More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

神经网络模式识别及其实现

Pattern Recognition with Neural Networks in C++

〔美〕Abhijit S. Pandya　Robert B. Macy 著

徐勇　荆涛　等译

電子工業出版社

Publishing House of Electronics Industry

北京·BEIJING

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

- 尽管有许多有关统计模式识别的著作，但它们并不包括神经网络方法。反过来，有关神经网络的教科书也极少讨论模式识别问题。

本书试图在同一框架下讨论模式分类和神经网络方法，并使之适合于专业人员。认为神经网络是由复杂机制控制、能提供令我们惊奇或失望结果的黑盒子是不应该的。只要有了对基本理论和实际例子的理解，专业人员就能够更好地在设计中作出正确选择，使神经网络的应用更加具有预见性和更为有效。

对于每一个网络示例，我们努力为每一个方法给出直观的解释，并在合适的地方用严格的数学方法加以论证和扩充。为了丰富和更加明确地描述各种概念、模型和算法，给出了大多数所讨论的神经网络模型的实际 C++ 实现程序，提供了论题的解释图表和方法的描述。我们还提供了数学模型必要的推导过程，以便聪明的读者能够随着本学科领域新的发展，将自己新的思想加到程序中。

对于每一个方法，我们试图简单明了地阐述已知的理论成果、发展趋势，以及对如何从该方法中获得最好结果的建议。一些人也许对这些建议持有异议，但至少专业人员可以从中受益和获得实验经验。我们力图使几乎没有或不清楚神经网络背景知识的工程师们，容易理解这些方法。然而，由于所提供的材料有足够的深度，那些预先具有背景知识的读者会发现本书对他们也是有裨益的。

本书所讨论的内容，适用于计算机科学和大多数工科专业的高年级大学生和研究生作为学习神经网络方面的课程。通过学习模式识别的知识，可以很容易地观察和理解神经网络的原理和应用。此外，在个别情况下，如果需要重点学习神经方法或与其它介绍统计方法的教科书结合起来学习时，本书也适合于用作模式识别的教程。

那些具有某种神经网络范例的实践经验的读者，也许对于其中的网络结构之一已经进行过编程应用，那么他应该能够顺利地应用本书的材料。我们希望本教科书能够对加强学术和实践的基本原理研究，提供有价值的帮助。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

译 者 的 话

神经网络是人们在模仿人脑处理问题的过程中发展起来的一种新型智能信息处理理论，它通过大量的称为神经元的简单处理单元构成非线性动力学系统，对人脑的形象思维、联想记忆等进行模拟和抽象，实现与人脑相似的学习、识别、记忆等信息处理能力。神经网络在经历了 40 多年的曲折发展之后，在信息科学领域等许多应用方面已显示出巨大潜力和广阔的应用前景。

神经网络模式识别方法是近几年兴起的模式识别领域的一个新的研究方向。由于神经网络的高速并行处理、分布存贮信息等特性符合人类视觉系统的基本工作原则，具有很强的自学习性、自组织性、容错性、高度非线性、高的鲁棒性、联想记忆功能和推理意识功能等，能够实现目前基于计算理论层次上的模式识别理论所无法完成的模式信息处理工作，所以，采用神经网络进行模式识别，突破了传统模式识别技术的束缚，开辟了模式识别发展的新途径。同时，神经网络模式识别也成为神经网络最成功和最有前途的应用领域之一。研究神经网络模式识别系统，无论对于神经网络理论的发展，还是对于模式识别技术的实际应用，都具有特别重要的意义。

为适应从事神经网络和模式识别领域学习、研究的各种读者的需要，我们翻译了美国 CRC 出版公司出版的专著“Pattern Recognition with Neural Networks in C++”。本书将神经网络理论与模式识别技术有机地结合起来，重点论述神经网络在模式识别系统中的应用问题，给出了大量的用于模式识别的神经网络模型的 C++ 实现程序。其突出特点是内容新颖，概念清晰，既有深入浅出的理论分析，又有实用易行的计算机实现方法。我们将本书翻译过来介绍给国内读者，希望能为高等院校有关专业的研究生和高年级大学生提供相应的教材或教学参考书，并对我国从事神经网络和模式识别领域工作的研究人员和工程技术人员有所帮助，对该领域的研究和应用起到一定的促进作用。

承蒙电子工业出版社购买了原著的中文本版权，并给予大力支持，本书才得以顺利出版。在本书的翻译过程中，赵晓晖帮助校阅了译文的部分章节，王伟、赵岩、邓小英、魏静参加了部分翻译工作。在此一并表示衷心的感谢。

对于原书中的一些印刷错误尽可能作了订正。由于译者水平有限，翻译中错误和不妥之处在所难免，欢迎读者批评指正。

译 者

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

序　言

为什么在已经有许多优秀著作的情况下，我们还感觉有必要写一部专著，来论述模式识别这个传统的论题？答案在于，我们是要深入全面地讨论作为自然模式分类器或聚类器的神经网络。人工神经网络计算是一个极为活跃的研究领域，它的一个重要应用就是处理模式化信息以及含有潜在模式的信息。神经网络研究无论过去还是现在的迅速进展，促进了模式识别其它相关方法的发展。较之传统的统计模式识别的范畴，模式识别已经成为更为广泛意义上的方法学。将人工神经网络计算引入传统模式识别中，则使模式识别这一方法学变得更加具有新意和有力，这正是本书要向这方面的专业人员介绍的内容。

模式识别系统是基于复杂模式或对象的可测属性，或由可测属性得到的特征，对这些模式进行自动分类或聚类的系统。根据这个观点，神经网络可以被看作为一种识别模式的系统。然面，确切地说，神经网络的优势是完成寻求潜在规则方面的任务。所以，在这个意义上，神经网络的研究和模式识别的研究是相互交叉覆盖的，任何一方缺了对方，它的学科就不是真正完整的。在一些通常明确不属于模式识别的领域（例如控制）中，神经网络得到了许多有效的应用。这些领域利用的也是神经网络探测和识别输入空间潜在规则的能力，从这个意义上说，它们也可以被认为是属于模式识别的范畴。

神经网络应该体现出多少生物系统的性能，一直是一个有争议的问题。本书主要讨论工程方法，不涉及任何生物学上似乎真实的论述。尽管如此，我们也认识到，设计一个系统时，观察一下能很好完成预期功能的其它系统是很有用的。生物系统是超级模式识别器。人们观察鸟类，分析其飞行特性（例如，翅膀形状、质量/体积比，等等）并进行模仿，对设计第一架飞机起了很重要的作用。但是，在模仿过程中，人们必须避免跟着生物系统亦步亦趋，否则，就不可能出现能进行超音速飞行的航天飞机。所以，我们可以相信，在从生物系统获得灵感的同时，人工神经系统最终一定能使其本身更加真正有效。

我们的目的是引导人们实际应用人工神经网络。尽管已有许多神经网络方面的教科书，并且可以作为很好的参考资料，但是以我们所见还不能完全满足该领域应用的需要，其中一些理由如下：

- 大多数有关人工神经系统的论著，是由从事理论研究的学者撰写的，对一般理论进行了很好的论述，但通常忽略了各种网络结构和学习算法应用的实用信息。它们经常给出的是某段程序代码和个别算法的计算机实现结果，面很少提供能有助于实验设计的完整程序代码。
- 最近出版的神经网络论著中，绝大部分是将个人或几个人的研究论文编辑在一起，阅读起来必须需要预先从事相关专业领域的经历。对于一些神经网络算法，即使加以详尽论述，网络所完成的计算细节还是不容易实现，面且它们的结果也不容易跟踪。
- 目前出版了几本有关神经网络方案的详细说明书，其中包含有简化公式和实现相应方法的 C++ 程序，但是这些书是为外行编写的。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

- 尽管有许多有关统计模式识别的著作,但它们并不包括神经网络方法。反过来,有关神经网络的教科书也极少讨论模式识别问题。

本书试图在同一框架下讨论模式分类和神经网络方法,并使之适合于专业人员。认为神经网络是由复杂机制控制、能提供令我们惊奇或失望结果的黑盒子是不应该的。只要有了对基本理论和实际例子的理解,专业人员就能够更好地在设计中作出正确选择,使神经网络的应用更加具有预见性和更为有效。

对于每一个网络示例,我们努力为每一个方法给出直观的解释,并在合适的地方用严格的数学方法加以论证和扩充。为了丰富和更加明确地描述各种概念、模型和算法,给出了大多数所讨论的神经网络模型的实际 C++ 实现程序,提供了论题的解释图表和方法的描述。我们还提供了数学模型必要的推导过程,以便聪明的读者能够随着本学科领域新的发展,将自己新的思想加到程序中。

对于每一个方法,我们试图简单明了地阐述已知的理论成果、发展趋势,以及对如何从该方法中获得最好结果的建议。一些人也许对这些建议持有异议,但至少专业人员可以从中受益和获得实验经验。我们力图使几乎没有或不清楚神经网络背景知识的工程师们,容易理解这些方法。然而,由于所提供的材料有足够的深度,那些预先具有背景知识的读者会发现本书对他们也是有裨益的。

本书所讨论的内容,适用于计算机科学和大多数工科专业的高年级大学生和研究生作为学习神经网络方面的课程。通过学习模式识别的知识,可以很容易地观察和理解神经网络的原理和应用。此外,在个别情况下,如果需要重点学习神经方法或与其它介绍统计方法的教科书结合起来学习时,本书也适合于用作模式识别的教程。

那些具有某种神经网络范例的实践经验的读者,也许对于其中的网络结构之一已经进行过编程应用,那么他应该能够顺利地应用本书的材料。我们希望本教科书能够对加强学术和实践的基本原理研究,提供有价值的帮助。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

目　　录

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

第一章 引 言

什么是模式？模式实际上可以说是存在有某种基本结构组织的排列或有序化。我们可以把整个世界看作是由模式构成的。Watanabe[1985]把模式定义成一个能够给出名字，但不能明确定义的一个实体。

模式也可称为对某一事物或其它一些感兴趣项目的定量或结构上的描述。一组具有公共特性的模式可以看作为一个模式类。通过机器进行模式识别的主要问题，就是如何采用更好的处理技术自动地、尽可能少由人介入地把模式分到各自的类中。例如：在邮局根据用五位数字的邮区代码自动分拣邮件的机器，就要识别各位数字。在这种情况下有 10 个模式类，每个模式类中有 10 个数字。邮区代码识别机的功能就是识别有一定几何图形的模式(每个模式代表一个输入数字)，即识别出它是可用的某个模式类中的哪一个。

模式可以用一个由标准激励或用取自标准激励和它们相互关系的属性组成的矢量来表示。通常，一个模式的特性是通过组成它的各元素的顺序来表示，而不是通过这些元素的内在性质来表示。广泛地说，模式识别就是把测量结果、激励或输入模式，划分或分配到有意义的类别中。自然地，它包含从来自不相关细节的背景数据提取重要属性。语音识别是从代表语音的波形中映射变换出话音。在字符识别中，模式识别就是从象素(或笔划)矩阵中识别出字符和词。模式识别的其它例子包括：核实签名、从象素图中识别人的面孔以及敌友识别。同样地，一个接收声纳数据的系统可以通过输入数据判定目标是一艘潜艇还是一条鱼，这样的系统就是一个模式识别系统。

1.1 模式识别系统

对于一个典型的模式识别系统，类的判别仅仅是整个系统任务的一个方面。总的来说，模式识别系统以“原始”测量结果的形式接收数据，这些原始数据集合形成一个激励矢量。寻求存在于激励矢量内部特征中的相关属性，是这种系统一个典型的基本要求(在一些情况下可能是全部要求)。这些能够更真实、更清晰地表达模式基本结构相关属性的一个有序集合构成一个特征矢量。

类别仅仅是根据问题的性质，可以或不可以必须确定的属性之一。属性可以是离散的值、布尔实体、按句法规则的句子或模拟值。在模式识别中，学习就相当于模式特征和属性之间相互关系规则的判定。

通常来说，实际的图像识别系统除了识别机器本身外通常还包括几级。在集中研究神经网络识别方法之前，我们先简要描述一下某种典型的识别系统[Chen,1973]。

图 1.1 中示出了一个典型模式识别任务的所有方面，它们包括：

预处理：把图像划分成独立的识别对象(即字符等)。

此外，它还能按比例变换图像以便将任务集中于对识别对象的处理上。

特征提取：提取各个模式的高层信息使其容易识别。

分类器：分类器识别出模式所属的类别或者大体上确定给定模式的属性。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

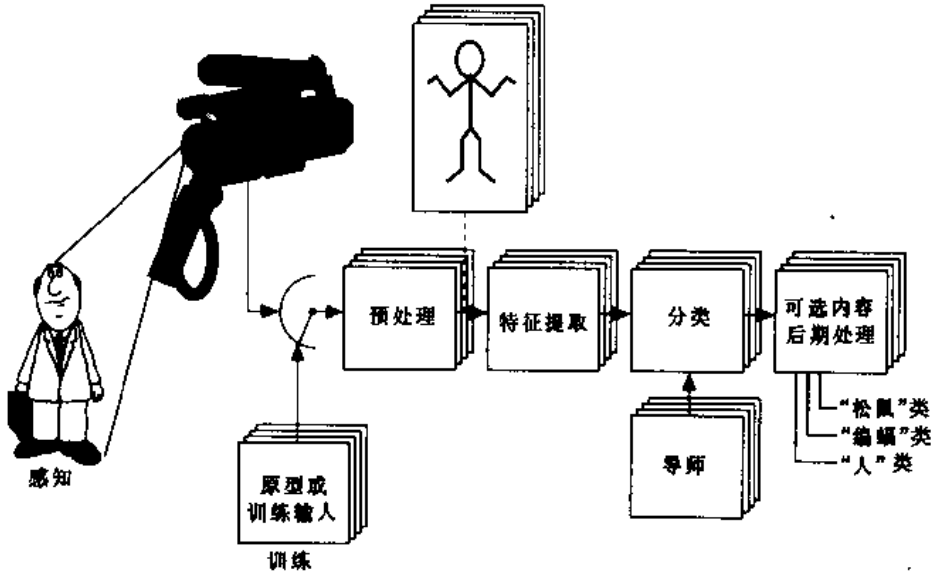


图 1.1 模式识别系统的构成

相关处理:相关处理器通过提供关于识别对象周围环境的相关信息来提高识别精度。例如,在字符识别的情况下,它可以是字典或语言模型支持。

图 1.2 给出了一个典型模式识别系统的设计步骤。如何选择合适的传感器、预处理技术和判决算法,由这个领域问题的特性来确定。不像专家系统,在设计中,特定领域的知识是隐含的并且不能通过一个独立的模块来代表。

模式分类系统能完成:

(1)有监督分类:对一个给定模式可以识别它是一个已知的或已定义的类别的那一类;

(2)无监督分类或聚类:对一个模式需要把它分到一个到目前为止还未知的模式类别中。

模式识别可以是静态的,也可以是动态的。在异步系统的情况下,时间或相继顺序不起任何作用。这种情况可以使用静态模式识别。图像识别/理解属于这一类。对动态模式识别,相对定时是很重要的。输入和输出之间时间关系具有重要作用,学习过程必须决定控制这些时间关系的准则。这包括诸如使用人工神经网络进行控制,或使用神经网络来预测等的一些应用。例如,在识别手写字符的情况中,在一个数字化的便笺簿上,笔划出现的顺序可提供很多对识别处理有用的信息。

当各类别间具有重叠区时(参见图 1.3),模式识别的任务就更复杂了。在这种情况下,识别系统必须设法使分类误差最小化。分类误差很明显地受到训练集中样本数目的影响。一些研究者(例如 Jain 和 Chandrasekaran[1982],Fukunaga 和 Hayes[1989],Foley[1972])已经讨论过这个问题。

设计一个模式识别的三种主要方法为:(1)统计方法;(2)按句法规则的或结构化方法;以及(3)人工神经网络方法。统计模式识别技术使用统计信息和估计理论的结果,去获得从表达空间到解释空间的映射。这些结果依靠一个合适的特征值组合来确定,这些特征值提供两种类别之间差别的测度。然而在某些情况下,特征本身并不是重要的。而关于模式类别或模式属性的关键信息,包含在特征的结构化关系之中。有关形象化的模式(由可识别的形状表示其

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@hotmail.com

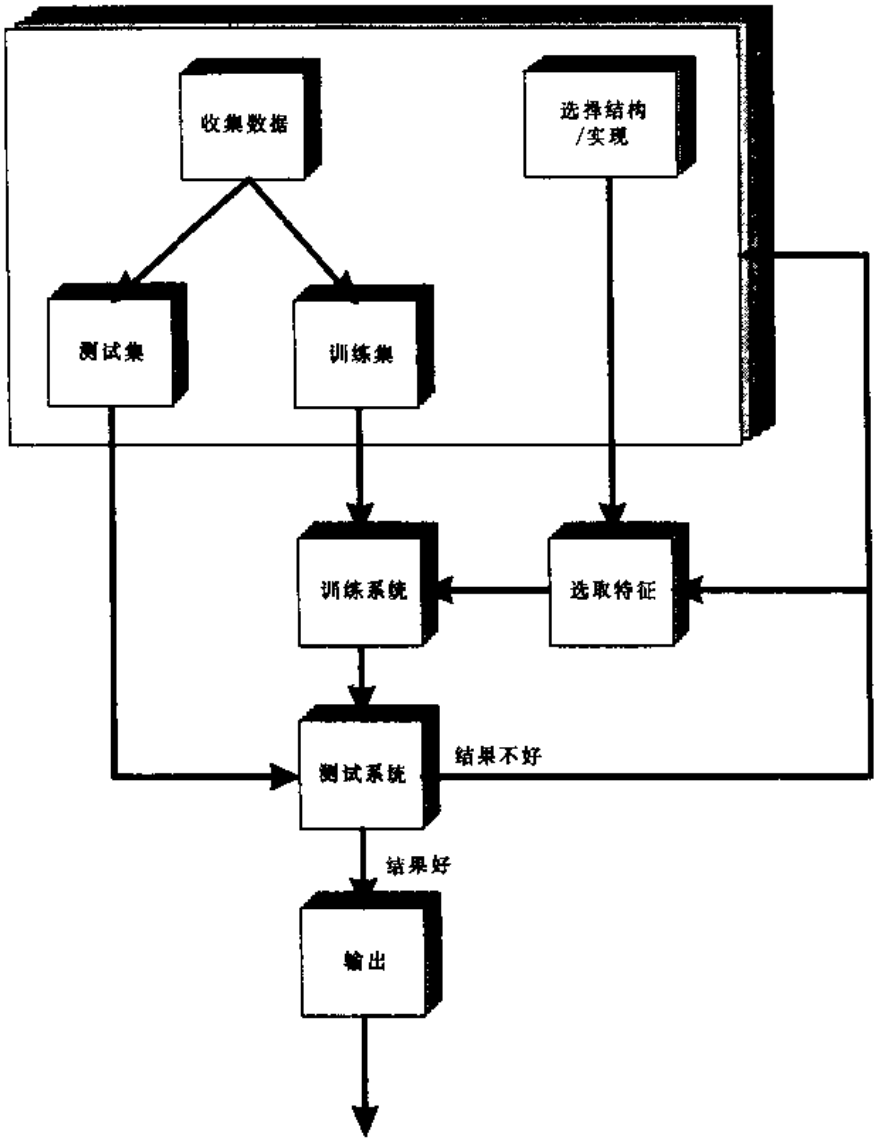


图 1.2　设计一个用于模式识别的学习机的过程流程图

特征)识别的应用,诸如字识别、染色体鉴别、基本粒子碰撞照片等都属于这一类。既然按句法规则的模式识别具有统计模式识别方法所缺乏的结构处理能力,那么它的目标就是处理这方面的问题。在这个领域的许多技术,都源自于数学语言学的早期工作和计算机语言学的研究成果。在这个领域,像 Watanabe[1972],Fu[1974,1977],Gonzalez 和 Thomason[1978]等已有大量文献来论述这些问题。

尽管存在着许多好的统计的、按句法规则的(以语法为基础)和图解的方法可用于模式识别,我们这本书还是仅限于讨论各种基于人工神经网络的方法。然而,统计的方法和神经网络技术之间的关系非常密切,所以也讨论了适当的统计方法。除此之外,不应该忽视的是神经识别器能够并且已经用于和其它类型诸如缓冲模式匹配器等识别机器的组合之中。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

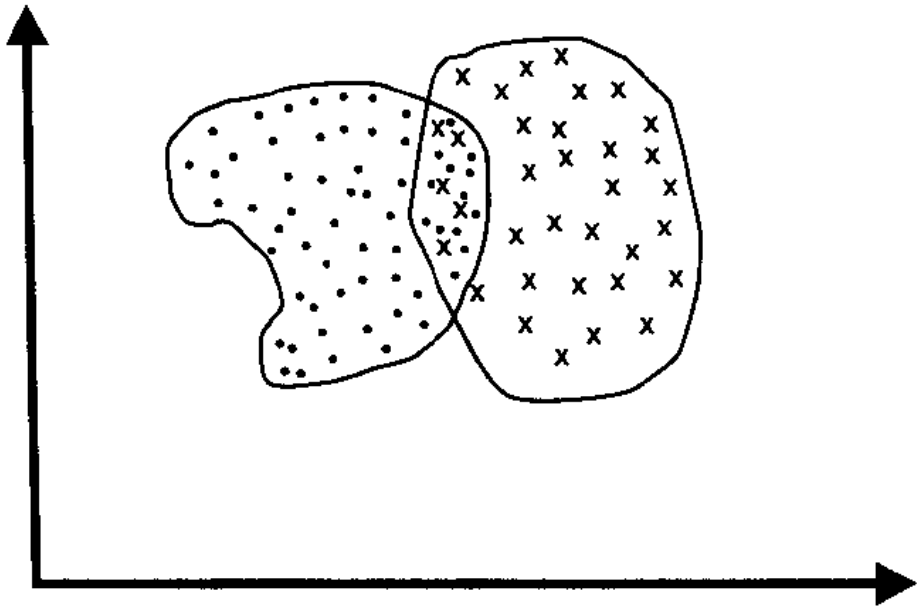


图 1.3 模式空间中的两类模式，可以看到在重叠区域中的模式属于两个模式类

1.2 人工神经网络方法的产生

计算机作为一个不仅仅只是用于计算的机器，它的发展标志着模式识别领域的诞生。我们已经看到，关于使用机器来执行通常和人的行为相关的智能任务的研究兴趣的提高。模式识别技术就是在机器智能领域已经使用的重要工具之一。识别终究可以看作是生物组织的一种基本属性。生物系统(包括人类)模式识别能力的研究，属于诸如心理学、生理学、生物学和神经科学等学科的范围。对于用机器实现一个给定任务的识别的实际技术和用于设计这些系统的必要数学构架研究，属于工程、计算机科学和应用数学领域。随着神经网络技术的出现，在从事生物系统领域研究的工程师和学生以及心理学家、生理学家、语言学家等等之间，建立了共同的研究领域。可以指出，在模式识别和神经网络理论中使用的数学运算，通常形式上相似甚至相同。所以，从数学的角度，有很好的理由将这两个领域的内容放在一起讲授。

识别模式(并以识别为基础产生动作)是所有生物系统共有的主要活动。总的来说，生物系统，特别是人类，是已知的最灵活、有效和万能的模式识别器；而且他们的行为提供了用于研究模式识别问题的足够数据。例如，我们能够不费力地识别出手写字符，尽管这个字符写得有失真、遗漏或较大的变化。在语音识别中，也可以发现同样的能力。人也具有在只有模式的一部分存在时，根据模式的相互联系来获得信息的能力。以鸡尾酒会中出现的现象为例，在酒会大厅任一方向，当人们的谈话中提到你的名字，你都会听到，即使这时因为噪音大多数谈话都听不到。类似地，你能在远处的人群中辨别出朋友的身影，即使这时大多数身影你都看不到。

人类做决定的过程通常与模式识别有关。人类善于寻找模式的相关关系，并根据它们提取规律性。这种观察力使人们能按预期想法行动，这样可以缩短反应时间并给出反应行为的界限。人们通常把机器设计成以某些特定事件的出现作出的反应为基础进行工作。这样使它们在诸如控制等应用上的速度减慢了。被识别模式的特性可以是感觉识别或理性识别。前者包括用诸如视觉或听觉刺激等感觉信息对具体实体作出识别。物体、字符、音乐、语音、签名等的识别可以看作是感觉识别的例子。另一方面，理性识别包括诸如对解决一个问题或一个旧

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

观点的识别。它包括抽象实体而且在这种情况下不需要外界刺激。在这本书中,我们将只涉及具体实体的识别。

然而,模式识别的真正问题是产生一种理论。这种理论通过一种机器能稳健识别物体的方式来确定其特性。生物系统工作方式的研究,使人们对如何解决这个问题有了深入了解。图 1.4 中的图像表示了我们所讨论问题的复杂性。在图 1.4(a)中,图像为象素间具有明显边界的人脸图像。所以,一种图像理解/模式识别算法,如果用具有不同明暗度的区域标识不同部分的表面,在识别人脸这种模式时就会有困难。另一方面,一个观察者对如图 1.4(b)所示的象素之间边界模糊不清图像,则更容易识别成熟悉人的面孔。这种能力可以归因于,在人的视觉系统中存在着的高低空间频率通道的相互作用。

图 1.4(a) 低分辨率具有可见象素网格的面部图像

工程和人工智能界的一个很大的目标,就是创造出行为如人的"智能"系统。这种智能行为能使人类在一种更为自然的方式下进行人/机交互。那就是,我们将提供感觉的和认识的能力,使计算机能以一种自然的、直观的方式与我们交流信息。设计具有判决能力的机器是众多

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

目标之一。为了达到这个目的,这样的机器应具有和人相同的模式信息处理能力。

图 1.4(b) 象素之间边界模糊不清的同一面部图像

早期的建立模式识别系统的一些工作,实际上是出于生物学上的动机。最常见的历史例子就是所谓的感知机和自适应线性组合器(ADALINE)设备。这些研究的目的,就是研制出一个识别系统,它的结构和工作方式同人的识别系统相仿。在 Minskey 和 Pappert[1969]的书中表现出的认为感知机能力有限的悲观结论,使这个领域的研究大大地减少了。后来,随着更有力的神经技术的出现,神经网络研究领域再一次充满活力。在人工神经网络和联接主义学说领域中,目前发生的重大事情使人联想到神经计算机研究兴起的早年时期。

早年对感知机方法的悲观失望,使许多研究人员把注意力集中到格式化模式信息处理方面的数学或计算机科学领域。例如:研究重点转移到了统计模式识别和具有按句法规则结构的模式分类领域。神经网络或联接主义学说,提供了通向真正拥有智能能力的计算机系统这一充满希望的道路。最近十年在人工神经网络领域中的进展,已经使我们离创造出能够像人一样工作的系统这一目标更近了。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xaut.edu.cn

