优化技术
１２．３　小结
１２．４　例题
１２．５　结束语
参考文献
习题
第１３章　联想学习
１３．１　目的
１３．２　理论和实例
１３．２．１　简单联想网络
１３．２．２　无监督的Ｈｅｂｂ规则
１３．２．３　简单的识别网络
１３．２．４　ｉｎｓｔａｒ规则
１３．２．５　简单回忆网络
１３．２．６　ｏｕｔｓｔａｒ规则
１３．３　小结
１３．４　例题
１３．５　结束语
参考文献
习题
第１４章　竞争网络
１４．１　目的
１４．２　理论和实例
１４．２．１　Ｈａｍｍｉｎｇ网络
１４．２．２　竞争层
１４．２．３　生物学意义上的竞争层
１４．２．４　自组织特征图
１４．２．５　学习向量量化
１４．３　小结
１４．４　例题
１４．５　结束语
参考文献
习题
第１５章　Ｇｒｏｓｓｂｅｒｇ网络
１５．１　目的
１５．２　理论和实例
１５．２．１　生物学的启发：视觉
１５．２．２　基本非线性模型
１５．２．３　两层竞争网络
１５．２．４　与Ｋｏｈｏｎｅｎ规则的关系
１５．３　小结
１５．４　例题
１５．５　结束语

参考文献
习题
第１６章　自适应谐振理论
１６．１　目的
１６．２　理论和实例
１６．２．１　自适应谐振概述
１６．２．２　第一层
１６．２．３　第二层
１６．２．４　调整子系统
１６．２．５　学习规则：Ｌ１－Ｌ２
１６．２．６　学习规则：Ｌ２－Ｌ１
１６．２．７　ＡＲＴ１算法小结
１６．２．８　其他ＡＲＴ体系结构
１６．３　小结
１６．４　例题
１６．５　结束语
参考文献
习题
第１７章　稳定性
１７．１　目的
１７．２　理论和实例
１７．２．１　递归网络
１７．２．２　稳定性概念
１７．２．３　Ｌｙａｐｕｎｏｖ稳定性定理
１７．２．４　单摆例子
１７．２．５　ＬａＳａｌｌｅ不变性定理
１７．３　小结
１７．４　例题
１７．５　结束语
参考文献
习题
第１８章　Ｈｏｐｆｉｅｌｄ网络
１８．１　目的
１８．２　理论和实例
１８．２．１　Ｈｏｐｆｉｅｌｄ模型
１８．２．２　Ｌｙａｐｕｎｏｖ函数
１８．２．３　增益效应
１８．２．４　Ｈｏｐｆｉｅｌｄ网络设计
１８．３　小结
１８．４　例题
１８．５　结束语
参考文献
习题
第１９章　结束语
１９．１　目的
１９．２　理论和实例
１９．２．１　前馈和联想网络
１９．２．２　竞争网络
１９．２．３　动态联想存储器网络
１９．２．４　神经网络的经典基础
１９．２．５　参考书目和杂志
１９．３　结束语
参考文献
附录Ａ　文献目录
附录Ｂ　符号
附录Ｃ　软件
索引