Jain 和 Mao[1994]很好地讨论了人工神经网络方法和统计模式识别方法之间的共同之处。

总之,神经网络是有效的和能满足预期需要特性的天然分类器。这些特性包括:

1. 噪声抑制能力。
2. 对失真图像或模式的容错能力(再生能力)。
3. 对具有丢失信息或低质量图像的超级识别能力。
4. 并行处理潜力。

1.3 模式识别序言

一个模式可以由一组 n 个数字的测量值一个 n 维模式或测量值矢量 Z 来表示:

$$\mathbf{Z}=z_1,z_2,\cdots,z_{N_m} \tag{1-1}$$

然后,一个特征矢量 **X** 可以取自模式矢量:

$$\mathbf{X}=x_1,x_2,\cdots,x_N \tag{1-2}$$

从而,一个模式可以看作是一个在 N_m 维测量超空间,或者是在 N 维特征超空间中的一个点。典型的特征空间的维数,要选得比相应的测量空间维数低。模式分类,即把一个模式正确地从特征/测量空间映射到一个类空间中。所以,模式分类的判决过程可以归纳如下。

考虑一个由 n 维特征矢量来表示的模式:

$$\mathbf{X}=(x_1,x_2,\cdots,x_n)^T \tag{1-3}$$

其中 T 表示转置。

模式识别的任务,就是判别出一个模式属于 K 个类别 $C_1,C_2,\cdots,C_K$ 中的哪一个。注意到测量矢量代表检测到的数据,其中 N_m 是测量值的数目。例如,如果一幅图像由 $m\times m$ 具有 16 个灰度级的象素阵列来表示,那么可以确定模式矢量的维数,$n=m^2$。假设矢量 **Z** 的每个元素 z_i 是来自于 16 个可能灰度值中的一个相应的灰度值。

在语音模式识别问题中,声信号是时间的函数。感兴趣的实体是变量 t 的连续函数,而不像前面的例子中它是离散的灰度值。为了完成这种类型的分类,我们必须首先测量可观察到的抽样值特性。在这种情况下,这些抽样包括观测一段时间内的语音波形。一个模式矢量可以通过以离散的时间间隔 $t_1,t_2,\cdots,t_n$ 等对这些函数进行抽样来形成。图 1.5 示出了对波形进行抽样的值 $\mathbf{z}(t_1),\mathbf{z}(t_2),\cdots,\mathbf{z}(t_n)$。

用于语音识别的特征矢量,可以由捕获波形的前 N 个傅立叶系数构成。

模式识别系统的设计也包括选择一种合适的模式描述方法,这种描述应考虑到是机器可以接受的形式。这种决定也受到识别系统所应用问题范围的影响。例如,在人脸识别问题中,图像可以被转化成一个象素阵列,象素具有通过光敏矩阵设备(或具有帧抓取功能的摄像机)来表示的灰度。在彩色编码应用中,使用红、蓝、绿(RBG)信号的强度来描述是更合适的。

所以,特征提取器首先应设计成可以找出用于表示输入模式的合适特征,以便在特征空间中增大来自不同类的模式之间的差别。在定义了特征集和选择好适当的特征提取算法之后,典型的识别过程包括两个阶段:训练和预测。一旦特征空间的映射建立起来,训练过程就可以开始。在这个问题相关范围内有代表性的训练数据是必须要获得的。调节识别机器以便它能把特征矢量(取自训练数据)映射到分类误差最小的类别中。在第二个阶段(预测阶段)中,训

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

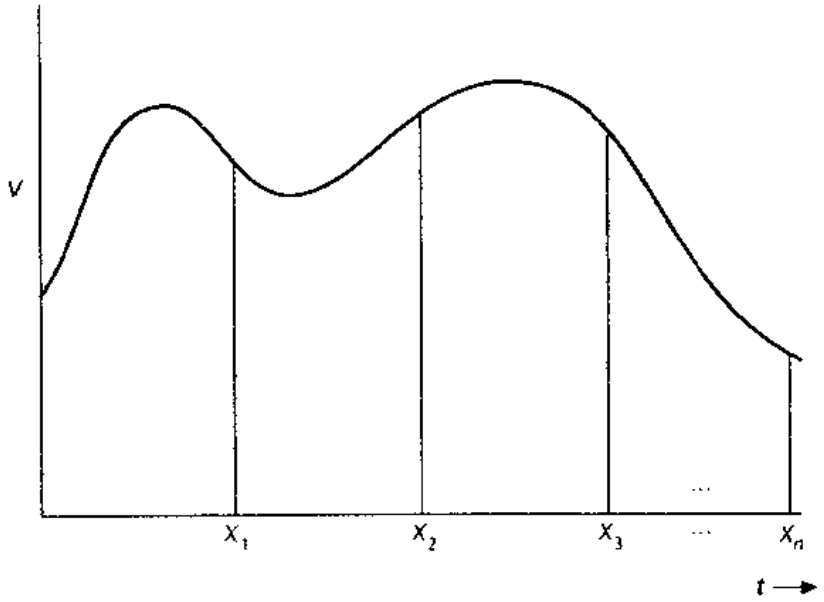


图 1.5 以离散时间间隔波形的抽样

练过的分类器把未知输入模式根据提取出的特征矢量分到某一类别中。如果预测结果不可接受,则需要重复处理,特征选择需要重做或者在不同参数下的训练需要重新执行。

两个原始数据表示(比特图或在字识别情况中的笔划)都不特别适合于直接输入到神经识别器。就像将会看到的,对变化的范围上讲"坏"的程度与讨论中的识别器特性是不同的。在上面的情况下,使用原始数据输入形式作为直接输入送到神经识别器中,存在如下一些问题:

1. 它们是非正交的。

2. 它们不可能表示出被识别模式的显著特征。

3. 它们是冗长的。多余的大输入矢量会导致需要更大的网络,使得在训练和识别过程中网络性能下降。

4. 它们对图像中的微小变化,即各种字体或笔划的变化很敏感。

5. 它们可能包含许多无关的或不相干信息,从而需要提供给识别器过多的训练。

6. 对于旋转变换,它们不会是恒定的,等等。

对于给定的这些问题,必须注意采用其它更好的数据表示方法把数据提供给识别器。

1.4 统计模式识别

在这个领域,模式分类问题可以被系统地阐述成统计判决问题。统计模式识别是一种相对成熟的理论,并且已经基于这种方法设计了许多商用的识别系统。关于这个领域,几本早期的书仍是可借鉴的,如:Tou 和 Gonzalez,1974;Duda 和 Hart,1973;Fu,1977;Fukunaga,1990。Pao[1989]的著作是一本相当好的书,它给出了从实际工程应用前景来寻找最合适技术这一思想。这些书把模式识别问题表示成多维空间中密度函数的估计问题,并且把这个超空间分成多个类别或分类区域。在这种情况下,判决是使用合适的判别函数来完成的。从而数学统计成为这个领域的基础。

由于这个学科利用判别函数来划分模式空间,那么它也可称为决策理论方法。这些函数也可称为判决函数,它是模式 X 的标量函数。以被判别函数所规定的边界包围的模式空间中

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@hotmail.com

的区域都对应有各自的类。n 维模式空间判决函数可以表示为：

$$d_k(\mathbf{x}) = w_k l(\mathbf{x}) \quad k = 1,2,\cdots,M \tag{1-4}$$

式中 w 是相应于分类 C_k 的判决函数系数，$l(\mathbf{x})$是模式 $\mathbf{x}$ 的单值实函数。

这种方法就是建立 M 个判决函数$d_1(\mathbf{x}), d_2(\mathbf{x}), \cdots, d_k(\mathbf{x})$，每个判决函数对应一类。如果模式 $\mathbf{x}$ 属于分类 C_i，则：

$$d_i(\mathbf{x}) > d_j(\mathbf{x}) \quad j = 1,2,\cdots,M, j \neq i \tag{1-5}$$

因此，我们可以得到指定判决准则的关系。为了对给定模式进行分类，首先将它代入所有判决函数中，然后把模式分配到具有最大数值的类中。那么我们得到判决边界方程为：

$$d_i(\mathbf{x}) - d_j(\mathbf{x}) = 0 \tag{1-6}$$

通过上式将 C_i 和C_j 两个类分开。图 1.6 所示为使用判别函数发生器(DFGs)的自动分类系统框图。

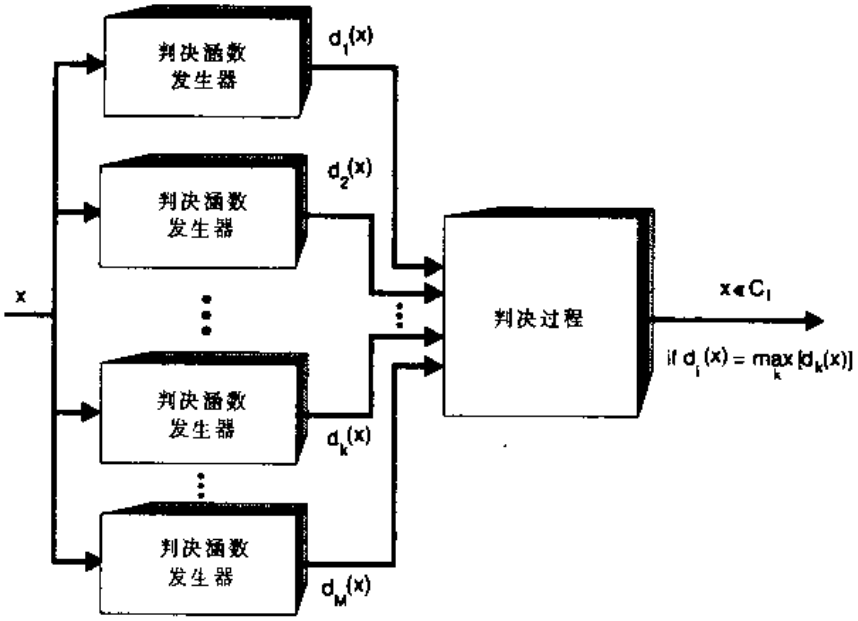


图 1.6　使用判别函数发生器(DFGs)的模式分类器框图

让我们考虑一个简单的例子。在每一个产生于二维模式空间的实体上执行两种测量，这个二维空间是容易具体化的，如职业足球运动员类和驾驶员类。在这种情况下每个模式可由两个度量值来表征，即高度和重量。图 1.7 所示为在这个二维模式空间中的两个模式类别 C_1 和 C_2。

所以，$M=2$，对图 1.7 类别 C_1 内所有模式有：

$$d_1(\mathbf{x}) > d_2(\mathbf{x}) \tag{1-7}$$

相反，对类别 C_2 内所有模式有：

$$d_2(\mathbf{x}) > d_1(\mathbf{x}) \tag{1-8}$$

我们现在可以定义：

$$d(\mathbf{x}) = d_1(\mathbf{x}) - d_2(\mathbf{x}) \tag{1-9}$$

则有：

$$d(\mathbf{x}) > 0 \qquad \mathbf{x} \in C_1 \tag{1-10}$$

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

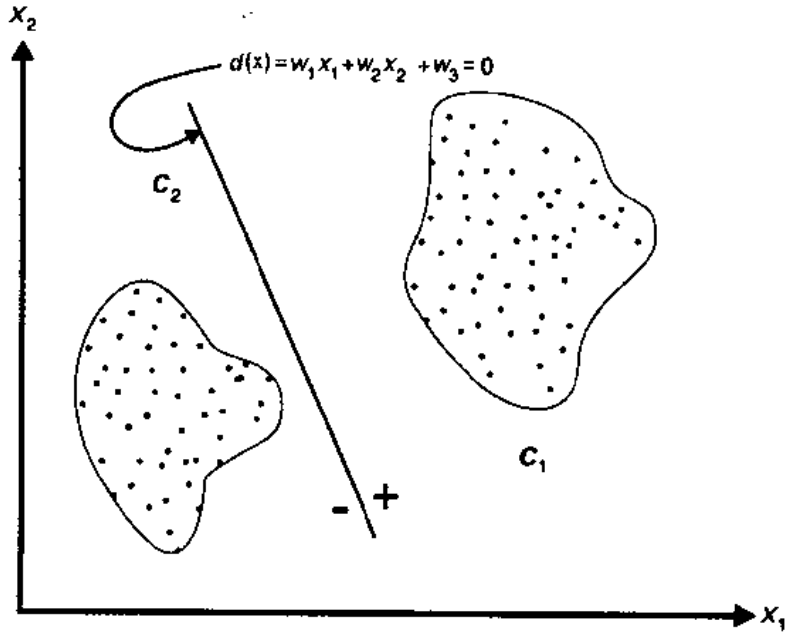


图 1.7 两个不相交模式类别特征矢量的散布图。可以用一简单的线性判决函数将它们分开

和

$$d(\mathbf{x})>0 \qquad \mathbf{x}\in C_2 \tag{1-11}$$

对于图 1.7 情况下的两个类别,可由一条直线来分开它们。那么我们有:

$$d(\mathbf{x})=w_1x_1+w_2x_2+w_3=0 \tag{1-12}$$

这是一个由等式(1-4)来描述的判决规则的特例。注意到在这种情况下(x_1,x_2)表示一个模式,并且 w 是参数。类别 C_2 的模式在边界的负侧;相反,在类别 C_1 中的所有模式在正侧。

注意到等式(1-4)中的判决函数在感觉上是很普通的,但它在 n 维模式空间能表示各种复杂的边界(包括非线性边界)。

各种各样的分类方法可以用于设计系统识别器。这些方法的选择,决定于和分类条件密度有关的可用信息种类。当 $\mathbf{x}$ 取自类别 C_i 时,分类条件密度是模式 $\mathbf{x}$ 的概率函数(用于估计分布)并可用下式表示:

$$p(\mathbf{x}/C_i),\ i=1,2,\cdots,K \tag{1-13}$$

如果所有的分类条件密度事先完全已知,模式类别之间的判决分界可使用最佳贝叶斯判决准则(参见图 1.8)。因为在这种情况下,该问题即是统计假设检验问题,那么贝叶斯分类器就能给出从给定分布中得到的最小误差。从而,贝叶斯分类器是最优分类器。然而在实际应用中,模式矢量的维数通常很高。这是因为,为了确保测量值能携带有包含在原始数据中的所有信息,测量值数目 n 相应地很大。在这种情况下,贝叶斯分类器的应用会因为它的复杂性而变得相当困难。

实际上,分类条件密度很少是事先已知的,所以需要一组训练模式来预先确定分类条件密度。在许多情况下,分类条件密度函数形式是已知的,那么主要任务就是确定一些未知参数的确切值。这种问题也称为参数判决问题。在这种情况下可以考虑更简单的参数分类器。选择具有线性的、二次的和分段判决函数的分类器是最普遍的。

如果密度函数的确切形式是未知的,那么必须估计它或使用无参数的方法去获得判决准

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xidian.edu.cn

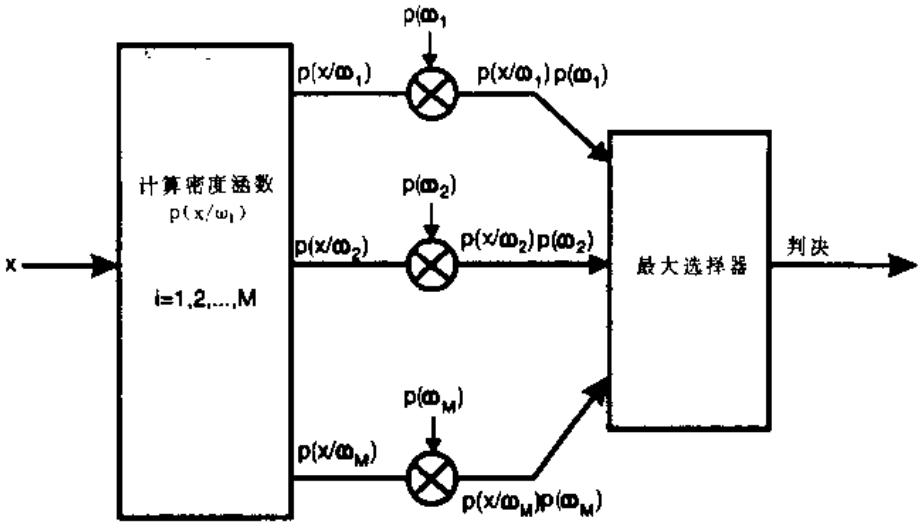


图 1.8 贝叶斯分类器(摘自 Tou 和 Gonzalez[1974])

则。虽然统计技术在一些神经方法中确实起作用,但在这本书中我们的注意力不是集中在统计模式识别上。最近邻算法属于无参数范畴,这部分内容在第八章中讨论。Karhunen-Loeve 技术通常在决定哪一个特定特征集在可容许范围内是精确的这方面是有用的,有关这部分内容在第七章中详述。径向基函数网络也依靠统计聚类方法用于训练隐层神经元,这方面内容详见第五章。图 1.9 给出了出现在统计模式识别设计中各种二分法的树形图(详细内容请参阅 Jain and Mao,[1994])。对于这些算法的进一步研究,有兴趣的读者可参阅诸如 Tou 和 Gonzalz[1974]等人的著作。

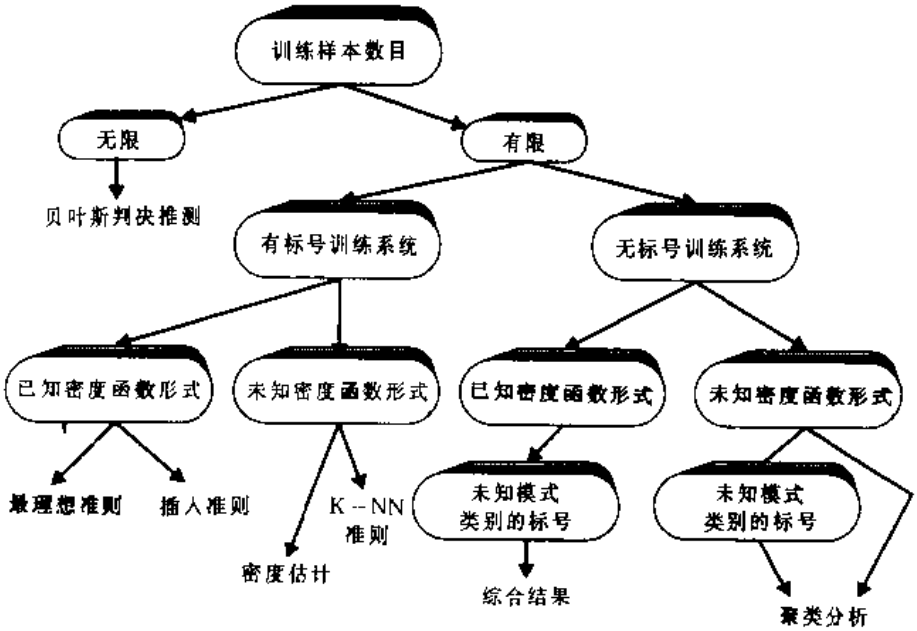


图 1.9 统计模式识别系统中的二分法(摘自 Tain 和 Mao[1994])

1.5 按句法规则的模式识别

在关于能够有意义地表示出模式的应用中,使用矢量来标志统计模式识别的方法是理想

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

的方法。然而这种方法缺乏用于处理模式结构和它们之间关系的合适形式。例如,在类似景象分析这样的应用中,一个模式的结构在分类处理中起着重要作用。在这种情况下,只要识别出各个重要组成部分,它们的结构以及它们之间的关系能被恰当地表示出来,就可建立起一个有价值的识别方案。

在二十世纪五十年代,一些研究人员(例如:参见 Chomsky[1956])在形式语言理论领域开始研究语法的数学模型。语言学家设法把这些数学模型用于描述自然语言,如英语。一旦成功地研制出这种模型,就可能为计算机提供解释自然语言的能力,达到翻译和问题求解的目的。迄今为止,这种期望还未实现,但是这些语法的数学模型已经明显地影响了编辑器设计、计算机语言和自动化理论领域的研究。按句法规则的模式识别主要受到了来自形式语言理论的观念影响。从而,各种语言的,语法的和结构的模式识别术语也常在文献中用来表示按句法规则的方法。

在按句法规则的方法中,模式是按级来表达的。也就是把模式看作是由子模式组成。这些子模式可以由其它子模式组成,或者它们就是最低级的子模式。图 1.10(a)和(b)给出两种不同染色体的结构。图 1.10(a)所示为一个称作第六染色体的模式类别原形,而图 1.10(b)为第二类原型,称为中枢末端染色体。这些模式可以根据其本原进行分解,分解后的本原定义了各种曲线形状(参见图 1.10(c))。图 1.10 中所示的每一个染色体,可以通过按顺时针方向沿其结构的边缘编码成字母或符号来表示。对于第六染色体我们可以检测以一串字符 abcdabdbabcbabdb 表示的本原。而对于中枢末端染色体可以由字符串 ebabcbab 来表示。

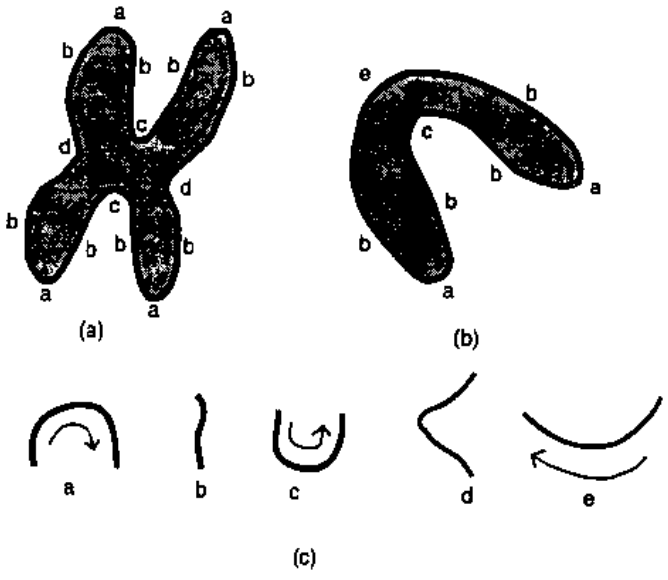


图 1.10 (a)第六染色体;(b)中枢末端染色体;
(c)可以用于编码上述两种类型染色体的五个本原

在属于第六染色体这类的各种不同结构中,我们可以将其中相似之处看作为一组从本原产生的字符串的语法规则。一组决定句法的规则,可看作是用给定符号产生句子(字符串)的语法。每一个本原可以解释成是在某种语法中允许的一个符号。从而我们可以想象两种语法 G1 和 G2,这两种语法规则允许分别产生相应于第六染色体和中枢末端染色体字符串。换句话说,由代表第六染色体的句子(字符串)构成的语言 L(G1)则由 G1 产生。类似地,由 G2 产生的 L(G2)语言是由表达中枢末端染色体的句子组成。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

因此,对于用按句法规则的模式识别方法来判别染色体类别,首先必须建立两种语法 G1 和 G2。为了建立一个给定的输入模式(即决定它是哪类染色体),它被分解成本原的一个字符串。这个句子表达了输入模式,那么现在的问题就是判决这个输入模式表达一个有效的句子的语言。在染色体识别应用中,如果相应于输入模式的句子属于语言 L(G1),那么它属于第六染色体类。另一方面,如果它属于语言 L(G2),那么它就归类于中枢末端染色体。如果它既属于 L(G1)又属于 L(G2),那么它的类别就不好判断。如果发现表达输入模式的句子对两种语言都是无效的,那么输入模式被分到一个包含所有无效模式的拒绝类中。如何建立按句法结构的类别隶属度技术,以及如何形成语法的问题在 Gonzalez 和 Thomason[1978]的文献中做了讨论。

对于多类模式识别问题必须确定更多的语法(至少每类别一个)。如果模式是一个仅属于 L(G1)而不属于其它语言的句子,那么它就被分到类别 i 中。从而在这种情况下,按句法规则的模式识别方法与两个类别的模式识别是一样的。

即使在模式是由其它数据结构表达而不是由串结构(即树和枝的结构)来表达的情况下,前面提出的想法也是有效的。

图 1.11 给出了一个使用按句法规则方法进行模式分类的典型模式识别系统。

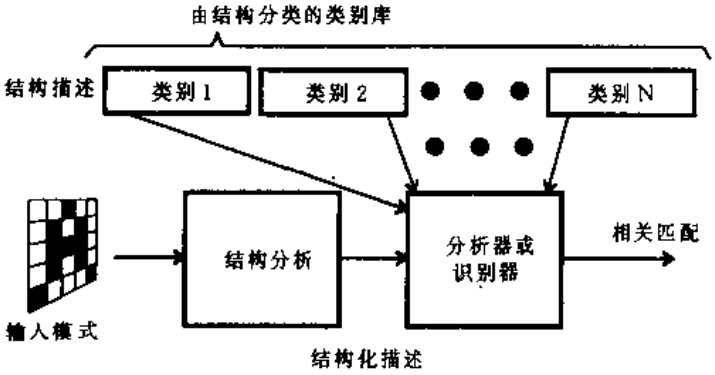


图 1.11 按句法规则分类的模式识别系统框图

1.6 字符识别问题

设计能用于识别模式的机器所面临的问题是相当多的。它似乎涉及各种学科中不同形式的问题,其中有实际的又有非常艰深的。模式识别问题的多样化使我们很难说出是什么模式识别。然而考虑一些典型的模式识别任务是这个领域范围内的一个好想法。

甚至在"识别动作"开始之前就经常需要大量的预处理。所以有大量可用的数据和最低限度的预处理,我们就可以把注意力集中在字符识别上。毕竟,字符识别是模式识别的一个典型例子。几乎不失一般性,基本的神经识别技术类似于用在其它问题领域的那些技术。而且,我们将字符识别作为一种典型的模式识别问题,目的是为了便于使用那些已经驾轻就熟的方法。

字符识别问题在模式识别有关的文献中得到了广泛研究。然而,这个问题却远没有得以解决。不过由于很容易获得大量的数据,从这个意义上说它又是易处理的。在字符识别系统之间最普遍的区别之一在于识别器主要用在手写文本还是机器打印文本上。

出现在系统中的手写文本数据可以是在线的或离线的。从便笺簿中输入的在线手写文本

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

可用一坐标序列 $v(x,y,t)$来表示,其中 t 是时间。在这里笔顺对识别起帮助作用。另外,手写文本比打印文本的约束条件要少得多。产生于手写文本上下文中的另一个问题就是如何区别词和字符之间的分隔。哪些笔划应组合在一起形成字符,以及确定词的边界在哪里是一个重要问题。从分隔的观点看非草写体文本可以分为三种主要的类别。它们是:

1. 盒子方式:各个字符都写到一个预先定义好的盒子中。

2. 准则方式:各个字符(和词)写到一条预先定义好的线上。

3. 无准则方式:各个字符(和词)写到输入表面的任意位置并且可以任意倾斜。

在盒子方式中,分隔是不重要的。在准则方式和无准则方式中,分隔问题是很难解决的问题。不应该低估分隔的特殊重要性。不正确的分隔将导致整个系统识别能力降低。这些问题和其它问题使得手写字符的识别比机器打印出的字符识别所遇到的问题更难以克服,尽管识别的目的是进行全部字体的识别。

在光学字符识别(OCR)(也可称为离线识别)情况下,典型的做法是用扫描器获得的比特图来表示打印文本或手写文本。在这种情况下虽然仍需要预处理阶段,但分隔问题就不那么重要了。比特图的预处理问题将在第三章中讨论。虽然打印文本比手写文本约束条件多,但机器打印文本的识别仍然有困难。图 1.12 给出了在机器打印文本识别中的字符混乱情况。

在下面部分,我们不会对手写字符和打印字符之间的区别进行过多的讨论,其原因在于不打算去利用笔顺信息来处理这个问题。在这个意义上说,我们处理手写字符(在大多数文献中像都是这种情况)就好像它们是一种完全不受约束的打印字符。对于草写文本识别的讨论只限于在改进型新认知机中有选择性地作简要的描述。

在字符识别的研究中,一个相当麻烦的问题是国家标准和技术机构(NIST)数据库提供了大量的测试数据,这个数据库包含的样本字符超过 100 万。如何描述这些手写文本的变化性是特别重要的,同时建造具有正确判决边界的识别系统又是极其困难的。

许多基于神经网络的识别器研究结果在文献中已有报道。如果在给定的不同样本大小、不同测试数据以及许多情况下不同目的存在模糊性,那么对基于精确性不高的数据的系统相对有效性得出结论,则是不合理的。已经遇到从大约 80% 到高达 90% 的结果的变化范围。在这个方面突出的一些问题有:

1. 识别精度达到什么程度才足够好。

2. 可靠性达到什么程度才足够好。

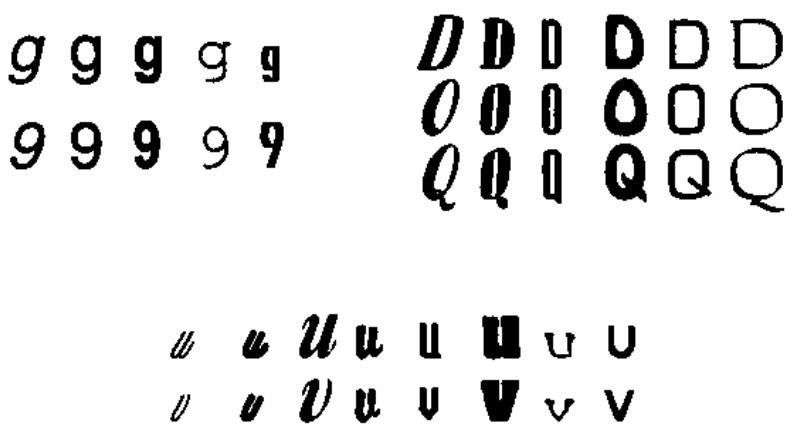


图 1.12 具有噪音字符的一些机器打印文

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xidian.edu.cn

这些问题大部分必须由其相关内容来决定。可靠性标准(在十二章中详细讨论)对于银行系统要比个人数据辅助(PDA)系统更加至关重要。相反,识别精度低到45%的范围,也许听起来还不错,但会使一个PDA用户灰心丧气到难以忍受的程度。同样,在PDA中,处理能力和性能会成为相关问题。神经系统均普遍讨论并行处理潜力,但实现这种优势的专用硬件在商用系统中通常还不能得到。

1.7 题目的组织

在下面的章节中,我们将广泛讨论用神经网络进行模式识别的方法。总的来说,我们将从讨论基本理论开始,然后,再进一步讨论特定的应用和实际建议,最后,给出实例以及带有实验数据的文献中所给出的改进。

参考书与文献

Chen, C. H., *Statistical Pattern Recognition*, Hayden, Washington, D.C., 1973.

Chomsky, N., "Three models for the description of language," *Proc. Group. Inform. Th.*, vol. 2, no. 2, pp. 113–124, 1956.

Devijer, P. and Kittler, J., *Pattern Recognition: A Statistical Approach*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

Duda, R. O. and Hart, P. E., *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, 1973.

Foley, D. H., Considerations of sample and feature size, *IEEE Trans. Inf. Theory*, IT-18, pp. 618–626, 1992.

Fu, K. S., *Syntactic Methods in Pattern Recognition*, Academic Press, New York, 1974.

Fu, K. S., *Syntactic Methods in Pattern Recognition: Applications*, Springer-Verlag, New York, 1977.

Fukunaga, K. and Hayes, R. R., "Effects of sample size in classifier design," *IEEE Trans. PAMI*, PAMI-11, pp. 873–885, 1989.

Fukunaga, K., *Introduction to Statistical Pattern Recognition*, 2nd ed. Academic Press, New York, 1990.

Gonzalez, R. C. and Thomason, M. G., *Syntactic Methods in Pattern Recognition*, Addison-Wesley, Reading, MA, 1978.

Jain, A. K. and Chandrasekaran, B., "Dimensionality and sample size considerations in pattern recognition practice," In *Handbook of Statistics 2*, P. R. Krishnaiah and Kanal, L. N. (eds.), North-Holland, Amsterdam, 1982.

Jain, A. and Mao, J., Neural Networks and Pattern Recognition, In *Computational Intelligence: Imitating Life*, J. Zurada, R. J. Marks II, and C. J. Robinson (Eds.), IEEE Press, Piscataway, NJ, 1994.

Ledley, R. S., "High speed automatic analysis of biomedical pictures," *Science*, vol. 146, no. 3461, pp. 216–223, 1964.

Miclet, L., *Structural Methods in Pattern Recognition*, Springer-Verlag, New York, 1986.

Minsky, M. and Pappert, S., *Perceptrons: An Introduction to Computational Geometry*, MIT Press, Cambridge, MA, 1969.

Pao, Y., *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley, Reading, MA, 1989.

Pavilidis, T., *Structural Pattern Recognition*, Springer-Verlag, New York, 1977.

Tou, J. T. and Gonzalez, R. C., *Pattern Recognition Principles*, Addison-Wesley, Readiag, MA, 1974.

Ullman, J. R., *Pattern Recognition Techniques*, Butterworths, London, 1973.

Watanabe, S., *Frontiers of Pattern Recognition*, Academic Press, New York, 1972.

Watanabe, S., *Pattern Recognition: Human and Mechanical*, John Wiley & Sons, New York, 1985.

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xaut.edu.cn

第二章 神经网络概述

2.1 生物神经网络概述

试图进行生物神经模拟很可能会对设计和理解人工神经系统产生重要影响。工程师经常从神经生物学获得关于神经计算机结构的新思想，解决一些复杂的问题。这些问题要比容易用传统的技术方法解决的问题更复杂。

例如，人的视觉系统就是一个相当复杂的信息处理系统[Marr, 1982; Levine, 1985; Churchland 和 Sejnowski, 1992]。人脑能处理日常的感性任务。它的视觉系统能提供对周围环境的表象，并且在 100－200ms 时间内即能提供需要与大脑相互作用的必要信息。然而，大型的传统计算机对那些相对不太复杂的任务，尚需几天的时间才能处理完[Churchland, 1986]。视网膜利用视网膜神经元间的侧抑制来提取边缘信息，从而实现对视觉信息的处理。对一幅有边界的图像，大脑皮层通过侧刺激过程来计算亮度。然后，两只眼睛中的图像信息对比，导致了更深层感觉的形成。这种深层次的感觉经常要经过许多次，然后才能由大脑皮层得出结论。对视觉系统的理解，产生了在工程应用中的视网膜和耳蜗芯片[Mead, 1989]。

另一例子，就是雷达或声纳工程师模拟蝙蝠用回波定位方法追逐或捕获目标的原理，研制出雷达或声纳。蝙蝠的大脑仅仅有李子那么大，但是它的声纳却是一个有效的回波系统。对于一个目标，比如说一个飞行的昆虫，蝙蝠通过声纳传送距离、方位、高度和相对速度和大小的信息。蝙蝠的大脑通过目标的回波完成一些复杂的神经运算，来获得所有这些信息。

Simmons 等人根据蝙蝠的回波定位原理研制成了声纳接收机模型，这种模型包括三部分：

1. 模仿蝙蝠内耳的前端部分。用于对输入波型进行编码。
2. 延迟子系统。用于计算回波延迟。
3. 用于计算回波频谱的子系统。在多目标跟踪情况下，频谱可用来估计回波的时间差。

许多类似这样的生物处理例子，都为研究人工神经网络提供了有用的线索。在这种思想指导下，我们现在将讨论大脑结构的组织和一些有趣的神经生理学方面的知识。

2.2 背 景

本书的主要兴趣仅限于从计算智能的前景来研究人工神经系统。所以，我们把神经网络看成一种数学算法，讨论怎样用这些算法来解决许许多多特定的问题。科学界也将神经网络看作模拟生命组织中生物神经网络的综合结构。关于神经网络和大脑计算方面的基本原理，请参阅 Churchland 和 Sejnowski[1992]的著作。

对于从事神经模型的计算和应用工作的学者来说，以下两个方面的神经生物学知识是很有用的：

1. 神经元的基本结构，包括轴突、树突、细胞体或体细胞和突触。
2. 突触的化学传送器，以及神经脉冲的连接方法是如何受细胞周围和细胞内各种离子活

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

动的影响。

另一方面，本书将对大脑中部分结构和功能（特别是认识功能和各部分之间的传导通路）进行讨论。

我们对实际生物结构给出一个极其粗略的概括，提供一些有关大脑功能的认识。对一些更详细的有关神经层的基本电子和化学工作过程的专题，请读者参考 Katz[1966]或 Byrne 和 Shultz[1988]的著作。Shepherd[1988,1990a,1990b]在神经结构的认识方面、给出了更深刻的见解。关于生理心理学的问题，特别是条件作用和记忆的心理学方面的知识，读者应参考 Thompson[1967]或 Carlson[1977]的著作。有几本关于大脑不同功能区域的神经解剖学的好书，例如 Truex 和 Carpenter[1969]，Nauta 和 Feirtag[1986]，Kandel 和 Schwarz[1991]等的著作。Anderson[1994]，Levine[1991]，Koch 和 Segev[1989]等人讨论了神经生理、神经模型和认识等神经网络问题。

在无法获得装置本质特性数据表示细节的情况下，也许就需要采用大量的并行机制来处理，这是神经科学提供给神经网络工程应用的有益经验。了解这些问题很有用，面且很实用，因为对这些问题已经经过了 5000 万年的自由研究和开发。读者可以参考 Anderson[1994]的著作中所进行的详细讨论。如果连最起码的生物神经系统都不讨论，那么对神经网络系统的描述将是不完全的。

2.3 生物神经网络

目前对神经网络观点的形成应归功于两个开拓者：Ramony Cajal[1934]和 Sherrington[1933]。他们提出了大脑是由不同的单元（神经元）组成的思想。大脑具有大约 10^{13}个神经单元（神经元）而且据估计大约有 10^{16}个连接（突触）。连接密度达到每神经元 1000 个连接。即使神经元的计算速度比硅逻辑门慢得多，但由于神经元和突触的数目多得令人惊愕，所以大脑仍具有极高的效率。在硅芯片中，处理速率达几纳秒，而神经元处理速率则需几毫秒。

人和其它灵长类动物（猿等）的神经系统包括三部分[Arbih,1987]，如图 2.1 所示。外界环境或人体内的感觉刺激被诸如眼、耳、鼻、皮肤等接收体转换成电脉冲。这些携带信息的信号通过大脑前向链路，传到大脑——神经系统的中枢。在图 2.1 中，由神经网络代表大脑。

大脑不断地接收信息，进行处理、判断，并同已存储的信息对比，然后作出合适的判决。最后产生必要的命令通过前向链路传送到效应器（如：说话用的舌头、声带等运动器官）。效应器把这些电脉冲转换成看得清的反应作为系统输出。同时，运动器官通过反馈链路被中枢神经系统监控，以证实动作的正确性。这些命令功能是通过诸如手、眼协调的内部或外部反馈动作来完成的。所以，整个系统类似于一个闭环的控制系统。

图 2.2 所示为一个“普通神经元”的分解图，它的主要部分为轴突、细胞体（体细胞），树突和突触。图 2.2 也给出了在细胞内、外大量存在的特有的钠、钾和氯离子的情况。

树突（像树一样有许多小分支）是来自于其它细胞的电信号的接收者。轴突是传输线，它把来自神经元的信号送走。轴突具有较平滑的表面，较少的分支和很长的长度，而树突的表而却是不规则的。体细胞（细胞体）包括细胞核（遗传物质的载体）负责给整个神经元提供必要的支援功能。这些支援功能包括产生能量，合成蛋白质等。细胞体的功能像一台信息处理机，通过它将来自许多树突的电势能加在一起。

神经元之间的相互作用通过被称为突触的基本结构和功能单元来传递。电突触的动作电

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

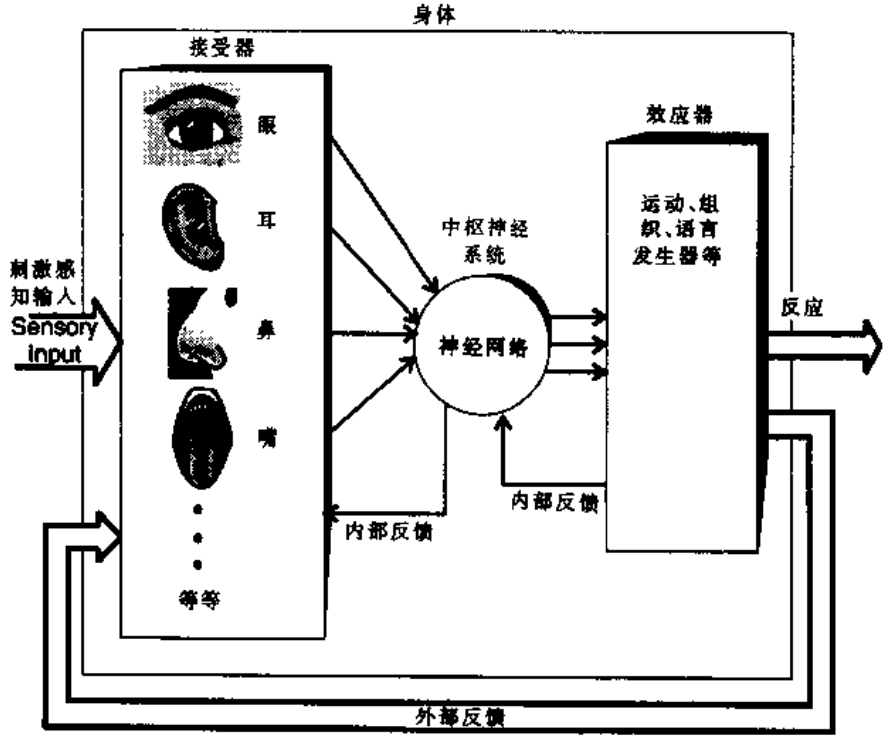


图 2.1 通过链路显示信息流动的神经系统框图

位(如图 2.2 所示)在相邻细胞间通过直接电方式进行传递。而更多的则是化学突触,它的信息传递是以化学方式来完成的。化学突触比电突触更普遍存在。突触能把兴奋或抑制强加给接收神经元。关于对突触的详细论述,请读者参考 Dayhoff[1990,第八章]的著作。

化学突触按如下方式工作[Shepherd 和 Koch,1990]:

1. 传送神经元(即突触前细胞)通过突触连接释放传送物质,将电信号转换成化学信号。

2. 化学神经传送部分引起突触后膜电位正向增加(兴奋性连接)或降低(抑制性连接)。接收神经元也称为突触后细胞。

3. 从而,在突触后细胞上,化学信号又转换回电信号,而后传播到其它神经网络单元中。

在大脑各个不同部分中都有许多各种各样的神经元,每个神经元有不同形状和大小。同样,在细胞之间不同类型突触连接数目也是很大的[Shepherd,1983]。细胞膜由髓鞘(电绝缘层)组成,它们含有用作离子传输通道的 Ranvier 节点。

它在神经细胞的活动中起着很重要的作用,如脉冲传导(对更详细的有关动作电位的产生和传输的论述,请参考 Shepherd[1983,P.107]的著作)。

图 2.3(a)所示为显示在示波器上的神经脉冲波形(动作电位),这种神经脉冲序列能由放在轴突附近的微电极记录下来。

图 2.3(b)示出了对应的神经脉冲序列。人们认为,在神经系统中长期记忆是根据突触的强度变化而定义的。突触效率的变化,可通过与学习和记忆相关的生化变化传递。这种通常的假设,其实验论据如下:

1. 在海马的神经元中,特定突触内强度的改变依赖于多种输入的结合能力。

2. 树突突起的形态变化实际上是中枢神经元中的学习和记忆动作在起作用。

3. 氯离子突触传递效率的变化使突触后接收器、凸脊中的蛋白质合成、树突毛细管的传

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

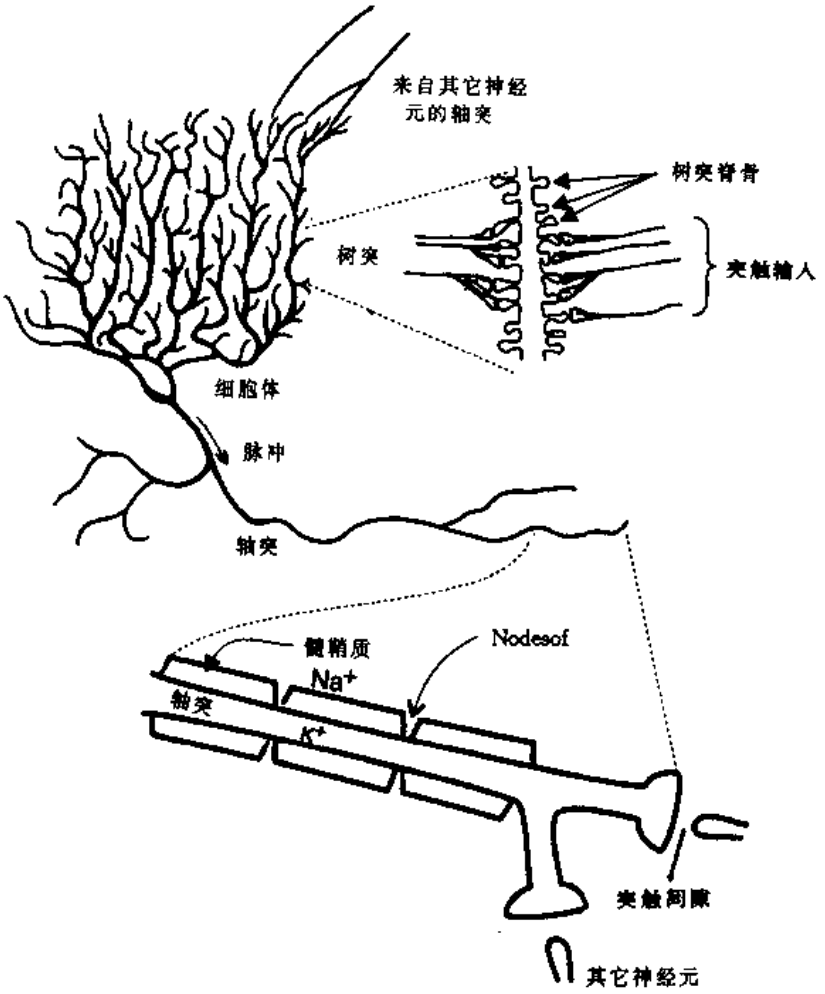


图 2.2 神经元组成(选自 Dayhoff[1990])

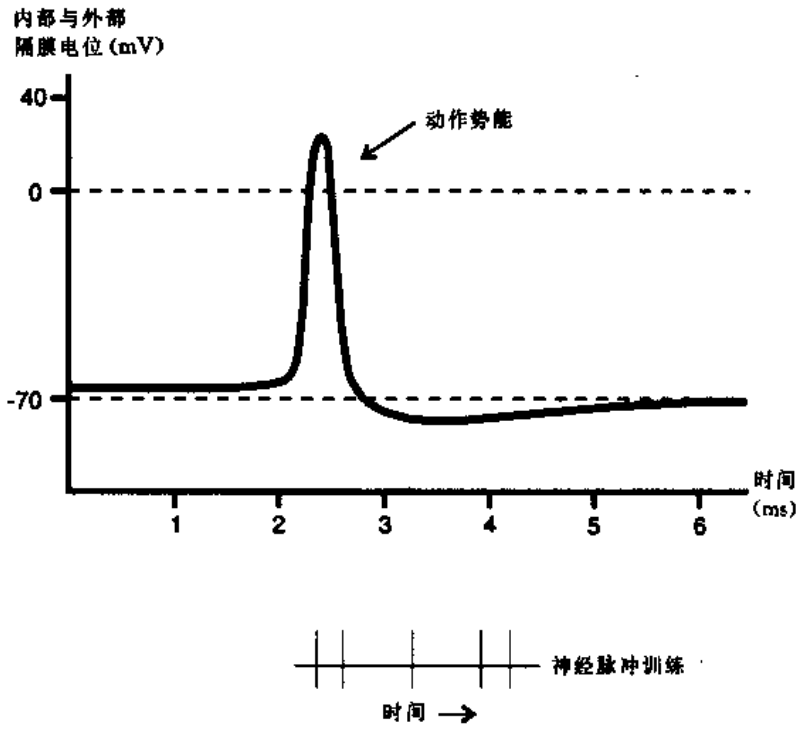


图 2.3 (a)神经脉冲波形跟踪 (b)相对应的神经脉冲序列

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

输以及突触前发送器的输出都有增大。

关于这部分内容更好的论述,请参考 MacGregor[1987]的著作。

2.4 大脑中的分层组织

对大脑局部区域剖析的广泛研究,已经揭示出了大脑的结构组织[Churchland 和 Sejnowski,1992;Shepherd,1988]。大脑的不同层具有不同功能。图 2.4 所示给出了大脑中各组织层的等级。传统的数字计算机也可看作是一个结构化的系统[Tanenbaum,1990],但这种组织与大脑的组织有根本的区别。所以,通过观察神经系统组织的层次,可以为设计与神经系统组织完全不同的计算机提供新观点。

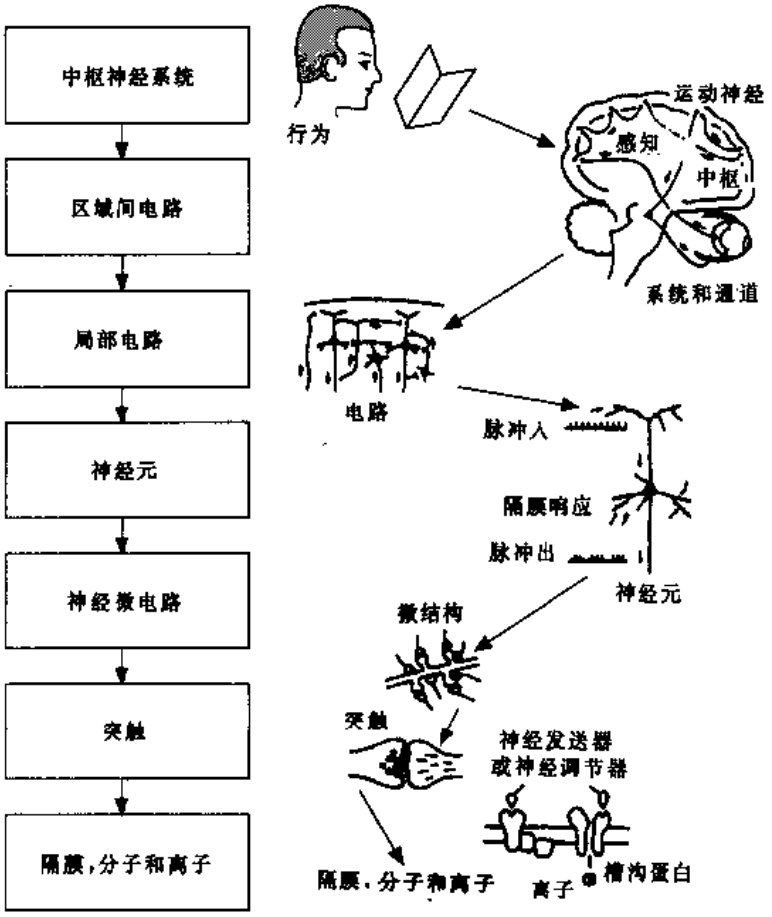


图 2.4 在生物神经系统中各层的结构组织

从最顶层来看,人的行为是由"整个大脑"来决定的。行为通过局部部位感觉图、系统和在其下面的区域间线路层的通道传递。局部部位感觉图包括位于大脑不同部分的多个区域,它们通过一定的组织对感觉信息作出反应。实际上,视觉系统、运动系统和听觉系统可作为一个适于这个范畴的整体。大脑结构的第三层称为局部环路,并且由具有相似或不同特性的神经元构成。这些神经元组合在一起负责局部处理。在下一层是神经元本身,大小约有 100 微米,包含若干树突子单元。在这层下面有神经微组织(类似于计算机中由三极管组合构成的硅芯片),它用于产生功能动作。这些微组织是影响突触周围区域的组织,有几个微米那么大,并且具有几毫秒的处理速度(比传统计算机中三极管的纳秒级处理速度慢得多)。神经元处理速

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

度慢的事实,也许从一定程度上说明毫秒级器件构成的大型生物计算机(即大脑)就需要大规模并行机制。接下来的一层由突触连接组成,在这里细胞从一个向另一个发送信号。在这层的下面,突触反过来依靠分子和离子的作用。

人脑和低级动物脑的一个最大不同,在于覆盖人脑的皮层在大小和复杂性方面比低级动物大得多。图 2.5 所示为覆盖大脑皮层的人脑的侧面示意图。

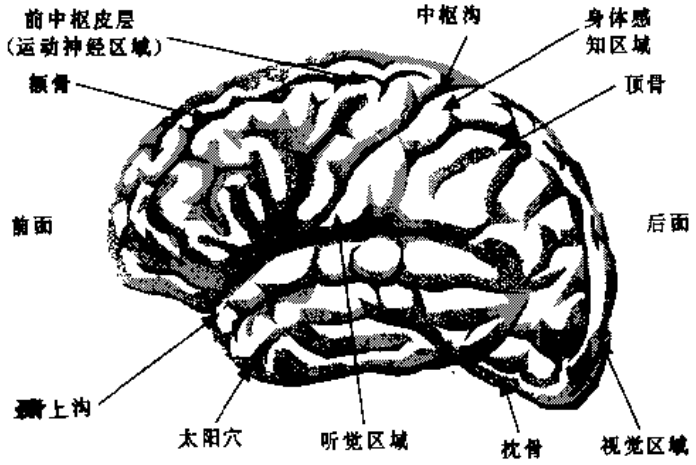


图 2.5 人脑和大脑皮层的不同区域

神经科学家已经根据特定的复杂任务,诸如看、听、说等,为大脑标识出不同的功能区域。视觉信息在大脑的后侧(在枕叶中)进行分析。听觉器官把信息送到听觉皮层(颞颥叶的上部),并在此对信息进行分析。视觉皮层实际上包含一个反映视网膜表面结构的映射。耳蜗是内耳的一部分,它能接收听觉输入和反映在内耳中的接收器表面的听觉皮层的映射。图 2.5 也示出了中央沟的前面,称为前中央皮层,它负责通过控制肌肉运动来组织运动神经的活动。顶叶(图 2.5 的中间区)负责处理来自皮肤和身体的信息。缔合皮层完成像消化、感觉等更高级的大脑功能。

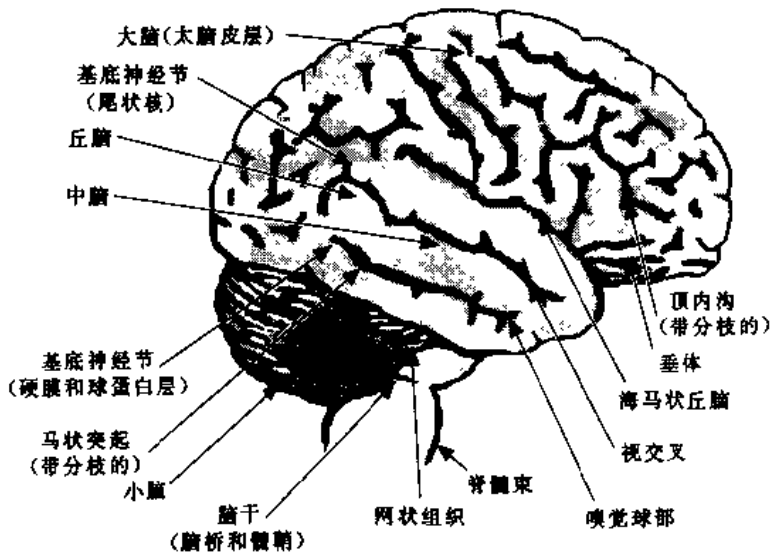


图 2.6 大脑内的主要神经结构

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

在大脑皮层的内部或下部主要的神经结构如图 2.6 所示。小脑在最底部,参与诸如走、跳、弹奏乐器等需要感觉运动肌协调的肌肉活动。小脑附近是脑干(脑桥和髓鞘),脑干参与呼吸、心搏动和胃肠功能。脊髓在脑的下部,传送信号给大脑,接收来自大脑的信号,并作出相应的反应。原始动物中的海马根据环境中的各种不同气味作出相应反应,但对人来说它(气味)却起着新的作用。

对视觉系统来说,视网膜是感觉器官,它作为转换器把光(刺激能量)转化成相应的神经信号,进一步由大脑中的视觉皮层处理(参考图 2.5)。视网膜为视觉系统感受刺激(见图 2.7(a))。视网膜分五层,感受细胞(由杆细胞和圆锥细胞组成)接收外部亮度信号,然后通过细胞的不同层把信号发送出去,并在这里做一些水平的预处理。最后,节细胞把信号送到主视觉皮层,以便主视觉皮层把本区域的视觉刺激进行编码。图 2.7(b)给出了源于视网膜、通过侧膝状核(LGN)到主视觉皮层的视觉传导通路。大脑在视觉皮层将视觉区映射成局部部位感觉

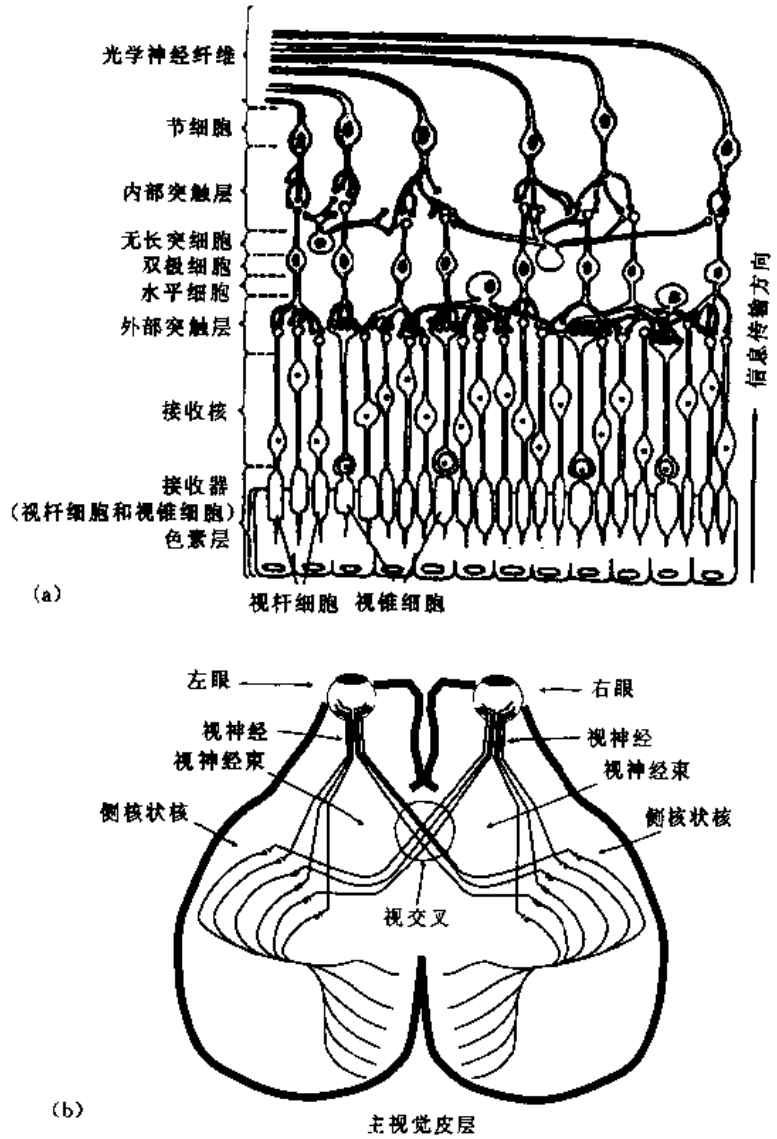


图 2.7 (a) 具有多层细胞的视网膜 (b)人的视觉系统图

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

图。Hubel 和 Weisel[1962,1965]所做的大量研究表明,在猫的视觉系统中各种各样的神经元有选择性地对边界、方向、运动、线长等等作出反应。

类似地,在听觉系统中,耳蜗(感觉器官)把声波(刺激)转变成神经信号,进一步由听觉皮层处理(见图 2.5)。图 2.8 所示给出了听觉系统从耳到听觉皮层的基本传导通路。

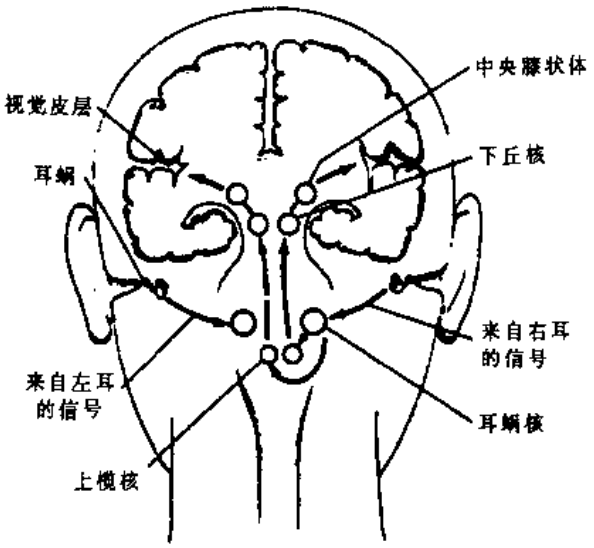


图 2.8 听觉传导通路图

实际上,从大脑解剖学可知大脑存在着多得令人惊异的数据。人们对神经系统中各种主要结构(组织)的位置和功能都很了解。然而,关于神经线路每部分作用的精确结论尚很缺乏。神经网络模型,很可能对有关大脑执行各种功能的机理和原理有一个更好的理解,起一定的作用。

例如,Suzuki,Kawato 和一些同事[Kawato 等人,1987;1988]已经研制出了一系列任意运动的模型。图 2.9 所示给出了由感觉刺激驱动的控制线路。它们的模型建立受到已知几个大脑区的局部解剖学和生理学的启示,包括从皮层到肌肉、脊髓、大脑和小脑(见图 2.6)。这些网络实际上以一种速度学习一个动作,然后又以不同的速度执行同样的动作。

怎样形成局部部位感觉图的问题,相当令人难以捉摸。Wilishow 和 Van de Malsberg [1979]讨论了一种关于这个过程的理论模型,它和连接主义学习技术具有密切的关系。Hinton 等人[1986]提出了一种称为粗编码的技术,它允许信息在细胞群中被精确地表示,其中每一个都有一个很大的感受区。实际上,实验观察表明,在神经系统中有很多细胞,即使在能十分精确确定的区域也具有很大的感受区。用粗编码来解释这一点是不会有矛盾的,但通常的观点则认为高精度相应地要求小感受区。Walters[1987]已经分析了一整套可能具有代表性的方案,并指出人们观察的神经元感受区是和一种非常有效的表现相适应的。有关在神经代码中发现的分布程度和这种代码的细节问题是十分有意义的,Anderson[1995]对此进行了探讨。

目前,关于大脑和对神经系统功能的理解的文献非常之多。MacGregor[1987]的著作中包括有详尽的论述和相应的参考文献。从我们的观点来看,下列范围能为计算机和信息科学提供有用的经验:

1. 大脑功能的信息处理范例。
2. 有关学习和记忆的数据表示细节。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

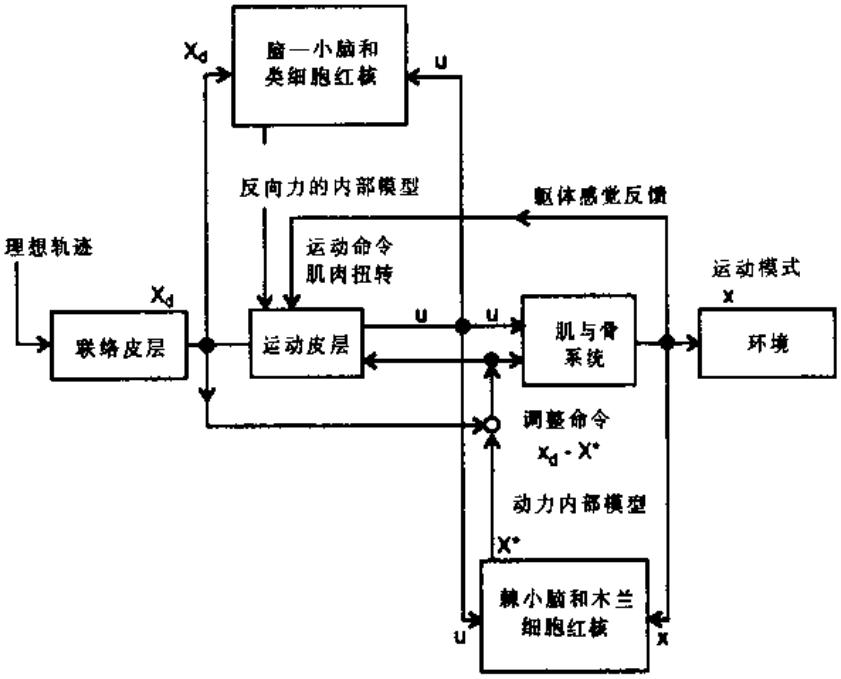


图 2.9 用于运动控制和学习的神经网络

在关于信息处理的例子中,系统的主要目的就是支持各种信号,这些信号反过来根据它们携带的信息表示出其特性(MacGregor[1987])。另一特性就是系统是按照信息传送的通路和信号所完成的动作来组织的。通过系统的通路可由工作流程图来表示。算法表示系统组成部件中信号所执行的动作。

一些研究人员已经提出了用于学习和记忆处理的典型的全局流程图。Kessner[1973]提出记忆能进一步分成提示存取、短期记忆和长期记忆存储以及可恢复系统。反过来,它们又由诸如匹配——失配、衰减、可选性留意、期望、练习、唤起、合并和读出过程等动作来控制。他接着提出了在神经系统中完成这些动作的主要位置。

在 Neelakanta 和 DeGroff[1994]的著作中,对数学的神经生物学概念及神经网络的信息理论方面给予了很好的讨论。

2.5 历 史 背 景

我们今天所知的神经网络开始于 McCulloch 和 Pitts[1943]的开拓性工作,而且从二十世纪四十年代初起各学科间就开始了对它的研究。McCulloch 是一位精神病医生和神经解剖学者,而 Pitts 是一位数学天才。他们关于全或无神经元的典型研究描述了神经网络的逻辑运算。图 2.10 所示为 McCulloch-Pitts 神经元模型,其输入为 $x_i, i=1,2,\cdots,N$。

W_i 表示第 i 个输入连接到神经元的权值(突触强度)。θ 是神经元的阈值,神经元的输入权值之和超过 θ,神经元的输出 O 为 1,否则为 0。如果连接(突触)是兴奋的则权值 W_i 为正,如果连接(突触)是抑制的则权值 W_i 为负。输入 x_i 是一位二进制数(0 或 1),它可以直接来自感官或来自其它神经元。神经元的工作方式(激活规则)可用下式表示:

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

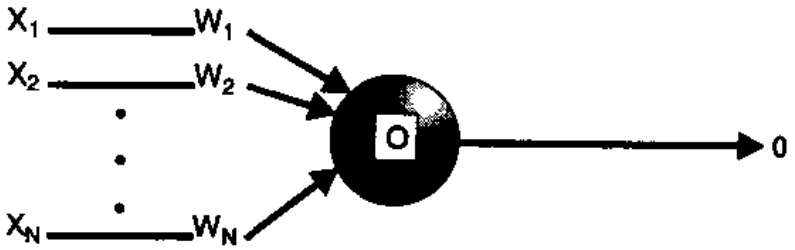


图 2.10 McCulloch-Pitts 神经元模型

$$O = g\left(\sum_{i=1}^{N} W_i x_i\right) \tag{2-1}$$

这里 $g(x)$是激活函数,定义如下:

$$g(x)=\begin{cases}1, \text{如果 } x \geqslant \theta \\ 0, \text{如果 } x < \theta\end{cases} \tag{2-2}$$

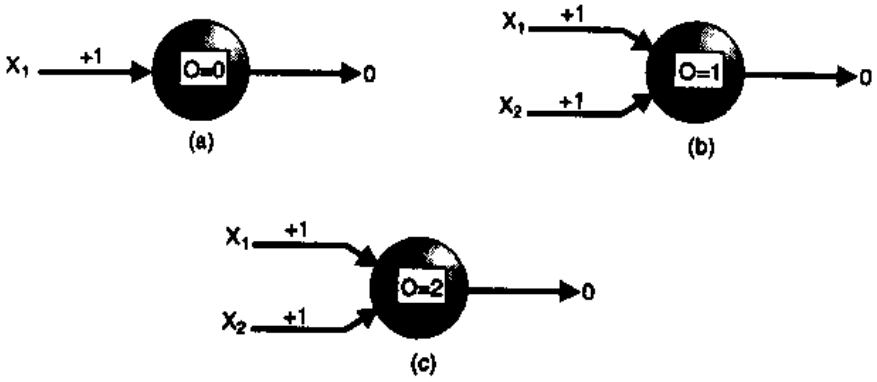


图 2.11 (a)非门的实现 (b)或门的实现 (c)与门的实现

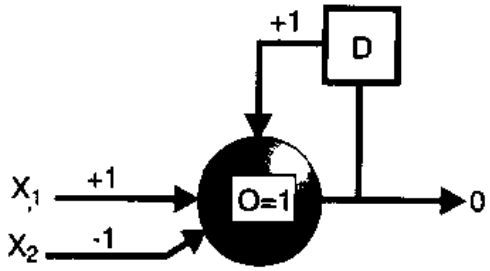


图 2.12 通过使用反馈和设定一个单元的时延实现记忆单元

这种简化模型即能显示出它实际的计算潜力,通过选择合适的权值,它可以执行诸如与、或、非等逻辑运算。图 2.11 给出了执行每一个这种运算的合适权值。我们已经知道,任一种多变量的组合功能都能用与非门或者或非门来实现。

如果我们设想在 M-T 神经元的输入和输出之间存在一个单元时延(如图 2.10 所示),我们用它(时延)即可建立一个时序逻辑电路。图 2.12 示出了能保持输入单记忆细胞的实现。

从图中可见,x_1 输入"1"使输出置位($O=1$),而 x_2 输入"1"则使输出复位($O=0$)。因为具有反馈环路,使在无输入时,也能保持输出。

基于神经元是二进制的事实(就像数字计算机中的开关一样),从而导致了计算机大脑模

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

拟的产生，这种理论称为控制论[Wiener，1948]。基于计算机和大脑的相似性，Wiener[1948]提出了有关控制、通信和信号处理等的概念，这刺激了人们研究控制论科学的兴趣。他讨论了学习系统中统计机制的重要性，但真正的统计机制和神经结构之间的联系是由 Hopfield 建立的。

Von Neumann 曾根据 M-T 神经模型，理想化地用门时延单元来构成 EDVAC 计算机[Aspray 和 Burks，1986]。实际上，他指出使用“大脑语言”去设计类似大脑处理机的研究是很有意义的[Von Neumann，1958]。

当心理学家 Hebb 提出了一种用于更新神经元之间突触强度的学习方案时，这种理论得到进一步发展。他提出当生物体学习不同的功能任务时，大脑内部的连接不断地发生变化。他也首先提出了：神经结构就是通过这种变化创造出来的。他的著名学习原理(我们现在称为 Hebbian 学习规则)表明：信息可以存贮在突触连接中，而且突触强度可通过一个神经元重复地被另一个经过突触的神经元激发而增大。引用 Hebb[1949]的论述如下：

当一个细胞 A 的轴突离另一个细胞 B 很近，甚至可以激发细胞 B 并且重复地或不断地激发它时，一些发育过程或新陈代谢的变化就会在一个或两个细胞中发生，以至于作为激发细胞 B 的细胞 A 的能量将提高。

这种学习规则称为 Hebb 规则或相关学习规则。这种规则对后来的学习计算模型和适应系统领域的研究产生了深远的影响。最初的 Hebb 规则并不包含对有选择性地削弱(或消除)突触强度的规定。Rochester 等人[1956]在数字计算机上进行了模拟，来测试 Hebb 关于神经元结构在大脑中的学习理论。他们的研究说明，在实际的神经元结构的工作理论中加入抑制是很必要的。

在 20 世纪 50 年代，心理学家 Frank Rosenblatt 通过实验提出了一种称作感知机的类似神经元的单元[Rosenblatt，1958]。他在著作中阐述[Rosenblatt，1958，p.387]：

总的来说，它体现出了智能系统的一些基本特性，没有深深地陷入特殊和不断未知之中，对特殊的生物组织保持一定的状态。

他摆脱了传统的符号逻辑思想的束缚，使用概率理论去分析这些模型。感知机是一种可训练的机器，它通过修改突触强度将模式进行分类。在早期的模式识别中，感知机结构的产生相当令人振奋。虽然感知机不常用，但它具有一定的历史价值。几十年来，在模式识别教科书中都有它的内容，所以我们只简要地对这种方法加以描述[Duda 和 Hart 1973；Tou 和 Gonzalez，1974]。

图 2.13 所示为一个具有感知单元(S 单元)、连接单元(A 单元)和反应单元(R 单元)的简单感知机。传感单元可以是模拟视网膜接受光刺激信号的光接收设备。

以全或无的方式产生反应的若干个 S 单元，通过固定的兴奋或抑制连接到在连接层中的每一个 A 单元。A 单元将连接修改后再连接到反应层的 R 单元。

具有单一 R 单元的感知机，即能完成对两个模式类的分类。而对多于两类的分类情况，则需要在反应层有多个 R 单元。Rosenblatt[1958，1962]提出了调整图 2.13 所示网络中的自由参数的算法。这种算法称为感知机收敛算法。该算法的收敛性证明指出：如果用于训练感知机的参数取自两个线性可分的类别，则感知机算法收敛，并且可以用一超平面对其进行划分。

另外，在这时期(20 世纪 60 年代初期)，Widrow 和 Hoff[1960，1962]提出了一种学习算法使网络输出方差和最小。基于这种有力的学习规则(也称为 Widrow-Hoff 学习规则)，他们设计了一种自适应线性组合器(ADALINE)。在 60 年代，ADALINE 及它的扩展形式

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

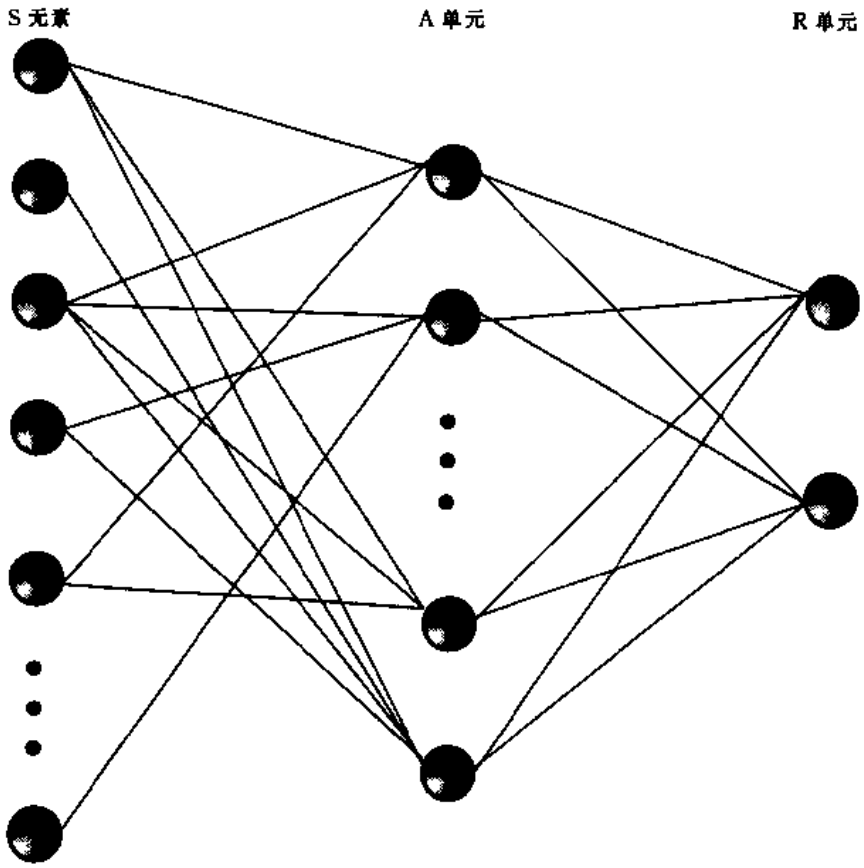


图 2.13 三层连接的简单感知机结构

MADALINE(多层自适应线性组合器),已被用于许多模式识别和自适应控制应用领域。在通信业中,它们可以作为长距离电话通信中用于回波抑制的自适应滤波器。

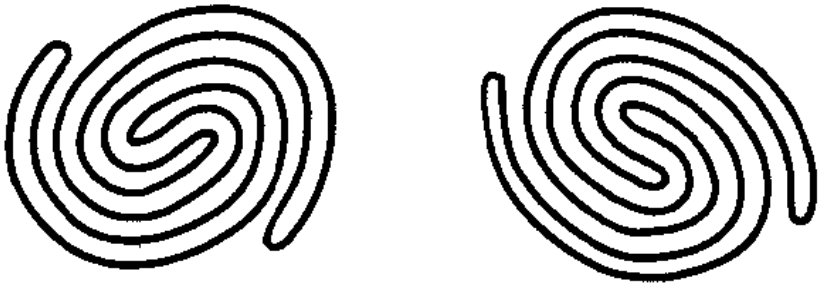


图 2.14 两种模式:一种是具有连续的弧,另一种具有不连续的弧

在 60 年代期间,感知机好像没有任何基本的限制,并且它们能做所有的事。Minsky 和 Pappert[1969]用巧妙的数学方法证明了单层感知机计算的局限性。他们的理论阐明:在多维空间中,单层感知机完成非线性模式的分离(像要求用于异或问题的分离,$f(0,0)=0$, $f(0,0)=1$, $f(1,1)=0$)。图 2.14 所示给出了两种模式,左边是连续的,不像右边那个包括不连续的弧。Minsky 和 Pappert 证明出,一个有限层的感知机不能把这些模式分到不同的类别中。

Minsky 和 Pappert 使用抽象的计算几何去研究的感知机,实际上是 Rosenblatt 研究的感知机的一个部分。与 Rosenblatt 提出的概率方法不同,他们通过在一个感知机结构中增加附加限制,采纳了预测算法的思想来分析感知机。他们证明了这种巧妙的理论用于他们抽象的感知机形式,以显示出单层感知机不能用于某种重要的几何分类。作者在他们的书中继续做

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

了一个假设(后来证明是完全不正确的):他们所发现的单层感知机的那些局限性可能也会对其它变形——更特殊的多层神经网络保持正确性。他们在书中的 13.2 章节[Minsky 和 Pappert,1969]写道:

尽管(甚至是因为)感知机具有很大的局限性,但它已经显示出它自己的研究价值,它有许多特性值得注意:它的线性;它的迷人的学习理论;作为一种并行计算它所具有的清晰的典型简化性。没有理由设想其中的任何一个优点会传到多层系统中。尽管如此,去阐明(或否认)我们通过直观判断所认为的向多层系统扩展将会是毫无结果的论断,我们认为仍是一个重要的研究问题。

结果,他们的理论被广泛地解释成怀疑所有类似感知机的装置作为学习机的效用。从而,由 Minsky 和 Pappert(他们后来在 MIT 参与建立了人工智能实验室)开始,形成了一股放弃神经网络研究的尝试,并且把研究基金转而投向人工智能领域的浪潮。正如 Cowan 后来所说,在那时由于缺乏计算能力以及重要的心理和经济原因(缺乏基金会对这项研究的支持),导致了对于类似感知机结构的研究工作的不景气局而。但是,Minsky 后来说,回想当时对感知机的怀疑有些反应过度[Rumelhart 和 McClelland,1986,pp.158－159]。

Nilsson [1965]在他的《学习机》一书中指出,多层感知机能用于分离像异或问题这样的非线性模式。但是感知机收敛理论只适用于单层感知机的学习。因此,那时多层感知机中的学习机制是不清楚的。Rosenblatt[1962,p.262]实际上也说过:

这里描述的过程被称为"反向传播误差修正过程",它是从 R 单元的误差中获得信息,将其反传回网络的感知端,以使网络的反应端迅速作出满意的修正。

这个问题的难点就是:为了更新连到隐层神经元的权值,如何确定网络中隐层神经元的误差。在 70 年代期间,提出了许多思想和概念来解决这个问题。例如,Paul Werbos[1974]在他哈佛大学的博士论文中,提出了用于多层感知机的一种新训练方法的数学框架。类似地,Harth[1976]和他的同伴提出了一种基于交叉相关的学习方法。这种学习方法无需明确地计算神经元输出中的差错,即能更新连到隐层神经元的权值[Pandya 和 Szabo,1991;Pandya 和 Venugopal,1994]。然而,直到 80 年代,这些基本问题才得以解决。

在 70 年代期间,少数几个研究人员开始研究神经网络,并且做了许多先期工作。这期间研制出了几种神经网络模型,这有助于对大脑完成各种功能的机制和原理有更深刻的理解。那时,在脑科学领域存在的令人难以理解、似是而非的论点之一,即是 Lashley[1929]认为大脑对事件的记忆是全分布的而不是局部的。但是,像 Mountcastle[1957]和 Hubel 以及 Wiesel [1962,1965]等神经科学家,已经发现大脑各区域对视觉和身体感觉(触觉)信息采用的是局部编码方式(具有良好组织的局部感觉图)。Erich Harth 及其同伴[Harth 等,1970;Anninos 等,1970]提出了一种具有随机连接的神经网络模型,从而解决了这一问题。他们通过对于随机网模型的计算机仿真,阐明了"随机性小但结构大"的原理。他们所提出的网状结构,实际上已经在大脑的视觉和触觉区得到了证明。他们将随机网络的各活动层的特性,归结为振荡、单稳、双稳等,并且研究了稳定性标准。Harth-Anninos 模型提出了一种更完善的理论,抓住了反馈连接型神经网络内部活动的重要特征。后来,Harth[1976]基于强调正反馈的"alopex"原理,提出了一种视觉感知模型。这种反馈使输入的指定特性增强。在这个模型中,"知觉"是通过空间派生和正反馈进行特性提取而产生的。

Amari[1972,1977]研制了一种阈值神经元的自适应模型,并且用它去研究随机网的动态行为。Nakano 在东京大学也提出了一种称为联想器的联想记忆模型,并且通过实际的机器人

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

实验证明了它的功能。芬兰的 Kohonen[1972,1977,1980]、Anderson[1972,1994,1973]等一直从事关于联想记忆的研究。日本的 Fukushima 提出了一种称为认知机[Fukushima,1975]的模型和它的变体模型(称为新认识)[Fukushima,1980],用于以大脑中的视觉通路知识为基础的视觉模式识别。

Grossberg 提出了许多神经网络结构和理论,包括神经元的自适应模型[Grossberg,1968]以及该模型在短期记忆方面的应用。然后,他在早期的工作中,研究了自上面下的模板匹配和由低到高的自适应滤波问题。这使通过学习反馈匹配和自适应共振进行模式识别成为可能。Carpenter 和 Grossberg[1987a,1987b]把基于这种现象的网络称为自适应共振理论(ART)。

20 世纪 80 年代是人工神经网络的复兴时期。这个时期发表了许多卓越的成果,有效地显示了人工神经网络的潜力。Hopfield 的论文提出了神经元的全连接网络,并研究了应用于联想记忆的潜力[Hopfield,1982,1984]。

1986 年,随着由 Rumelhart 和 McClelland[1986]编辑的两卷关于并行分布处理著作的出版,人工神经网络获得了又一次新生。该书介绍的概念和学习范例,对 Minsky 和 Pappert 过低估计多层感知机的潜力提出了批评。这部著作成功地消除了神经网络训练问题的障碍。在 60 年代,这一障碍实际上曾使神经网络的研究几乎完全停止。

2.6 人工神经网络

一般情况下,人工神经元网络是作为信息处理单元来模仿大脑,执行特定的任务或完成感兴趣的功能。Aleksander 和 Morton[1990]在广义上定义了一种神经网络,使得在研究领域中引入了实际大脑的神经网络,并为生物学的发现提供了空间。他们给出的定义如下:

神经计算是对具有自适应节点的网络的研究,通过对任务的学习过程,存储经验知识并使其可用。

学习算法是以一种有规律的方式用于突触权值的过程。广泛应用于各种领域的线性自适应滤波器理论[Haykin,1991;Widrow 和 Sterns,1985]使用的即是一种类似的方法。然而,模仿大脑(在这里细胞死了又再生)的神经网络也能具有可塑性(能修改自身拓扑结构的能力)。

下面是 Hecht-Nielsen[1990,p.2]给出的神经网络定义:

神经网络是一种并行的分布式信息处理结构,它通过称为连接的单向信号通路将一些处理单元(具有局部存储并能执行局部信息处理能力)互连而组成。每一个处理单元都有一个单输出到所期望的连接。每一个处理单元传通相同的信号——处理单元输出信号。处理单元的输出信号可以是任一种所要求的数学类型。在每一个处理单元中执行的信息处理在它必须完全是局部的限制下可以被任意定义,即它必须只依赖于处理单元所接受的输入激励信号的当前值和处理单元本身所存储记忆的值。

图 2.15 所示为一个具有突触连接和能执行非线性转换的简单处理单元的神经元一般模型。

一台典型计算机最基本的结构包括一个能执行各种指令的中央处理单元(CPU)。为了装载和存储信息,CPU 也访问一定的存储地址。串行计算机也包括一个存储单元,在这个单元中,数据和指令存储在各自不同的位置。在一个典型的计算过程中,CPU 取指令及执行该指令所要求的数据。然后它执行指令并在有结果时将结果存入合适的存储单元中。和传统的计算机相比,神经网络不包括独立的用于存储信息(即数据和指令)的存储空间,也没有一个单

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

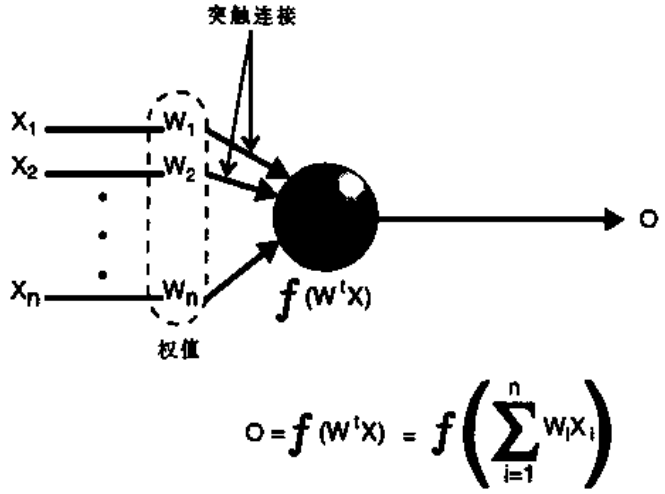


图 2.15 能提供连续值输出的一般神经元

一通用的能执行各种各样指令的 CPU。相反,神经网络是由许多单一的处理单元构成,在典型情况下,这些处理单元仅仅能够执行对输入求加权和。

与传统的计算机不同,神经网并不执行一系列指令,而只是对加予它的输入作出响应(反应)。神经网并不将结果存入指定的存储区中,而是在它达到一定的平衡状态之后,通过网络的全局状态来给出响应信息。

在传统计算机中,我们很容易从比如地址为 3541 的存储区存取信息,取出变量 X 的当前值。既然神经网按照处理器间的各种相互连接和每一个输入到处理单元的重要程度来存储信息,那么,信息更确切地说就是体系结构或网络结构的一种功能(函数),而不是网络中特定存储区的内容。

另一方面,这种使用地址(在存储区中的位置)来存取单元的方法,使传统的计算机很难从它内容的任一子集里查到某个单元的地址。而对于人工神经网络系统(ANNS)来说,由于不采纳通过地址来存取单元的这种基本方法,而是使用相互连接的简单单元进行并行计算,那么就很容易访问到要存取内容的存储区。即可以用一个单元的部分内容去获得余下的内容。

神经网络在一些文献中也称为人工神经系统,连接性网络,连接主义学说,神经计算机,并行分布处理器(系统),分层自适应系统,自组织网络,网络计算,神经吗啡系统等等。

换句话说,人工神经系统是能获取、存储和利用经验知识的物理细胞系统。神经网络的如下特性在其广泛的应用中起着重要的作用:

1. 自适应性——强有力的学习算法和自组织规则使它能在不断变化的环境中对每一要求进行自适应。

2. 非线性处理——具有执行非线性任务和去除噪音的能力,使它能够很好地用于分类和预测问题。

3. 并行处理——大量广泛互连的处理单元组成的结构,提供了并行处理和并行分布信息存储的能力。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

参考书与文献

Albus, J., *Brains, Behavior and Robotics*, McGraw-Hill, Peterborough, NH, 1981.

Aleksander, I. and Morton, H., *An Introduction to Neural Computing*, Chapman & Hall. London, 1990.

Amari, S. I., "Learning patterns and pattern sequences by self-organizing nets," *IEEE Trans. Comput.*, vol. 21, pp. 1197–1206, 1972.

Amari, S. I., "Neural theory of association and concept formation," *Biol. Cybern.*, vol. 26, pp. 175–185, 1977.

Anderson, J. A., "A simple neural network generating an interactive memory," *Math. Biosci.*, vol. 14, pp. 197–220, 1972.

Anderson, J. A. and Bower, G. H., *Human Associative Memory*, V. H. Vincent, Washington, D.C., 1973.

Anderson, J. A., Introduction to Practical Neural Modeling, MIT Press, Cambridge, MA, 1994.

Anninos, P. A., Beek, B., Csermely, T. J., Harth, E. M., and Pertile, G., "Dynamics of neural structures," *J. Theor. Biol.*, vol. 26, pp. 121–148, 1970.

Arbib, M. A., *Brains, Machines, and Mathematics*, 2nd ed., Springer-Verlag, Berlin, 1987.

Aspray, W. and Burks, A., Papers on John von Neumann on Computing and Computer Theory, Charles Babbage Institute Reprint Series for *History of Computing*, vol. 12, MIT Press, Cambridge, MA, 1986.

Byrne, J. H. and Schultz, S. G., *An Introduction to Membrane Transport and Bioelectricity*, Raven Press, New York, 1988.

Carlson, N. R., *Physiology of Behavior*, Allyn & Bacon, Boston, 1977.

Carpenter, G. A. and Grossberg, S. "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Comput. Vision, Graphics Image Process.*, vol. 37, pp. 54–115, 1987a.

Carpenter, G. A. and Grossberg, S. "ART 2: Self-organization of stable category recognition codes for analog input patterns," *Appl. Optics*, vol. 26, pp. 4919–4930, 1987b.

Churchland, P. S. and Sejnowski, T. J., *The Computational Brain*, MIT Press, Cambridge, MA, 1992.

Churchland, P. S., *Neurophilosophy: Toward a Unified Science of the Mind/Brain*, MIT Press, Cambridge, MA, 1986.

Cowan, J. D., "Neural Networks: The early days," in *Advances in Neural Information Processing Systems 2*, D. S. Touretzky (Ed.), Morgan Kaufmann, San Mateo, CA, pp. 828–848, 1990.

Dayhoff, J., *Neural Network Architectures: An Introduction*, Van Nostrand Reinhold, New York, 1990.

Duda, R. O. and Hart, P. E., *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, 1973.

Fukushima, K., "Cognition: a self-organizing multilayered neural network," *Biol. Cybern.*, vol. 20, pp. 121–136, 1975.

Fukushima, K., "Neocognition: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.*, vol. 36, pp. 193–202, 1980.

Grossberg, S., "Nonlinear difference-differential equations in prediction and learning theory," *Proc. Natl. Acad. Sci.*, vol. 58, pp. 1329–1334, 1967.

Grossberg, S., "A prediction theory for some nonlinear functional-difference equations," *J. Math. Anal. Appl.*, vol. 22, pp. 643–694, 1968.

Harth, E., "Visual perception: a dynamic theory," *Biol. Cybern.*, vol. 22, pp. 169–180, 1976.

Harth, E. M., Csermely, T. J., Beek, B., and Lindsay, R. D., "Brain functions and neural dynamics," *J. Theor. Biol.*, vol. 26, pp. 93–120, 1970.

Haykin, S., *Adaptive Filter Theory*, 2nd ed , Prentice-Hall, Englewood Cliffs, NJ, 1991.

Haykin, S., *Neural Networks: A Comprehensive Foundation*, Macmillan College Publishing, New York, 1994.

Hebb, D. O., *The Organization of Behavior: A Neuropsychological Theory*, John Wiley & Sons, New York, 1949.

Hetch-Nielsen, R., *Neurocomputing*, Addison-Wesley, Reading, MA, 1990.

Hopfield, J. J., "Neural networks and physical systems with emergent collective computational abilities," *Proc. Natl. Acad. of Sci.*, vol. 79, pp. 2554–2558, 1982.

Hopfield, J. J., "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Natl. Acad. of Sci.*, vol. 81, pp. 3088–3092, 1984.

Hubel, D. H. and Wiesel, T. N., "Receptive fields, binocular interaction and functional architecture in cat's visual cortex," *J. Physiol.*, vol. 160, pp. 106–154, 1962.

Hubel, D. H. and Wiesel, T. N., "Receptive fields and functional architecture in two non-striate visual areas (18 and 19) of the cat," *J. Neurophysiol.*, vol. 28, pp. 229–298, 1965.

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xidian.edu.cn

Kandel, E. R. and Schwartz, J. H., *Principles of Neural Science*, 3rd ed., Elsevier, New York, 1991.
Katz, B., *Nerve, Muscle and Synapse*, McGraw-Hill, New York, 1966.
Kawato, M., Furukawa, K., and Suzuki, R., "A hierarchical neural-network model for control and learning of voluntary movement," *Biol. Cybern.*, vol. 57, pp. 169–185, 1987.
Kawato, M., Isobe, M., Maeda, Y., and Suzuki, R., "Coordinates transformation and learning control for visually-guided voluntary movement with iteration: a Newton-like method in function space," *Biol. Cybern.*, vol. 59, pp. 161–177, 1988.
Kessner, R., "A neural system analysis of memory storage and retrieval," *Psychol. Bull.*, vol. 80, pp. 177–203, 1973.
Koch, C. and Segev, I., *Methods in Neuronal Modeling: From Synapses to Networks*, MIT Press, Cambridge, MA, 1989.
Kohonen, T., "Correlation matrix memories," *IEEE Trans. Comput.*, vol. C-21, no. 4, pp. 353–359, 1972.
Kohonen, T., *Associative Memory: A System-Theoretical Approach*, Springer-Verlag, Berlin, 1977.
Kohonen, T., *Content-Addressable Memories*, Springer-Verlag, Berlin, 1980.
Kung, S. Y., *Digital Neural Networks*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
Lashley, K., *Brain Mechanisms and Intelligence*, University of Chicago Press, Chicago, 1929.
Levine, D. S., *Introduction to Neural and Cognitive Modelling*, Lawrence Erlbaum, Hillside, NJ, 1991.
Levine, M., *Man and Machine Vision*, McGraw-Hill, New York, 1985.
MacGregor, R. J., *Neural and Brain Modeling*, Academic Press, London, 1987.
Marr, D., *Vision*, W. H. Freeman, New York, 1982.
McCulloch, W. S. and Pitts, W., "A logical calculus of the ideas immanent in nervous activity," *Bull Math. Biophys.*, vol. 5, pp. 115–133, 1943.
Mead, C. A., *Analog VLSI and Neural Systems*, Addison-Wesley, Reading, MA, 1989.
Minsky, M. L. and Pappert, S. A., *Perceptrons*, MIT Press, Cambridge, MA, 1969.
Mountcastle, V. B., "Modality and topographic properties of single neurons of cat's somatic sensory cortex," *J. Neurophysiol.*, vol. 20, pp. 408–434, 1957.
Nakano, K., "Associatron — A model of associative memory," *IEEE Trans. Syst. Man and Cybern.*, Vol. SMC-2, pp. 380–388, 1972.
Nauta, W. J. H. and Feirtag, M., *Fundamental Neuroanatomy*, W. H. Freeman, New York, 1986.
Neelakanta, P. S. and DeGroff, D. F., *Neural Network Modeling: Statistical Mechanics and Cybernetic Perspectives*, CRC Press, Boca Raton, FL, 1994.
Nilsson, N. J., *Learning Machines: Foundations of Trainable Pattern-Classifying Systems*, McGraw-Hill, New York, 1965.
Pandya, A. S. and Szabo, R., "A fast learning algorithm for neural network applications," *Proc. of IEEE Conf. Syst. Man Cybern.*, pp. 1569–1573, 1991.
Pandya, A. S. and Venugopal, K. P., "A stochastic parallel algorithm for supervised learning in neural networks," *IEICE Trans. Inf. Syst.*, vol. E77-D, no 4, pp. 376–384, 1994.
Rámon Y. Cajál, "Les preuves objectives de l'unit'e anatomique des cellules nurveuses," *Trob. Lab. Inest. Biol. Univ. Madrid*, vol. 29, pp. 1–37, 1934. (translation: Purkiss, M. V. and Fox, C. A., Madrid: Instituto "Ramon y Cajal", 1954).
Rochester, N., Holland, J. H., Haibt, L. H., and Duda, W. L., "Tests on a cell assembly theory of the action of the brain, using a large digital computer," *IRE Trans. Inf. Theory*, vol. IT-2, pp. 80–93, 1956.
Rosenblatt, F., "The Perceptron: a probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, pp. 386–408, 1958.
Rosenblatt, F., *Principles of Neurodynamics*, Spartan Books, Washington, D.C., 1962.
Rumelhart, D. E. and McClelland, J. L. (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1, MIT Press, Cambridge, MA, 1986.
Shepherd, G. M., *Neurobiology*, Oxford University Press, New York, 1983; 2nd ed., 1988.
Shepherd, G. M., *The Synaptic Organization of the Brain*, Oxford University Press, New York, 1990a.
Shepherd, G. M., The significance of real neuron architectures for neural network simulations, in *Computational Neuroscience*, E. L. Schwartz (Ed.), MIT Press, Cambridge, MA, 1990b.
Shephord, G. M. and Koch, C., Introduction to Synaptic Circuits, in *The Synaptic Organization of the Brain*, G. M. Shepherd (Ed.), Oxford University Press, New York, 1990.
Sherrington, C. S. *The Brain and Its Mechanism*, Cambridge University Press, New London, 1933.
Suga, N., "Cortical computational maps for auditory imaging," *Neural Networks*, vol. 3, pp. 3–21, 1990a.
Suga, N., Computations of Velocity and Range in the Bat Auditory System for Echo Location, in *Computational Neuroscience*, E. L. Schwartz (Ed.), MIT Press, Cambridge, MA, 1990b.
Simmons, J. A., Saillant, P. A., and Dear, S. P., "Through a bat's ear," *IEEE Spectrum*, vol. 29, no. 3, pp. 46–48, 1992.

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

Tanenbaum, A. S., *Structured Computer Architecture,* 3rd ed., Prentice-Hall, Englewood Cliffs, NJ, 1990.

Tou, J. T. and Gonzalez, R. C., *Pattern Recognition Principles,* Addison-Wesley, Reading, MA, 1974.

Thompson, R. F., *Foundations of Physiological Psychology,* Harper & Row, New York, 1967.

Truex, R. C. and Carpenter, M. B., *Human Neuroanatomy,* Williams & Wilkins, Baltimore, 1969.

von Neumann, J., *The Computer and the Brain,* Yale University Press, New Haven, CT, 1958.

Werbos, P. J., Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences, Unpublished Ph.D. dissertation, Harvard University, 1974.

Widrow, B., Generalization and Information Storage in Networks of ADALINE 'Neurons', in *Self-Organizing Systems,* M. C. Yowitz et al. (Eds.), Spartan Books, Washington, D.C., pp. 435–461, 1962.

Widrow, B. and Hoff, M. E., "Adaptive Switching Circuits," IRE WESCON Convention Record, pp. 96–104, 1960.

Widrow, B. and Sterns, S. D., *Adaptive Signal Processing,* Prentice-Hall, Englewood Cliffs, NJ, 1985.

Wiener, N., *Cybernetics: Or, Control and Communication in the Animal and the Machine,* John Wiley & Sons, New York, 1948.

Zurada, J. M., *Introduction to Artificial Neural Systems,* West Publ., New York, 1992.

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

第三章 预 处 理

3.1 概 述

本章我们将讨论图像数据的预处理。图像压缩、骨架处理和边缘检测，都属于整个识别系统的预处理阶段。它们与模式识别的关系在于：它们能够将原始的输入数据进行处理，使之适合于神经网络的识别。

首先我们对从扫描仪得到的图像数据进行简单的讨论。

3.2 扫描图像的处理

通过扫描仪得到的图像，能够存成许多不同的格式。其中最常用的格式是位图格式(BMP)。这里我们简单地描述怎样将 BMP 文件转换成象素文件，而后面所要讨论的识别技术所用的就是象素文件。图 3.1 给出了一个扫描图像的例子，它的每一个象素是用 8 比特来表示的，或者换句话来说它是 256 级灰度的图像。图 3.2 给出了另一个扫描图像的例子，它的每一个象素是用 1 比特来表示的，或者说它是单色图像。

图 3.1 灰度图像

图 3.2 单色图像

BMP 文件通常包括三部分。首先，文件头含有它后面内容的一些必要信息，这些信息包括象素图的宽和高，每个象素的位数以及象素数据的起始位置。BMP 文件头的结构示于清单 3.1。第二，文件头后面就是调色板，它一般由红、蓝、绿(RGB)的浓度所构成，红、蓝、绿的合成就定义了实际的颜色。因为象素值就是调色板的索引，所以调色板的大小由每个象素的位数所决定。对于灰度图像，通常 RGB 的值相同并且表示明暗度。

清单 3.1

```
struct sHeaderBMP{
    char      id1,id2;          // "BM" identifies bitmap
    long      FileSize          // size of the file
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
    int         reserved[2];                // normally 0
    long        HeaderSize;                 // offset to pixel data
                                            // in row X column format
    long        InfoSize;                   // normally should be 0x28
    long        Width;                      // # of columns`
    long        Depth;                      // # of rows
    int         BitPlanes;                  // # of bit planes
    int         BitsPerPel;                 // number of bits per pixel
    long        Compression;                // normally 0, not compressed
    long        ImageSize;                  // size of the image.
    long        PelsPerMeterX;              // Resolution in X direction
    long        PelsPerMeterY;              // Resolution in Y direction
    long        ColorsUsed;                 // # of colors used in image
    long        ImportantColors;            // # of colors that are important
}; //Palette (if any follows this)
```

图像文件格式的详细论述已超出本书的范围;然而本书的配套磁盘提供了这样的一个程序,它能对 BMP 文件进行有限制的读和写。此外,我们也能找到许多这方面的书,这些书对于经常使用的图形格式进行了较为详尽的讨论。其中 Rimmer[1992]和 Levine[1994]所著的两本书,对位图(BMP)文件进行了非常详尽的描述,并且对其它的一些常用图像文件格式也进行了深入的讨论。

3.3 图 像 压 缩

对于识别器和接下来的预处理阶段来说,将输入图像压缩到可管理的水平上是非常有用的,并且是非常必要的。所需的压缩量随应用的不同而不同。在字符识别的场合下,实验已经证明 16×16 的表示就已经足够用来保持输入图像的形状[Darwiche 等,1992]。有许多进行图像压缩的方法。例如,CCITT4 行程编码算法以及 LZ77 滑动窗口压缩算法。可以在 Nelson [1992]所著的书中,找到这些技术的详尽和有益的讨论。

下面的这个方法[Darwiche 等,1992]使用了行程编码算法(RLC)。首先进行水平压缩,然后进行垂直压缩。假定图像表示成一个二值象素的矩阵。在后面的边缘检测的讨论中,我们将详细讨论灰度图像。行程编码算法是通过对连续的 1 和 0 进行记数来创建一个表(List)。例如,序列 00001110011 将被编码成 4,3,2,2。对于水平压缩从每一行进行,对于垂直压缩从每一列进行。这样压缩比 C 就必须从水平和垂直方向上分别进行定义。

在水平方向上:

$$C = C_{水平} = \frac{N_{水平}}{M_{水平}} = \frac{\text{输入数据的列数}}{\text{所需输出的列数}} \tag{3-1}$$

在垂直方向上:

$$C = C_{垂直} = \frac{N_{垂直}}{M_{垂直}} = \frac{\text{输入数据的行数}}{\text{所需输出的行数}} \tag{3-2}$$

原始图像就可以采用这种改进的 RLC 算法进行重建。但是在使用这种算法时必须注意到一些问题。尤其是,在计算(行程)/C 时,舍入误差会引起在压缩方向上出现不相等的象素数。这个问题可通过积累舍入误差来简单地克服,下面的 3.3.1 节的程序代码具体描述了克服这种问题的方法。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

3.3.1 图像压缩的例子

下表给出了行程压缩类的类定义。它的数据成员有上节描述的一行程矩阵和一给出最大索引 x 的索引。

清单 3.2

```
class RLCX {
  private:
    unsigned int RLmatrix[RLCXMAXX][RLCXMAXY];
     int        LargestX;
  public:
     RLCX();
    void Setup(unsigned char *data, int width, int height );
    void Clear();
    void Compress(int width, int height, double Scale);
    unsigned int  Query(int x, int y){return RLmatrix[x][y];}
    unsigned int  QueryIndx(){return LargestX;}
   };
```

表中最引人注意的是“Compress”。因为正是在这里进行压缩或进行比例调整,此方法示于下面清单 3.3。注意到对输入图像乘以比例因子的结果是:首先如果结果不为整数时,它被舍入到与它最接近的整数,接着当累积误差“足够大”时误差就加到下一个数上。本例中“足够大”被定义成超过 1。尽管清单 3.2 中“Setup”不太重要,但它还是被列于清单 3.3 中并且在下文也对它进行了讨论。

下面图 3.3(a)－(e)描述了对字符“A”进行压缩的过程。

清单 3.3

```
void RLCX::Compress(int width, int height, double Scale){
  int i,j,bpl,k;
  double erf;
  double  r,frac,ipart;
  bpl=width>>3;
  for (j=0; j<height; j++) {
     erf=0;
    while ((RLmatrix[i][j] & 0x8000) != 0) {
      r=(double)(RLmatrix[i][j] & 0x7fff);
       r=Scale*r;
       frac=modf(r,&ipart);
       if (frac>0.5) {
         k=(int)ipart+1;
         /* erf = erf- (1.0-frac); */
       } else {
         k=(int)ipart;
         /* erf += frac; */
       } /* endif */
       erf = erf + (r-(double)k);
       if (erf >= 1.0) {
         k=k+1;
         erf=erf-1.0;
       } /* endif */
       if (erf <= -1.0) {
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
            k=k-1;
            erf=erf+1.0;
          } /* endif */
/*        RLmatrix[i][j] = (2*(RLmatrix[i][j] & 0x7fff))/3; *
          RLmatrix[i][j] = k;
          RLmatrix[i][j] = RLmatrix[i][j] | 0x8000;
          i++;
        } /* endwhile */
      } /* endfor */
}

void RLCX::Setup(unsigned char *data, int width, int height ) {
  unsigned int Target,indx,Count,Query,i,j,x,Bit;
  unsigned char mask;

  Clear();
  x=0;
  for (j=0; j<height; j++) {
    Target=1;
    i=0;
    indx=0;
    Count=0;
    while (i<width) {
      //x=j*(width>>3)+ (i>>3);
      Bit=i&0x07;
      mask=0x80>>Bit;
      if ((data[x] & mask)!=0) Query=1;
                  else Query=0;
      if (Query==Target) {
        Count++;
        i++;
        if ((i&0x07)==0) x++;
      } else {
       RLmatrix[indx][j]=Count | 0x8000;
       if(indx>LargestX)LargestX=indx;
        indx++;
        if (Target==0) Target=1;
              else Target=0;
        Count=0;
      } /* endif */
    } /* endwhile */
    RLmatrix[indx][j]=Count | 0x8000;
  } /* endfor */
}
```

3.4 边 缘 检 测

本节中,我们将对灰度图像进行分割,以便于来提取点、线和更重要的边缘。分析灰度图像时,这常常是非常重要的第一步。上面所述的三种情况,都是检测出数字图像中的不连续部分。我们首先考虑一模板,此模板定义了象素 Z_5 的 8 邻域,如下:

$$\begin{bmatrix} Z_1 & Z_2 & Z_3 \\ Z_4 & Z_5 & Z_6 \\ Z_7 & Z_8 & Z_9 \end{bmatrix} \tag{3-3}$$

这里 $Z_i(i=1,2,\cdots,9)$表示象素的灰度值。

将不同的加权矩阵对图像进行加权,可容易地得出像点、线等不连续部分的这类特征,加权矩阵如下:

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
00000000000011111100000000000000000
00000000001111111111000000000000000
00000000111111111111000000000000000
00000001111111001111110000000000000
00000011111000000111111000000000000
00000111100000000011111000000000000
00001111000000000001111100000000000
00011111000000000000111111000000000
00111110000000000000011111100000000
01111110000000000000001111110000000
01111110000000000000011111110000000
01111111111111111111111111111110000
11111111111111111111111111111111110
11111111111111111111111111111111110
11111011111111111111111111111111100
11111000011111111111111111111110000
11111000000000000101110111111100000
11111000000000000000000011111100000
11111000000000000000000111111100000
11111000000000000000001111111000000
11111100000000000000001111110000000
11111100000000000000011111110000000
11111100000000000000111111110000000
11111100000000000000111111111000000
11111100000000000000111111111000000
11111100000000000000011111111000000
11111000000000000000001111111000000
01111000000000000000000111111000000
01110000000000000000000000000110
```

12, 6, 18
10, 10, 16
9, 12, 15
8, 6, 2, 6, 14
7, 4, 6, 5, 14
6, 3, 9, 5, 13
5, 4, 9, 6, 12
4, 5, 10, 5, 12
3, 4, 11, 6, 11
3, 4, 13, 5, 11
2, 5, 14, 6, 9
1, 6, 14, 6, 9
1, 6, 14, 7, 8
1, 31, 4
0, 35, 1
0, 35, 1
0, 6, 1, 27, 2
0, 5, 4, 23, 4
0, 5, 12, 1, 1, 3, 1, 8, 5
0, 5, 19, 7, 5
0, 5, 20, 6, 5
0, 5, 21, 6, 4
0, 6, 20, 6, 4
0, 5, 22, 5, 4
0, 6, 21, 5, 4
0, 6, 21, 6, 3
0, 6, 21, 6, 3
0, 6, 22, 5, 3
0, 5, 23, 5, 3
1, 4, 24, 4, 3
1, 3, 26, 2, 4

图 3.3(a) 输入字符　　图 3.3(b) 行程编码后的每行

$$\begin{pmatrix} W_1 & W_2 & W_3 \\ W_4 & W_5 & W_6 \\ W_7 & W_8 & W_9 \end{pmatrix} \tag{3-4}$$

加权矩阵作用到图像上的任意一点(以 Z_5 为中心的 8 邻域)的结果可如下计算：

$$R = \sum_{i=1}^{9} W_i Z_i \tag{3-5}$$

很明显，我们用下面这样一个加权矩阵就可检测出孤立点：

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \tag{3-6}$$

当加权后结果超过一预定阈值 T，即下式(3-7)成立时，这个孤立点就被检测出来：

$$|R| \geqslant T \tag{3-7}$$

同样，以类似的方式可用下面这些模板来检测不同方向的线段：

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

5, 3, 8	0000011100000000	0000111110000000
4, 5, 7	0000111110000000	0001111110000000
4, 5, 7	0000111110000000	0001000011000000
3, 3, 1, 3, 6	0001110111000000	0011000011100000
3, 2, 3, 2, 6	0001100011000000	0110000001100000
3, 1, 4, 2, 6	0001000011000000	0110000000110000
2, 2, 4, 3, 5	0011000011000000	1111101011110000
2, 2, 5, 2, 5	0011000001100000	1111111111111111
1, 2, 5, 3, 5	0110000011100000	1111111111111110
1, 2, 6, 3, 4	0110000001110000	1100000101111100
1, 2, 6, 3, 4	0110000001110000	1100000000011100
0, 3, 6, 3, 4	1110000001110000	1110000000011100
0, 3, 6, 3, 4	1110000001110000	1110000000001100
0, 14, 2	1111111111111100	1110000000001110
0, 16	1111111111111111	1110000000001110
0, 16	1111111111111111	1000000000000110
0, 15, 1	1111111111111110	
0, 2, 2, 10, 2	1100111111111100	
0, 2, 5, 1, 1, 5, 2	1100000101111100	
0, 2, 9, 3, 2	1100000000011100	
0, 2, 9, 3, 2	1100000000011100	
0, 2, 9, 3, 2	1100000000011100	
0, 3, 9, 2, 2	1110000000001100	
0, 2, 10, 2, 2	1100000000001100	
0, 3, 9, 2, 2	1110000000001100	
0, 3, 9, 3, 1	1110000000001110	
0, 3, 9, 3, 1	1110000000001110	
0, 3, 10, 2, 1	1110000000001100	
0, 2, 10, 2, 2	1100000000001100	
0, 2, 11, 2, 1	1100000000000110	
0, 1, 12, 1, 2	1000000000000100	

图 3.3(c) 除以水平压缩比 $C_{水平}$ 的每行

图 3.3(d) 水平压缩后的字符

图 3.3(e) 完整压缩后的字符

$$\begin{matrix}\begin{bmatrix}-1&-1&-1\\2&2&2\\-1&-1&-1\end{bmatrix}&\begin{bmatrix}-1&-1&2\\-1&2&-1\\2&-1&-1\end{bmatrix}&\begin{bmatrix}-1&2&-1\\-1&2&-1\\-1&2&-1\end{bmatrix}&\begin{bmatrix}2&-1&-1\\-1&2&-1\\-1&-1&2\end{bmatrix}\\ 水平 & 45^\circ & 垂直 & -45^\circ\end{matrix} \tag{3-8}$$

现在继续讨论更为有趣且更为重要的问题,即边缘检测。边缘就是灰度值有明显区别的两区域间的边界。也就是说,边缘检测就是寻找灰度发生急剧变化的区域。既然考虑到灰度的变化速率,自然地就想到了空间微分。图 3.4 给出了具有相同灰度值区域边缘的一阶和二阶微分。

注意到灰度值从低到高(暗——亮)变化处的一价微分为正,而从高到低变化处为负,其它处为零。因此,一价微分的量值本身就是一个边缘检测器。二价微分有一个特性:灰度值变化的中间点过零。本节中,将依据二价微分的这个特性更详细地讨论一下边缘检测。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

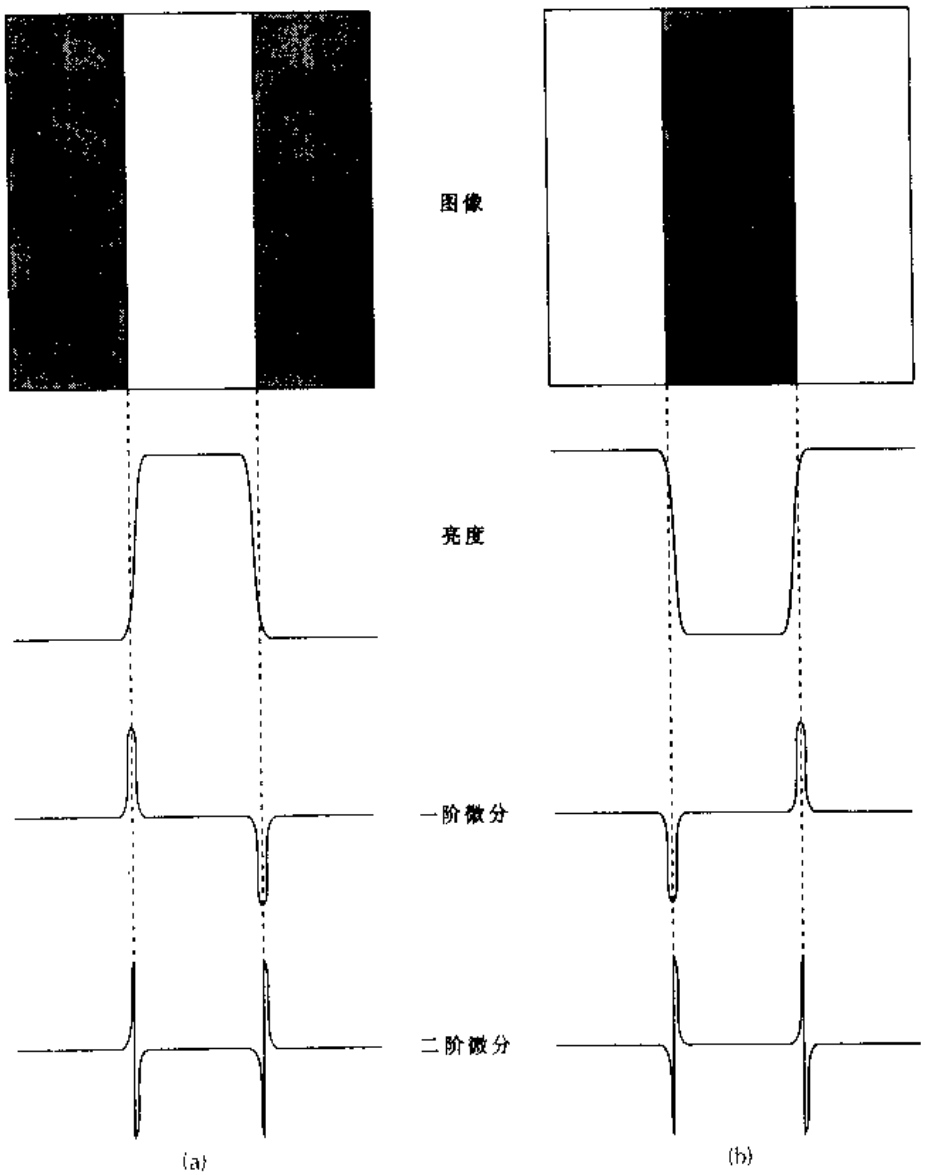


图 3.4 一阶和二阶微分进行边缘检测(摘自 Gonzalez 和 Woods[1992])

现在将讨论归结到更为有意义的二维图像上。在这种情况下,我们同样基于上述的方法进行操作。但现在用梯度来代替一价微分,并且用拉普拉斯算子来代替二价微分。函数 f(x,y)的梯度定义如下:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix} \tag{3-9}$$

梯度的大小用下式定义:

$$\nabla f = |\nabla f| = (G_x^2 + G_y^2)^{1/2} \tag{3-10}$$

梯度矢量的方向由下式给出:

$$\alpha(x,y) = \tan^{-1}\left(\frac{G_x}{G_y}\right) \tag{3-11}$$

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

为了能应用这些运算符，函数 f(x,y)必须是一个能进行微分的闭合形式的解析表达式。显然，对于代表图像密度(浓度)常量数据的数列，现在一定能近似求解偏微分$\partial/\partial x$ 和$\partial/\partial y$。如图 3.4 求解偏微分的最明显并且最直接的方式如下：

$$G_x = Z_5 - Z_8$$
$$G_y = Z_5 - Z_6 \tag{3-12}$$

上述方法的一个变形化就是用 3×3 的加权阵对图像进行加权，如下：

$$G_x = Z_7 + Z_8 + Z_9 - (Z_1 + Z_2 + Z_3)$$
$$G_y = Z_3 + Z_6 + Z_9 - (Z_1 + Z_4 + Z_7) \tag{3-13}$$

相应的加权矩阵(被称为 Prewitt 算子)如下：

$$\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \& \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \tag{3-14}$$

另一种被用于近似求解偏微分的方法，被称为 Sobel 算子，此算子具有平滑效果。对应于这种方法的偏微分的离散算子，如下：

$$G_x = Z_7 + 2Z_8 + Z_9 - (Z_1 + 2Z_2 - Z_3)$$
$$G_y = Z_3 + 2Z_6 + Z_9 - (Z_1 + 2Z_4 + Z_7) \tag{3-15}$$

相应的加权矩阵如下：

$$\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \& \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \tag{3-16}$$

最后定义二维函数 f(x,y)的二价偏微分，即拉普拉斯算子如下：

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{3-17}$$

离散形式的拉普拉斯算子计算如下：

$$\nabla^2 f = 4z_5 - (z_2 + z_4 + z_6 + z_8) \tag{3-18}$$

相应的加权矩阵如下：

$$\nabla^2 f = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \tag{3-19}$$

上面已详细地讨论了一些对离散数据进行偏微分的方法，如梯度、拉普拉斯算子，据此举一个边缘检测的例子。从图 3.4 中注意到，一阶微分的大小能有效测出一维图像的边缘。显而易见，当一阶微分值超过一定的阈值，一属于边缘的点就被检测出来。将其推广至二维图像，当其梯度值超过一阈值 T 时，就判定这个点属于边缘，即：

$$\sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} = \sqrt{G_x^2 + G_y^2} > T \tag{3-20}$$

下面图 3.5(a)和图 3.5 (b)示出了此技术的应用。图 3.5(a)是原始图像，面图 3.5 (b)是只有符合式(3-20)象素构成的图像。

另外一种边缘检测方法就是将图像与一函数卷积，此函数是对二维 Gaussian 函数进行拉普拉斯运算所产生的。二维 Gaussian 函数表示如下：

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@hotmail.com

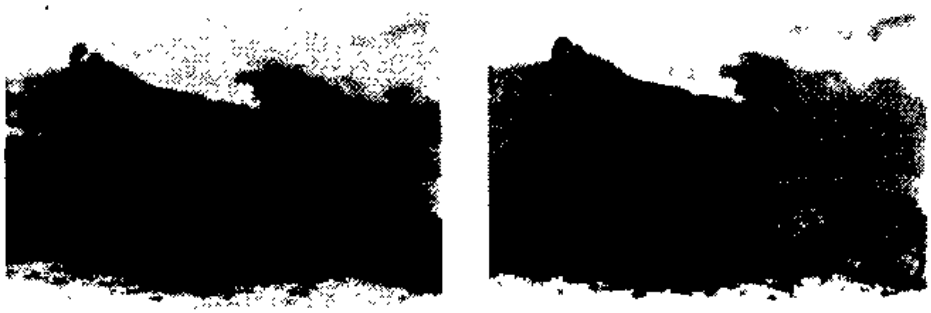


图 3.5(a) 原始图像　　图 3.5(b) 边缘检测后图像

$$h(x,y)=e^{-\left(\frac{x^2y^2}{2\sigma^2}\right)} \tag{3-21}$$

进行拉普拉斯运算后：

$$\nabla^2 h=\frac{x^2+y^2-\sigma^2}{\sigma^4}e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)} \tag{3-22}$$

结果以图形示于下图 3.6。

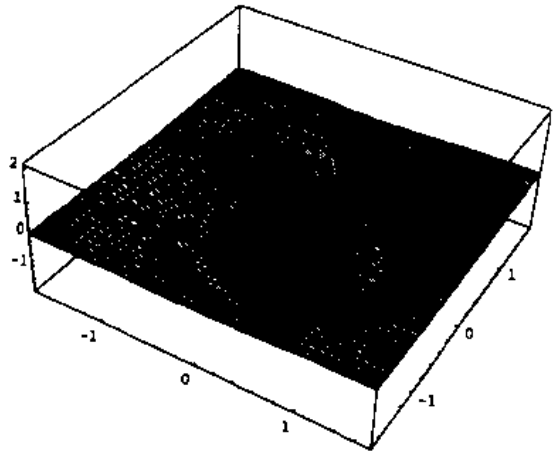


图 3.6(a) $\nabla^2 h$ 的三维图

一维情况下两个函数 f(x)与 g(x)的连续卷积定义如下：

$$f(x)\cdot g(x)=\int_{-\infty}^{\infty}f(\alpha)g(x-\alpha)d\alpha \tag{3-23}$$

离散形式的卷积定义如下：

$$f(x)\cdot g(x)=\sum_{m=0}^{M}f(m)g(x-m) \tag{3-24}$$

显然，在连续情况下，式(3-22)可扩展为二维函数卷积，如下：

$$f(x,y)\cdot g(x,y)=\int_{0}^{\infty}\int_{0}^{\infty}f(\alpha,\beta)g(x-\alpha,y-\beta)d\alpha\,d\beta \tag{3-25}$$

相应离散形式的二维函数卷积定义如下：

$$f(x,y)\cdot g(x,y)=\sum_{m=0}^{M}\sum_{n=0}^{N}f(m,n)g(x-m,y-n) \tag{3-26}$$

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

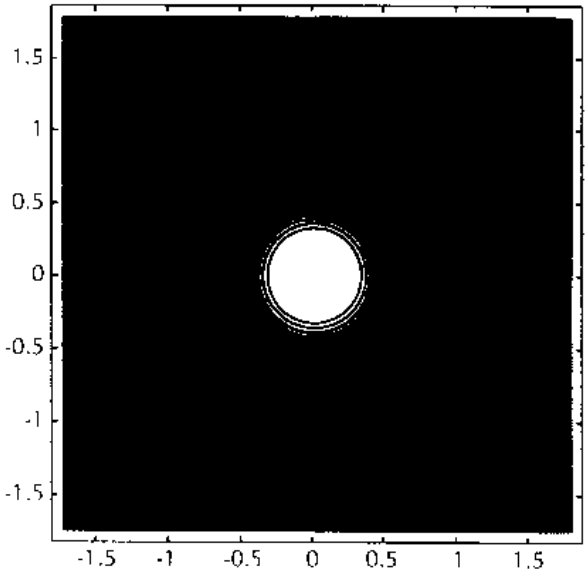


图 3.6(b) $\nabla^2 h$ 的等值线图

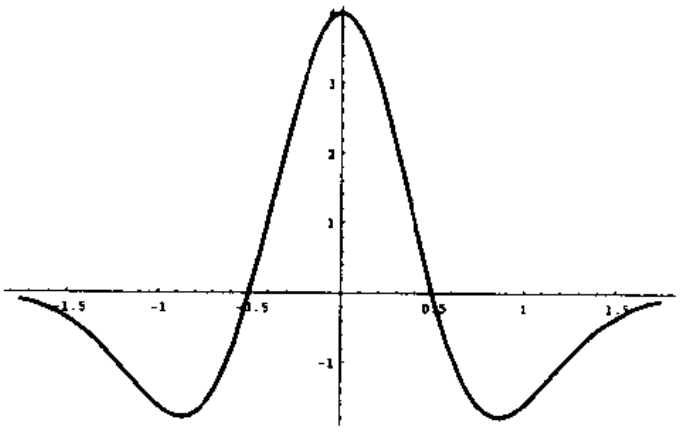


图 3.6(c) $\nabla^2 h$ 的二维图

如上所述，可以通过将图像与$\nabla^2 h(x,y)$进行卷积以检测出边缘。将卷积后得到的矩阵中的负值设为黑、正值设为白，然后保留黑色与白色相连处的象素，去除其它处的象素，这样就很容易地检测出边缘。

3.5 骨 架 处 理

骨架处理是一个表示平面区域的重要方法。骨架处理就是对于给定的图形使其线幅变细，从面提取线宽为一象素的中心线，面同时还保留其它的一些相关特征。注意到经过骨架处理，我们能很容易用图形表示区域。图 3.7(a)给出了一棵树的二值图像，而图 3.7(b)给出了经过骨架处理之后的相同一棵树。

对于识别器来说，字符笔画的宽度提供了非常少的有用信息，并且可能起到模糊分类的作用。因此，从某种意义上说，骨架处理，在此场合下有时称作字符细化，还能够从输入中除去不必要的或者冗余的信息。另外通过骨架处理还有助于提取几何特征(例如交叉点，终点，环

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xsyu.edu.cn

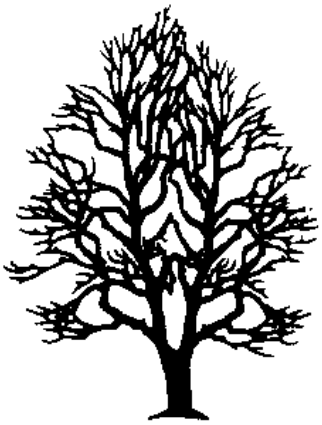


图 3.7(a) 原始图像

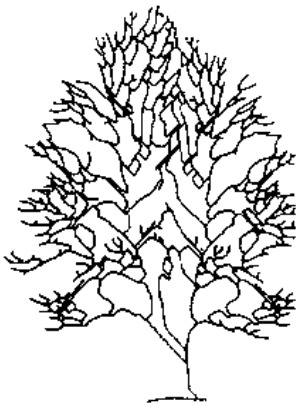


图 3.7(b) 骨架处理后的图像

等)。字符细化能理想地将给定图形表示成线宽为一的中心线,同时能保留其它所有相关特征。一种骨架处理的方法是去除图像边缘直到只剩下字符的骨架为止。这可通过对图像进行光栅扫描并使用存储器中的模板对图像检验,详细讨论见 1989 年 LeCun 等的论文。每个模块均提供一组条件,在这组条件下删除中心象素。图 3.8 给出了一些模板的例子,同时给出了用这些模板对字符细化后的结果。

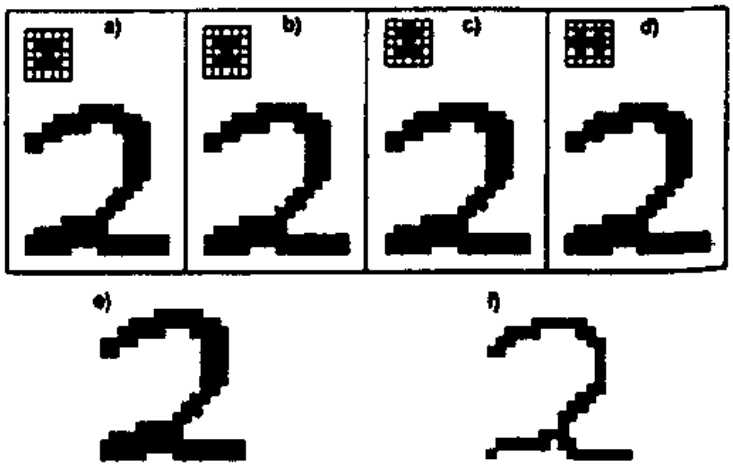


图 3.8 LeCun 使用的四个模板例子和字符细化后的结果

(摘自 LeCun, Y. , IEEE Commun. Mag., vol.27,No.11,pp.41-46,1989,征得 LeCun 的同意)

Darwiche 等于 1992 年详细地论述了一种字符细化方法。为了举个例子详细阐述以下细化的过程,我们沿用 Darwiche 这篇文献中的表示方法。四个象素模板定义如下:

$$\begin{bmatrix} X & 0 & X \\ X & 1 & X \\ X & X & X \end{bmatrix} \begin{bmatrix} X & X & X \\ X & 1 & 0 \\ X & X & X \end{bmatrix} \begin{bmatrix} X & X & X \\ X & 1 & X \\ X & 0 & X \end{bmatrix} \begin{bmatrix} X & X & X \\ 0 & 1 & X \\ X & X & X \end{bmatrix} \tag{3-27}$$

北 东 南 西

字符位图格式中,在"1"象素的指定方向上如果有一"0"象素,且此"1"象素满足上面的条件,那么此"1"象素就被删除。只有这样做,才能够不致于引起邻域中的"1"象素变得与之不连接。

字符细化这个问题实际上比上面所讨论的还要广泛,这可从 Jang 和 Chin[1990]的工作中

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

体会到。进而,当应用细化技术时要尽量注意。因为细化算法能引起模式失真,并导致假的特征[Liao,1992]。

3.5.1 细化的例子

下面清单 3.4 是细化算法的类定义以及其它最重要的一些方法的 C++ 程序代码,同时也列出了方法的主程序。此细化算法是 3.5 节中算法的改进。注意细化时,实际上调用了二次细化算法。首先调用 Thin1 算法,接着调用 Thin2 算法。Thin1 算法没有 Thin2 方法那样积极。我们发现有些细化算法,将目标图像一些有助于识别的特征给删除掉了,面清单中的算法能减少这种问题。随书磁盘提供了完整的源代码。

清单 3.4

```
class cThinner {
private:
 int   xMax, yMax;              // Size of the bitmap
 int   PelxMax, PelyMax;
 int   PelxMin, PelyMin;
 int   M[32][32];               // Space for the whole bitmap
 int   M1[3][3];                // 3x3 receptive field — Template sized window
 tMap  Map[7];
 tgrid ZeroPosn[4];
 int   ApplyTemplate(int, int, int);
 void  Rotate(int);
 int   isConnected();
 void  buildMap();
 int   isPath(int, int);
 int   WeakCond(int x, int y);
 int   StrongCond(int x, int y);

public:
 cThinner();
 void  SetGridSize(int Mx, int My){xMax=Mx; yMax=My;}
 void  Store(int x,int y,int Pel){M[x][y]=Pel;}
 int   QueryPel(int x,int y){return M[x][y];}
 void  Thin1();
 void  Thin2();
 int   doScan(int);
 void  ShowM();
 void  fShowM(FILE *);
 void  ShowMap();
 void  LocateEndPts();
 void  LocateIntersects();
 void  SetCharExtents();
 int   doScanLR();
 int   doScanInOut();
 int   ApplyAllTemplates(int, int);
};

cThinner::cThinner(){
 ZeroPosn[0].x=1;  ZeroPosn[0].y=0;        // SOUTH
 ZeroPosn[1].x=2;  ZeroPosn[1].y=1;        // EAST
 ZeroPosn[2].x=1;  ZeroPosn[2].y=2;        // NORTH
 ZeroPosn[3].x=0;  ZeroPosn[3].y=1;        // WEST
 Map[0].Next1= 1;  Map[0].Next2=-1;
 Map[1].Next1= 3;  Map[1].Next2= 2;
 Map[2].Next1= 3;  Map[2].Next2=-1;
 Map[3].Next1= 5;  Map[3].Next2= 4;
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
 Map[4].Next1= 5;   Map[4].Next2=-1;
 Map[5].Next1= 6;   Map[5].Next2=-1;
 Map[6].Next1=-1;   Map[6].Next2=-1;
}

void cThinner::Thin1(){
 int Tid;
 int dirty;
SetCharExtents();
dirty=1;
while (dirty) {
 //dirty=doScanLR();
 dirty=doScanInOut();
 } /* endwhile */
}

void cThinner::Thin2(){
 int dirty;
dirty=1;
while (dirty) {
 dirty =doScan(1);
 dirty+=doScan(3);
 dirty+=doScan(0);
 dirty+=doScan(2);
} /* endwhile */
}

nt cThinner::WeakCond(int x, int y){
  int i,sum;

sum=0;
if ((x==PelxMax)||(x==PelxMin)) {
 for (i=0; i<=PelyMax; i++) {
    sum+=M[x][i];
    } /* endfor */
 if (sum<2) return 0;
 } /* endif */
sum=0;
if ((y==PelyMax)||(y==PelyMin)) {
 for (i=0; i<=PelyMax; i++) {
    } /* endfor */
 if (sum<2) return 0;
 } /* endif */
return 1;
}


int cThinner::StrongCond(int x, int y){
if ((x>PelxMin) && (x<PelxMax) && (y>PelyMin) &&(y<PelyMax))
 return 1;
else
 return 0;
}

int cThinner::doScanLR(){
 int x1,y1,x2,y2;
 int dirty;
 char c;

dirty=0;
for (x1=1; x1<xMax-1; x1++) {
 x2= xMax-x1;
 for (y1=1; y1<yMax-1; y1++) {
   //if ((x1>PelxMin) &&(x1<PelxMax) && (y1>PelyMin) &&(y1<PelyMax)) {
    if (WeakCond(x1,y1)) {
     if ((ApplyTemplate(x1,y1,3)) || (ApplyTemplate(x1,y1,0))|| (ApplyTemplate(x1,y1,2)))
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
            dirty=1;
          } /* endif */
        y2=y1;
       //if ((x2>PelxMin) &&(x2<PelxMax) && (y2>PelyMin) &&(y2<PelyMax)) {
        if (WeakCond(x2,y2)) {
          if (ApplyTemplate(x2,y2,1))
            dirty=1;
          } /* endif */
        } /* endfor */
      } /* endfor */
    return dirty;
    }
    int cThinner::doScanInOut(){
     int Q1x,Q1y,Q2x,Q2y;
     int Q3x,Q3y,Q4x,Q4y;
     int CenterX1, CenterX2;
     int CenterY1, CenterY2;
     int dirty;
     int done;
     char c;

    CenterX1= PelxMin+(PelxMax-PelxMin)/2;
    CenterX2= CenterX1+1;
    CenterY1= PelyMin+(PelyMax-PelyMin)/2;
    CenterY2=CenterY1+1;
    Q1x=CenterX2; Q2x=CenterX2; Q3x=CenterX1; Q4x=CenterX1;
    Q1y=CenterY2; Q2y=CenterY1; Q3y=CenterY1; Q4y=CenterY2;

    dirty=0; done=0;
    while (!done) {
    // Apply the templates
    if (WeakCond(Q1x,Q1y) && (Q1x<=PelxMax))
      dirty |= ApplyAllTemplates(Q1x, Q1y);
    if (WeakCond(Q2x,Q2y) && (Q2x<=PelxMax))
      dirty |= ApplyAllTemplates(Q2x, Q2y);
    if (WeakCond(Q3x,Q3y)&& (Q3x>=PelxMin))
      dirty |= ApplyAllTemplates(Q3x, O3y);
    if (WeakCond(Q4x,Q4y)&& (Q4x>=PelxMin))
      dirty |= ApplyAllTemplates(Q4x, Q4y);
    // Move the templates
    Q1y++;
    if (Q1y>PelyMax) {
      Q1y=CenterY2;
      Q1x++;
      }
    O2y—;
    if (Q2y<PelxMin) {
      Q2y=CenterY1;
      Q2x++;
      }
    O3y—;
    if (Q3y<PelyMin) {
      Q3y=CenterY1;
      Q3x—;
      }
    Q4y++;
    if (Q4y>PelyMax) {
      Q4y=CenterY2;
      Q4x—;
      }

    if ((Q1x>PelxMax) && (Q2x>PelxMax) && (Q3x<PelxMin) && (Q4x<PelxMin))
      done=1;
    } /* endwhile */
  return dirty;
  }
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@163.com


```
int cThinner::doScan(int Tid){
  int x,y;
  int dirty;
dirty=0;
for (y=1; y<yMax-1; y++) {
  for (x=1; x<xMax-1; x++) {
    if (ApplyTemplate(x,y,Tid))
          dirty=1;
    } /* endfor */
  } /* endfor */
return dirty;
}

int cThinner::ApplyAllTemplates(int x, int y){
int rc;

rc=0;
if (ApplyTemplate(x, y, 1)) rc=1;
if (ApplyTemplate(x, y, 3)) rc=1;
if (ApplyTemplate(x, y, 0)) rc=1;
if (ApplyTemplate(x, y, 2)) rc=1;
return rc;
}
/*****************************************************************************
* int cThinner::ApplyTemplate(int x, int y, int Tid)                         *
*                                                                            *
*  Removes center pixel when template is applicable                          *
*                                                                            *
*  x,y locates center of 3x3 template placement                              *
*  Tid is the templates Identification                                       *
*****************************************************************************/
int cThinner::ApplyTemplate(int x, int y, int Tid){
  int i,j,sum;

sum=0;
for (j=0; j<3; j++) {
  for (i=0; i<3; i++) {
    M1[i][j] = M[x+i-1][y+j-1];
    sum+=M1[i][j];
  } /* endfor */
} /* endfor */

if(M1[1][1] != 1) return 0;                                  // Template mismatch
if (M1[ZeroPosn[Tid].x][ZeroPosn[Tid].y] != 0) return 0;     // Template mismatch
if(sum<=2) return 0;                                         // never delete an isolated pixel
if(sum==8) return 0;                                         // don't hollow out the core of fat lines
Rotate(Tid);
if (isConnected()) {
  M[x][y]=0;                                                 // Strip the pixel
  return 1;                                                  // DIRTY
  }
 else {
  return 0;                                                  // NOT DIRTY
  } /* endif */
return 0;
}

void cThinner::Rotate(int Tid){
 int t,i,j,m,n;
 int Temp[3][3];
for (t=0; t<Tid; t++) {
 for (i=0; i<3; i++) {
    for (j=0; j<3; j++) {
     Temp[i][j] = M1[2-j][i];
     } /* endfor */
  } /* endfor */
 for (m=0; m<3; m++) {
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication& Cooperation: luzhenbo@yahoo.com.cn


```
        for (n=0; n<3; n++) {
          M1[m][n] = Temp[m][n];
          } /* endfor */
        } /* endfor */
      } /* endfor */
    }

    void cThinner::buildMap(){
      Map[0].Pel=M1[0][0];
      Map[1].Pel=M1[0][1];
      Map[2].Pel=M1[0][2];
      Map[3].Pel=M1[1][2];
      Map[4].Pel=M1[2][2];
      Map[5].Pel=M1[2][1];
      Map[6].Pel=M1[2][0];
    }

    int cThinner::isPath(int i, int j) {
      int k;

    k=Map[i].Next1;
    while ((k<j) && (k>-1)) {
      if (Map[k].Pel) {
        if (Map[k].Next1==j) return 1;
        if (Map[k].Next2==j) return 1;
        k=Map[k].Next1;
        }
      else {
        return 0;
        } /* endif */
      } /* endwhile */
    printf("ERROR:Problem in isPath");
    return 1;
    }

    int cThinner::isConnected(){
      int i,j,rc;
    rc=1;
    buildMap();
    for (i=0; i<=4; i+=2) {
      for (j=i+2; j<=6; j++) {
        if ((Map[i].Pel==1) && (Map[j].Pel==1)) {
          If (!isPath(i,j)) rc=0;
          }
        else {
          } /* endif */
        } /* endfor */
      } /* endfor */
    for (i=1; i<=3; i+=2) {
      for (j=i+3; j<=6; j++) {
        if ((Map[i].Pel==1) && (Map[j].Pel==1)) {
          if (!isPath(i,j)) rc=0;
          }
          else {
            } /* endif */
          } /* endfor */
        } /* endfor */
      return rc;
      }

      int main(int argc, char *argv[])
      {
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
.
cThinner     Thinner;                    // create an instance
      .
      .
  // setup thinner data here.
  // Then Invoke.....
      .
 Thinner.Thin1();            // run first w/ weak condition
 Thinner.Thin2();            // run with strong condition
      .
      .
      .
}
```

3.6 处理手写板输入

手写板输入的数据是以笔划为单位的,这样输入的字符常常有一到四个这样的笔划。每一笔划都由笛卡尔坐标系中沿时间轴的有序(x,y)坐标表示。图 3.9(a)是笔划数据的典型坐标序列,图 3.9(b)是它的图形表示。

```
=S   19
 988 2836 986 2842 984 2848 984 2852 984 2854 982 2856 982 2858 980 2858
 982 2852  990 2826  998 2776 1010 2714 1024 2650 1030 2610 1036 2578 1040 2556
1040 2542 1040 2540 1034 2544
=S   15
 862 2838  864 2842  866 2842  868 2844  876 2844  888 2844  904 2844  930 2844
 986 2854 1034 2864 1078 2872 1112  2876 1140 2876 1154 2876 1162 2876
=S   44
1150 2786 1150 2790 1148 2790 1148 2792 1150 2792 1148 2792 1148 2790 1150 2786
1150 2782 1148 2786 1146 2788 1146 2792 1144 2794 1144 2796 1142 2790 1144 2794
1150 2780 1164 2740 1174 2702 1182 2664 1192 2634 1206 2600 1214 2584 1216 2568
1216 2552 1216 2540 1216 2538 1216 2540 1216 2542 1214 2544 1214 2546 1214 2554
1214 2564 1216 2580 1220 2604 1236 2636 1254 2650 1274 2650 1290 2640 1308 2608
1316 2586 1322 2570 1330 2562 1332 2556
=S   21
1404 2624 1414 2624 1438 2622 1458 2624 1478 2630 1488 2642 1494 2650 1494 2656
1490 2660 1478 2666 1456 2668 1454 2664 1416 2652 1404 2626 1406 2592 1418 2572
1438 2558 1468 2550 1506 2546 1528 2550 1546 2552
=E
```

图 3.9(a) "The"的笔划数据

下面的 C++代码,给出了如何将手写板数据变换成适于以后章节中将讨论方法所使用的位图表示。

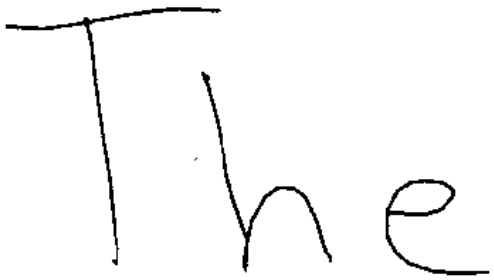


图 3.9(b) "The"的象素图表示

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

清单 3.5

```
struct aPoint {
  int x;                          // x coord of point in pixel map
  int y;                          // y coord of point in pixel map
  int id;                         // id of stroke to which point belongs
  };

struct aPoint point[1024];
char Grid[1024][1024];

int ReadStrokeFile(char *lnName,long& maxx, long& maxy,Int InvFlag){
   FILE *fpIn;
  char Name[180];
  char command[180];
  int  count,dun,sid;
  int  i,x,y,xyindx;
  long minx,miny;
strcpy(Name,lnName);
strcat(Name,".RAW");
minx=65535; miny=65535;
if((fpIn = fopen(Name,"r")) == NULL){
  printf("Unable to open input file:%s",lnName);
  return 0;
  }
sid=0; xyindx=0; dun=0;
while (!dun) {
  fscanf(fpIn,"%s",command);
  if (strcmp("=S",command)==0) {
     fscanf(fpIn,"%d",&count);
     for (i=0; i<count; i++) {
       fscanf(fpIn,"%d%d",&x,&y);
      point[xyindx].x=x;
      point[xyindx].y=y;
       point[xyindx].id=sid;
       xyindx++;
       minx=min(minx,x);
       miny=min(miny,y);
       } /* endfor */
      sid++;
      }
      else {
       if (strcmp("=E",command)==0) dun=1;
       if (feof(fpIn)) dun=1;
       } /* endif */
    } /* endwhile */
  maxx=0; maxy=0;
  for (i=0; i<xyindx; i++) {                    //shift image to remove
    if(sid != point[i].id)                      //excess space
      sid  = point[i].id;
    point[i].x=point[i].x-minx;
    point[i].y=point[i].y-miny;
    maxx=max(maxx,point[i].x);
    maxy=max(maxy,point[i].y);
    } /* endfor */
  if(InvFlag)  //optionally invert for screen coord system.
    for (i=0; i<xyindx; i++) {
      point[i].y=maxy-point[i].y; }
  folose(fpIn);
  return xyindx; //return number of entries in point array
  }
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@hotmail.com

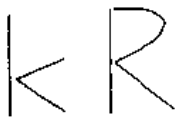

```
void PlotOnGrid(int x0, int y0, int x1, int y1){
  double m,b,rx,ry,rx0,rx1,ry0,ry1,rlasty;
  int xStart,yStart,xEnd,yEnd;
  int x,y;
//plot a line between points [x0,y0] & [x1,y1]
Grid[x0][y0]=1;
Grid[x1][y1]=1;
if (x1==x0) { //handle vert line as special case
  if (y0>y1) {
    yStart=y1; yEnd=y0; }
    else {
    yStart=y0; yEnd=y1; }
  for (y=yStart; y<=yEnd; y++) {
    Grid[x0][y]=1; } /* endfor */
  }
 else {
  ry0=(double)y0;
  rx0=(double)x0;
  ry1=(double)y1;
  rx1=(double)x1;
  xStart=min(x0,x1);
  xEnd=max(x0,x1);
  m=(ry1-ry0)/(rx1-rx0);                    //calc slope
  b=ry0-m*rx0;                              //calc intercept
  rlasty=-1.0;
  for (x=xStart; x<=xEnd; x++) {
    rx= (double)x;
    ry=m*rx+b;
    if (rlasty>=0.0){
      if (ry<rlasty) {
        for (y=(int)ry; y <= rlasty; y++) {
          Grid[x][y]=1;
          } /* endfor */
        }
       else {
        for (y=(int)rlasty; y <= ry; y++) {
          Grid[x][y]=1;
          } /* endfor */
        } /* endif */
      rlasty=ry;
      }
     else {
      Grid[x][(int)ry]=1;
      rlasty=ry;
      }
    } /* endfor */
  } /* endif */
}

void PlotStrokes(int xyPoints){
  int sid,xprev,yprev,i;
sid=-1;
//Now map all strokes onto a 2 dim grid
for (i=0; i<xyPoints; i++) {
  if(sid != point[i].id)                    //seed. must wait til a prev stroke exists
    sid  = point[i].id;
   else
    PlotOnGrid(xprev,yprev,point[i].x,point[i].y);  //plot it
  xprev=point[i].x;                         //curr stroke becomes previous stroke
  yprev=point[i].y;
  } /* endfor */
}
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
int main(int argc, char *argv[]) {
  int xyPoints;
  long Maxx,Maxy;
xyPoints=ReadStrokeFile(argv[1],Maxx,Maxy,1);     //Read in strokes from stroke file
PlotStrokes(xyPoints);                            //Plot strokes on grid.
//Pixel map is now in Grid Array
   .
   .
   .
return 1;
}
```

3.7 图像的分割

如同引言中讨论的,在识别直线模式和非直线模式的手写字符时,字符分割是非常重要的任务。字的分割可能是非常复杂的,因为字间距可能不是总比字符间距大。对于一组字符,属于不同字符的多个笔画有可能连在一起被误认为是一个字符;与此相反,一个字符的不连续笔画有可能分别构成单个字符,如图 3.10。



(a) (b)

图 3.10 字符分割的二义性

图 3.10(a)中字符 "1"和"c",而图 3.10(b)可能被误认为是"1"和"2"。可能有一种方法,就是利用时延神经网络来解决此问题。通常的方法在解决此问题时只是部分成功。用于字符分割的空间直方图方法一般只有 60%的准确度。基于距离算法的分割方法效果稍好[Seni 和 Cohen,1992]。

这些算法有:

1. 有界盒(bounding box)方法
2. 欧几里德距离方法
3. 最小游程距离方法
4. 平均游程距离方法
5. 采用有界盒启发的游程方法
6. 采用欧几里德启发的游程方法

上述几种技术的准确率据报导似乎有很好的效果,但不要太乐观,一定要记住分割是整个识别过程的前端。在分割时产生的误差将会传遍整个识别系统。说得更明白一些,如果分割出了一个错误,后面的识别系统就不可能正确地进行分类。误差是叠加的。

距离分类法不可能取得好效果的原因在于:这些方法去除了许多相关信息,也就是说:距离分类不能包含正确分类所需的所有必须信息。在考虑笔划之间的欧几里德距离时必须考虑笔画的形状。神经网络分类器就适合于解决此问题。Garris 和 Wilson 在 1992 年提出了两种神经网络解决方案,在他们的实验中两种方案均达到了 100%的准确率。与其它报导方法不同,他们的实验是基于一个大的数据库(1104 幅图像)得到的结果,因此是值得信赖的。这两

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

种方法是：

1．自组织多映射网络

2．前馈式多层感知器网络，网络输入为 Gabor 系数，网络训练使用共轭梯度算法

这里必须看到：有一种“有选择注意力”的神经认知机也能进行字符分割，此网络将分割作为整个识别的一部分。所有这些方法都将在后面的章节中详细的论述。

在结束分割主题之前，有必要注意到：如果分割系统能将笔划边界识别到极微小的单位（譬如在线系统），那么系统就不必只处理位图格式文件。在这种情况下，当单个处理笔划或聚成 n 笔划组时，我们对笔划计数。计数为最大者获胜，因此就定义了字符类。

参考书与文献

Darwiche, E., Pandya, A. S., and Mandalia, A. D., "Automated optical recognition of degraded characters," *SPIE*, vol. 1661, pp. 579-84, 1992.

Garris, M. D. and Wilson, C. L., "A neural approach to concurrent character segmentation and recognition," *Proc. IEEE Southcon*, pp. 154+159, 1992.

Gonzalez, R. C. and Woods, R. E., *Digital Image Processing*, Addison-Wesley, Reading, MA, 1992.

Jang, B. K. and Chin, R. T., "Analysis of thinning algorithms using mathematical morphology," *IEEE PAMI*, vol. 12, no. 6, pp. 541-555, 1990.

LeCun, Y., Le, L. D., Jackel, L. D., Boser, J. B., Denker, J. S., Graf, H. P., Guyon, J., Henderson, D., Howard, R. E., and Hubbard, W., "Handwritten digit recognition: applications of neural network chips and automatic learning," *IEEE Commun. Mag.*, vol. 27, no. 11, pp. 41-46, 1989.

Levine, J., *Programming for Graphics Files in C and C++*, John Wiley & Sons, New York, 1994.

Liao, H. Y., Huang, J. S., and Huang, S.-T., "Two-dimensional networks for handwritten Chinese character recognition," *IEEE Conf., Baltimore*, vol. 3, pp. 579-84, 1992.

Nelson, M., *The Data Compression Book*, M & T Books, San Mateo, CA, 1992.

Rimmer, S., *Supercharged Bitmapped Graphics*, Windcrest/McGraw-Hill, New York, 1992.

Seni, G. and Cohen, E., "Segmenting handwritten text lines into words using distance algorithms," *Proc. SPIE*, vol. 1661, pp. 61-72, 1992.

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xidian.edu.cn

第四章 有监督学习的前馈网络

4.1 前馈多层感知器结构

由反向传播算法[Rumelhart 等,1986]训练的多层感知器,是神经网络分类器中最普遍、最通用的形式。已经证明:由一个单隐层和非线性兴奋函数组成的多层感知器网络,是通用的分类器[Funahashi, 1989;Cybenko, 1989, Hartman 等,1990;Hornik 等,1989]。也就是说,这样的网络能逼近任意复杂的决策边界。这样,给定一个表示输入模式的由一系列特征组成的输入矢量,我们便知,具有固有鉴别能力的这个网络可以作为强识别器来使用。这并不意味着我们能得到零误差。类经常是重迭的,所以,通过极小化由于错误分类导致的误差而得出的分类结果不能产生零误差。网络容量缺陷更是一个模棱两可的问题。在神经网络当中,多层感知器网络是最有力的工具。在开始用神经网络方法讨论模式识别问题时,多层感知器网络占有重要位置。图 4.1 给出一粗略的多层感知网络结构。

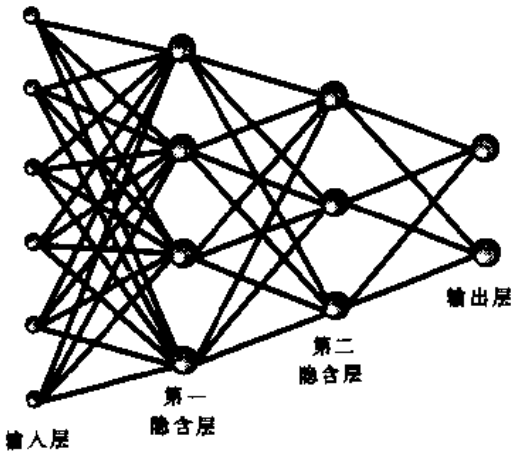


图 4.1 前馈多层感知器结构

将代表待识别模式的输入矢量输入至输入层,并传至后面的隐含层,最后通过连接权输出到输出层。该网络中每个神经元通过求输入权值和及经非线性兴奋函数传递结果来工作,其数学描述如下:

$$out_i = f(net_i) = f\left(\sum_j W_{ij} out_j + \theta_i\right) \tag{4-1}$$

这里 out_i 是所考虑层中第 i 个神经元的输出;out_j 是前一层第 j 个神经元的输出。对非线性兴奋函数 f 的使用有几种常用形式,其中经常采用的是 sigmoid 函数:

$$f(net_i) = \frac{1}{1 + e^{\frac{-net_i}{Q_0}}} \tag{4-2}$$

式(4-2)中 Q_0 代表神经元温度,温度越高,sigmoid 函数变化越趋平缓。在非常低的温度下,它接近于阶跃函数。图 4.2 给出了 sigmoid 函数温度变化曲线。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & cooperation: luzhenbo@yahoo.com.cn

注意在 $x=0$ 附近,几乎是一个线性区,而该区域宽度取决于温度。

训练一个神经网络,反对称兴奋函数似乎有一些优势。当一个函数满足式(4-3)时,它便是反对称函数:

$$f(-x)=-f(x) \tag{4-3}$$

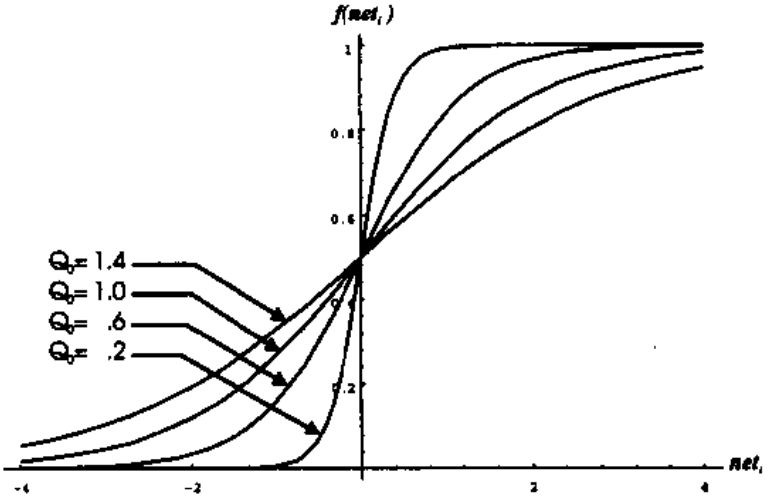


图 4.2 Sigmoid 函数

经常被用作兴奋函数的反对称函数是双曲正切函数。对双曲正切函数,式(4-2)变为:

$$f(net_i)=\tanh(net_i)=\frac{1-e^{-net_i}}{1+e^{-net_i}} \tag{4-4}$$

在图 4.3 中可见双曲正切兴奋函数反对称特性明显。

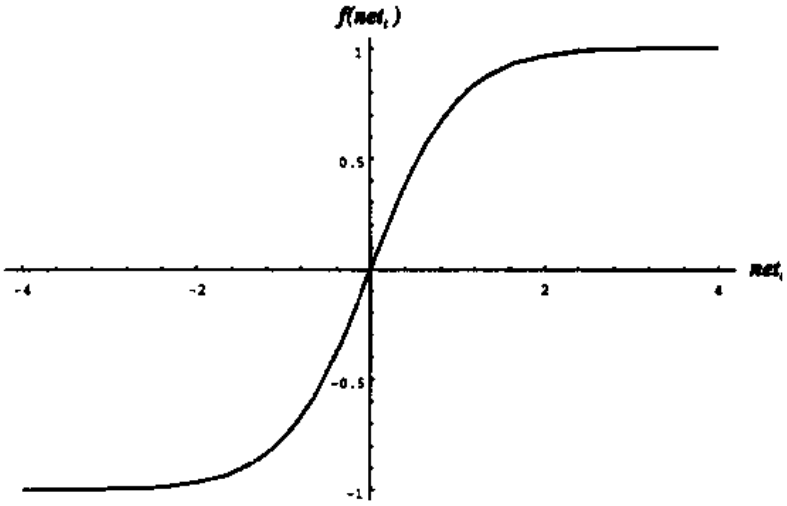


图 4.3 双曲正切函数

把输入模式映射到相应分类器所需知识由权值来体现。最初,被给定问题域的权值是未知的,直到找到有用权值之前,神经网络不能解决这个问题。寻找有用权值集合的过程,称为训练。训练首先是提供训练集合,它由输入样本和与之相对应的代表正确分类的输出组成。训练集合的每一矢量是否有特定的期望输出,就是有监督和无监督学习间的差别。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@hotmail.com

网络训练过程,包括从训练集合到权值集合的映射。至少在给定误差内,该组权值可对训练集矢量正确分类。实际上,网络所学正是训练集合所教。如果合理选择训练集合,并且训练算法有效,那么,网络应能正确对不属于训练集合的输入量正确分类。这个现象有时称作为推广。在后面章节中,将详细讨论如何得到一个"合理的"训练集。

这样,我们看到把神经网络应用于模式识别问题包括网个截然不同的阶段。在网络训练阶段,如图 4.4(a)调整网络权值以表现问题域。第二阶段或称工作阶段,权值固定不变,并且当把实验数据或实际数据输入到网络时,网络能够对其分类。这个阶段见如图 4.4(b)。

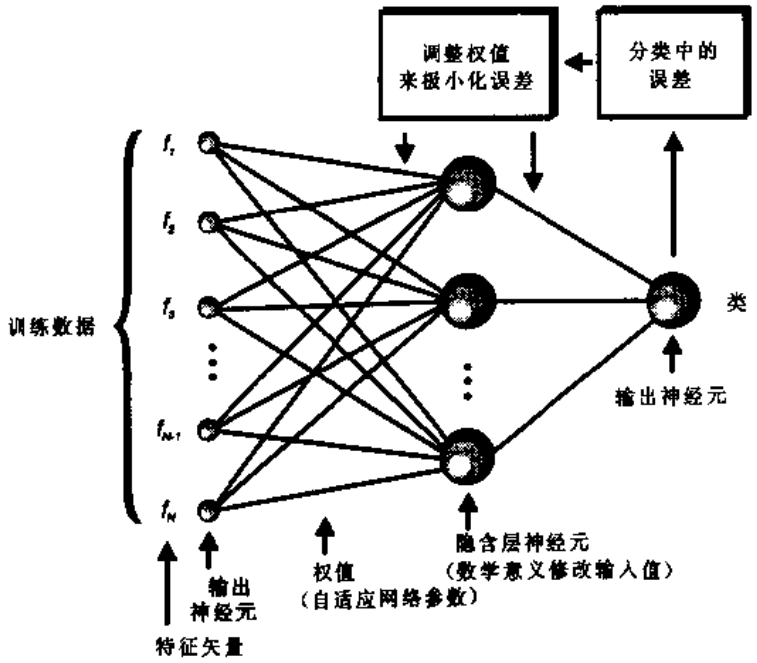


图 4.4(a) 训练阶段

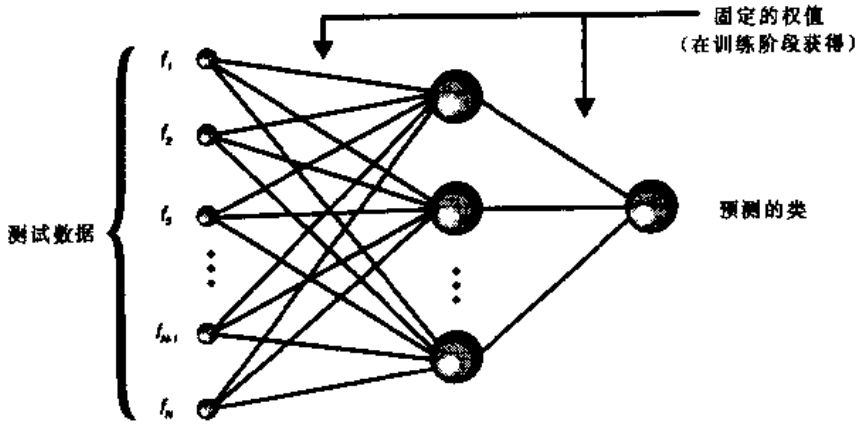


图 4.4(b) 预测阶段

4.2 用 C++ 实现前馈多层感知器

学习了一些有关理论知识后,我们现在来研究怎样具体实现前馈神经网络。首先看一下

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and Cooperation: luzhenbo@yahoo.com.cn

头文件,这样有助于了解网络的完整结构。

清单 4.1

```
/*-----------------------------------------------*
 *                                               *
 * Code listing PNET.HPP                         *
 *                                               *
 *-----------------------------------------------*/
#define     NEURONHIGH  1.0          // Neuron's high output value
#define     NEURONLOW   0.0          // Neuron's low output value
#define     TRUE        1
#define     FALSE       0

class       WEIGHT                   // Forward reference

struct      WEIGHTIMAGE {
 double     data;                    // Weight value
 int        sneuron;                 // Source neuron for this weight
 int        dneuron;                 // Dest neuron for this weight
 WEIGHTIMAGE *next;
};

struct NETRESULTS {
 int        index;                   // Neurons identification number
 double     value;                   // Neurons output value
 char       character;               // char representation of digit
};

class NETFILEDATA {
private:
 double     temperature;             // Neurons temperature
 double     threshold;               // Neurons firing threshold
 int        Nlayers;                 // Number of layers in the net
 int        neurons[MAXLAYERS];      // Number of neurons per layer
 int        status;                  // Error status ( 0 = OK)
 WEIGHTIMAGE *weights[MAXLAYERS - 1]; //Temp weight storage area
 void       ADDweights(int l, int d, int s, double w);
 double     GETweights(int l, int d, int s);

public:
 NETFILEDATA(void);
 int   SetupNet(char *);
 double GetTemp(void) { return temperature; }
 double GetThresh(void) { return threshold; }
 int   GetNlayers(void) { return Nlayers; }
 int   GetLayerSize(int layer) { return neurons[layer]; }
 double GetWeight(int l, int d, int s) { return GETweights(l-1, d, s); }
 int   GetStatus(void) { return status; }
};
//*****************************************************************
//   THE NEURON CLASS                                             *
//*****************************************************************
class NEURON {
private:
 static double temperature;        //Holds a single copy for all neurons
 static double threshold;          // Holds a single copy for all the neurons
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
 int          id;              // Holds a neuron identification number
 double       out;             // Holds a neurons output value
  WEIGHT      *weight1;        // Pointer to list of weights (head)
  WEIGHT      *weightL;        // Pointer to list of weights (tail)
 int          BiasFlg;         // 1 = Bias Neuron, 0 otherwise
 NEURON       *next;           // Hook to allow neurons to be list members

public:
 NEURON(void) { id = 0; out = 0;
                weight1 = (WEIGHT *)NULL; next = (NEURON *)NULL; }
 NEURON(int ident, int bias=0) { Id = ident; out = 0; BiasFlg=bias;
                weight1 = (WEIGHT *)NULL; next = (NEURON *)NULL; }
 void calc(void);              // Update out based on weights/inputs
 void SetNext(NEURON *N) { next = N; }
 NEURON *GetNext(void) { return next; }
 void SetWeight(double Wght, NEURON *SrcPtr);
 int GetId(void) { return id; }
 double GetOut(void) { return out; }
 void SetTemperature(double tmpr) { temperature = tmpr; }
 void SetThreshold(double thrsh) { threshold = thrsh; }
 void SetOut(double val) { out = val; }
 int IsBias(){return BiasFlg;}
};
double NEURON::temperature = 0.0;          // REQUIRED for static data elements
double NEURON::threshold  = 0.0;

//***************************************************************
//    THE WEIGHT CLASS                                          *
//***************************************************************
class WEIGHT {
private:
 NEURON   *SRCneuron;      // Source neuron for this weight
 double   WtVal;           // Magnitude of weight
  WEIGHT  *next;           // Hook so weights can be list members
public:
 WEIGHT(double W, NEURON *SN) { next = (WEIGHT *)NULL;
                SRCneuron = SN; WtVal = W; }
 NEURON *GetSRCNeuron(void) { return SRCneuron; }
 double getWeight(void) { return WtVal; }
 void  SetNext(WEIGHT *W) { next = W; }
 WEIGHT *GetNext(void) { return next; }
};
//***************************************************************
//     THE  LAYER CLASS                                         *
//***************************************************************
class LAYER {
private:
  int          LayerID;        // 0 for input, 1 for 1st hidden,...
 unsigned int Ncount;          // # of neurons in layer;
 NEURON   *Neuron1;            // Pointer to 1st neuron in layer
 NEURON   *NeuronL;            // Pointer to last neuron in layer
  LAYER      *next;            // Hook so layers can be list members

public:
 LAYER(int layer_id, NETFILEDATA *netdata);
 int SetWeights(NEURON *PrevNeuron, NETFILEDATA *neidats);
 void SetNext(LAYER *Nlayer) { next = Nlayer; }
 int GetLayerID(void) { return LayerID; }
 NEURON *GetFirstNeuron() { return Neuron1; }
 LAYER *GetNext(void) { return next; }
 unsigned int getCount(void) { return Ncount; }
 void calc(int);
};
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
//*************************************************************************
// THE NETWORK CLASS                                                      *
//*************************************************************************
class NETWORK {
private:
 int            Alive;          // True when weights are valid
 NETFILEDATA    netdata;        // Class to load saved weights
 int            Nlayers;        // Number of layers in the network
 LAYER          *INlayer;       // Pointer to the input layer
 LAYER          *OUTlayer;      // Pointer to the output layer
 NETWORK        *next;          // Need more than 1 network ... OK

public:
 NETWORK(void) { Alive = 0; next=(NETWORK *)NULL; }
 int Setup(char *);
 void ApplyVector(unsigned char *, int);
 void RunNetwork(void);
 int RequestNthOutNeuron(int, NETRESULTS *);
 double RequestTemp(void);
 double RequestThresh(void);
 int RequestLayerSize(int);
 int GetAlive() { return Alive; }
 NETWORK *GetNext() {return next;}
 void SetNext(NETWORK *netptr) {next=netptr;}
private:
 int SetWeights(void);
};
```

很明显,从类结构可以看出,层数和每层神经元数仅受可用内存的限制。如图 4.5 所示,动态分配的层对象链接表构成网络类。

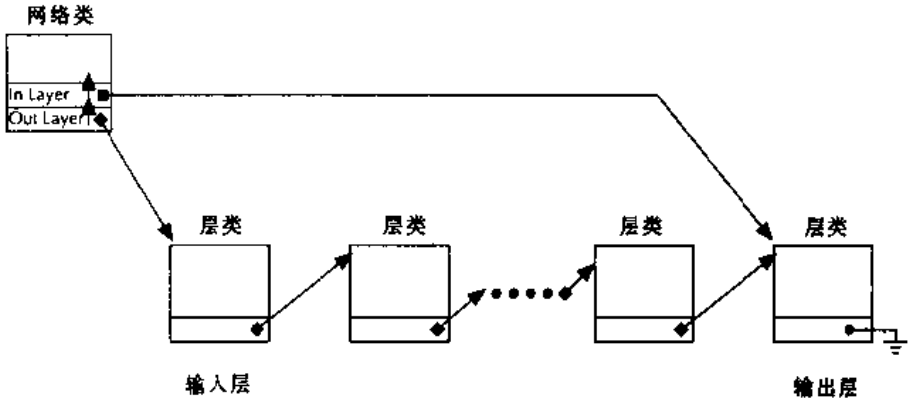


图 4.5 网络类

以类似方式,神经元对象的链接表构成层类。每个神经元对象依次指向权值对象链接表。最后,每一个权对象既包含该权值,又包含指向上一层源神经元指针。层类的完整结构见图 4.6。

同层类和神经元类相关的权类见图 4.7。

在上述类中只有网络类对于调用程序必须透明。网络类中的 **ApplyVector** 函数把一输入矢量输入至输入层神经元。**RunNetwork** 函数调用每个层 **calc** 函数,该函数起始于第一隐含层

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@hotmail.com

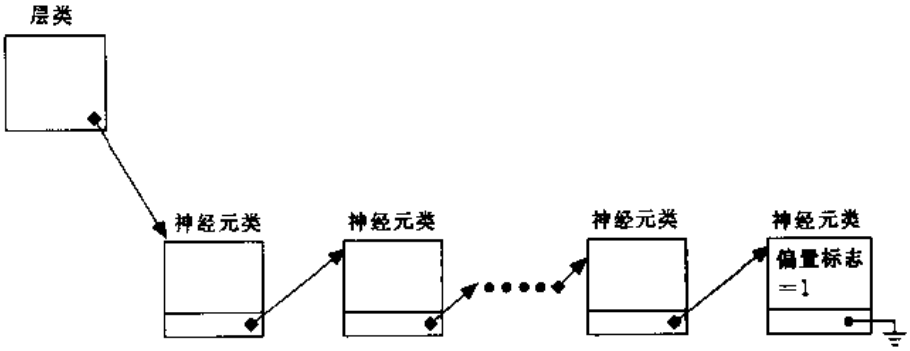


图 4.6 层类

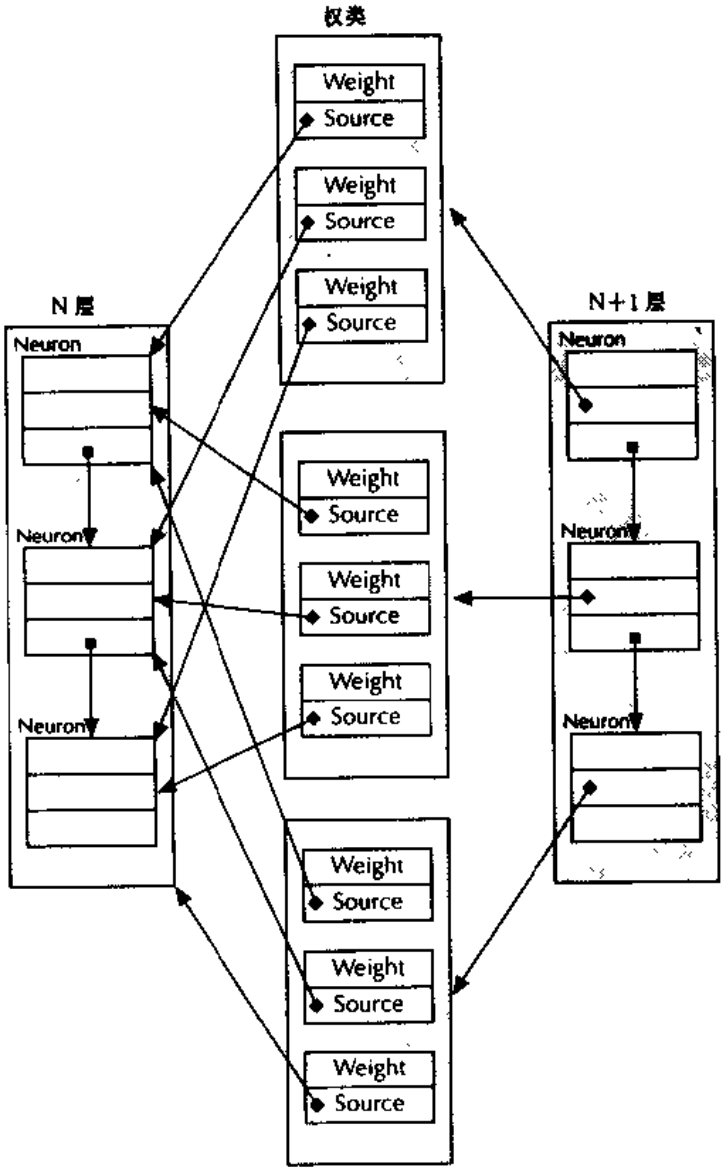


图 4.7 权类

终止于输出层。反过来，当调用层 calc 函数时，每个神经元 calc 函数也被调用。所有这些对调用程序透明。实际上，网络类要求每层都计算本层值，并且该层接到请求后也要本层每个神

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

经元计算它自己的值。最后，每个神经元同与它相关的权类对象链表被用来计算神经元兴奋值。下面给出一种更为重要方法的C++实现。随书附盘提供了完整的源程序。

清单 4.2

```
#include "pnet.hpp"

//*************************************************************************
//  METHODS FOR CLASS NETFILEDATA                                        *
//*************************************************************************

NETFILEDATA::NETFILEDATA(){
  int i;
for (i=0; i<(MAXLAYERS-1); i++) {
  weights[i]= (WEIGHTIMAGE *)NULL;
  } /* endfor */
}
 int NETFILEDATA::SetupNet(char *wgt_file_name) {
   FILE  *wgt_file_ptr;                          // Pointer to a weight file
   double AWeight;                               // Used to hold a temporary weight
   if((wgt_file_ptr = fopen(wgt_file_name, "r")) == NULL)
      return 1;
   fscanf(wgt_file_ptr, "%lg", &threshold);      // Read in threshold value
   fscanf(wgt_file_ptr, "%lg", &temperature);    // Read in temperature value
   fscanf(wgt_file_ptr, "%d", &Nlayers);         // Read in # of layers
   for(int j = 0; j < MAXLAYERS; j++)
     neurons[j] = 0;                             // Initialize array to zero
   for(int i = 0; i < Nlayers; i++)
     fscanf(wgt_file_ptr, "%d", &neurons[i]);    // Read # neurons in each layer
   for(int lyr = 1; lyr < Nlayers; lyr++)        // Traverse all layers
     {
      for(int dn = 0; dn < neurons[lyr]; dn++)   // Traverse dest layer nodes
        {                                        // Pick up bias neuron
         for(int sn = 0; sn <= neurons[lyr-1]; sn++)  // Traverse src lyr nodes
           {
           //Read in the weight from the weight file
            fscanf(wgt_file_ptr, "%lg", &AWeight);
            ADDweights(lyr-1, dn, sn, AWeight);  // Add new weight to the net
           }
        }
     }
   fclose(wgt_file_ptr);                         // Close the weight file
   return 0;
}

void NETFILEDATA::ADDweights(int l, int d, int s, double w){
  WEIGHTIMAGE *Wl   = weights[l],               //Point to weights for this layer
              *Wnew = new WEIGHTIMAGE,          //Create a new weight
              *cursor,
              *trailer;
  Wnew->data   = w;                             // Assign Wnew with
  Wnew->dneuron = d;                            //      the information
  Wnew->sneuron = s;                            //         that was passed in
  Wnew->next   = (WEIGHTIMAGE *)NULL;
  if(Wl) {
     cursor  = Wl;
     trailer = (WEIGHTIMAGE *)NULL;
     while(cursor)  {
        trailer = cursor;
        cursor  = cursor->next;
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@hotmail.com


```
      }
    trailer->next = Wnew;
   }
  else
    weights[l] = Wnew;
}

double NETFILEDATA::GETweights(int l, int d, int s) {
 WEIGHTIMAGE *Wl = weights[l];                    // Point to 1st weight in the current layer

 while(Wl) {
   if((Wl->sneuron == s) && (Wl->dneuron == d)) {
      status = 0;
      return Wl->data;
     }
    Wl = Wl->next;
  }
 status = 1;
 return 0.0;
}

//*************************************************************************
//  METHODS FOR CLASS NEURON                                              *
//*************************************************************************
void NEURON::SetWeight(double Wght, NEURON *SrcPtr) {
 WEIGHT *W = new WEIGHT(Wght, SrcPtr);
 if(weight1 == NULL) {
   weight1 = weightL = W;
  }
 else {
   weightL->SetNext(W);
   weightL = W;
  }
  }

  void NEURON::calc(void) {
   NEURON *Nptr;                                  // Pointer to neuron
   WEIGHT *Wptr = weight1;                        // Pointer to the first weight in the layer
   double NET  = 0.0,                             // Accumulates the sum
          PLNout,                                 // Previous layer neuron output
          Weight;                                 // Connection strength

   if (!BiasFlg) {
     while(Wptr) { // Traverse src layer & weights
        Weight = Wptr->getWeight();
      Nptr  = Wptr->GetSRCNeuron();               // Get weight between prev/curr layer
      PLNout = Nptr->GetOut();                    // Get the previous layer output (out)
      NET   += Weight * PLNout;                   // Sum(weight * out)  over the curr layer
      Wptr  = Wptr->GetNext();                    // Get the next weight in the weight list
      }
    // Calculate a neuron output using a sigmoid
    out = 1 / (1 + exp(- (NET + threshold)/temperature));
     }
   else {
    out = 1.0;                                    // force output on for bias neuron
     }
 }

 //*************************************************************************
 //  METHODS FOR CLASS LAYER                                               *
 //*************************************************************************
 LAYER::LAYER(int layer_id, NETFILEDATA *netdata) {
  NEURON *Nptr;
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and cooperation: luzhenbo@yahoo.com.cn


```
 LayerID = layer_id;
 Neuron1 = NeuronL = (NEURON *)NULL;
 next   = (LAYER *)NULL;
 //Get # of neurons in layer #layer_id
 Ncount = netdata->GetLayerSize(layer_id);
 for(int i = 0; i <= Ncount; i++) {  // include bias
     if (i==Ncount) {
      Nptr = new NEURON(i,TRUE);              // This is a bias Neuron
      }
     else {
      Nptr = new NEURON(i);                   // This is a normal Neuron
      } /* endif */
    // Attach neuron to the list
    if(Neuron1 == NULL) {
       Neuron1 = NeuronL = Nptr;
      }
     else  {
       NeuronL->SetNext(Nptr);
       NeuronL = Nptr;
      }
   }
}

void LAYER::calc(int out_layer)
{
 NEURON *Nptr = Neuron1;
  while(Nptr) {                                // Traverse the layer
   Nptr->calc();                               // Ask the neuron to calculate its own value
   Nptr = Nptr->GetNext();                     // Move to next neuron in the layer
   }
}

int LAYER::SetWeights(NEURON *PrevNeuron, NETFILEDATA *netdata) {
 NEURON *CurNeuron = Neuron1, *PrevPtr;
 double ZWeight = 0.0;
 int    curx = 0,                             // Current layer neuron index
        prevx,                                // Previous layer neuron index
        status = 0,                           // Error status ( 0 = ok)
 while(CurNeuron != NULL) {                   //Traverse curr lyr starting at 1st neuron
    if ( !CurNeuron->IsBias()) {              // Bias neurons dont have incoming wgts
       PrevPtr = PrevNeuron;                  // Pointer to 1st neuron in prev layer
       prevx   = 0;                           // Index of 1st neuron in prev layer
       while(PrevPtr) {                       // Read a weight from the file
         ZWeight = netdata->GetWeight(LayerID, curx, prevx++);
         status = netdata->GetStatus();
          if(status > 0)
            return status;
         CurNeuron->SetWeight(ZWeight, PrevPtr);
         PrevPtr = PrevPtr->GetNext();
        }
       }
    curx++;                                   // Bump index to next for current layer
   CurNeuron = CurNeuron->GetNext();          // Set pointer to next for current layer
  }
 return status;
}
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
//*************************************************************************
//  METHODS FOR CLASS NETWORK                                             *
//*************************************************************************
int NETWORK::Setup(char *wgt_file_name) {
 LAYER  *Lptr;                          // Pointer to a layer
 NEURON N;                              // Use to set statics (temp. & thres. for
NEURON class)
 int status = 0;                        // Error status > 0 if error occured
 char  *tbl_file_name;                  // holds the table file name
 // Setup network using info in the weight file
 status = netdata.SetupNet(wgt_file_name);
 if(status > 0)                         // error occurred opening weight file
   return 1;
 Nlayers = netdata.GetNlayers();        // Get the number of layers
 N.SetTemperature(netdata.GetTemp());   // Set the temp for all the neurons
 N.SetThreshold(netdata.GetThresh());   // Set the thresh for all the neurons
 for(int i = 0; i < Nlayers; i++) {
    Lptr = new LAYER(i, &netdata);
    if(i == 0)  {
      INlayer = OUTlayer = Lptr;
      }
    else  {
      OUTlayer->SetNext(Lptr);
      OUTlayer = Lptr;
      }
   }
 status = SetWeights();                 // Setup connection etrengths
 if(status > 0)                         // If > 0 then a Weight linked list
   return 2;                            //      out-of-bounds occured
 Alive = 1;                             // The network is in working condition
 return 0;                              // Return 0 on successful operation
}

int NETWORK::SetWeights(void) {
 LAYER *L1ptr = INlayer,                // Pointer to the input layer
       *L2ptr = INlayer->GetNext();     // pointer to the second layer
 int status = 0;
 while(L2ptr) {                         // Start at the hidden layer and traverse
    status = L2ptr->SetWeights(L1ptr->GetFirstNeuron(), &netdata);
    L1ptr = L1ptr->GetNext();           // Used to mark the current layer
    L2ptr = L2ptr->GetNext();           // Get the next layer in the layer list
   }
 return status;
}

void NETWORK::ApplyVector(unsigned char *InVecPtr, int size) {
  LAYER *Lptr = INlayer;                // Start at the 1st layer
  NEURON *Nptr = Lptr->GetFirstNeuron();// Start at 1st neuron in 1st layer
  double value;                         // Holds pixel value (on/off)
  unsigned char mask;                   // Holds the mask value
  while(Nptr && ( !Nptr->IsBias() ) ) {  // traverse the list or neurons applying
     for(int i = 0; i < size; i++) {          //  the inputs  (But not to bias neuron)
        mask = 0x80;
        for(int j = 0; j < 8; j++) {    // Cycle thru bit positions
if((InVecPtr[i] & mask) != 0)
 value = NEURONHIGH;
else
 value = NEURONLOW;
Nptr->SetOut(value);            // Set output of current neuron to value
Nptr = Nptr->GetNext();         // Get  next neuron in the input layer
mask = mask >> 1;
}
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and collaboration: luzhenbo@yahoo.com.cn


```
      }
    }
}

void NETWORK::RunNetwork() {
 LAYER *Lptr = INlayer->GetNext();          // Traverse layers starting w/ 1st hidden
 int out_layer=0;                           // Use to indicate that you are on output layer
 while(Lptr){                               //       and ending with the output layer
   if(Lptr->GetNext() == NULL)              // If NULL, then end of output layer so
     out_layer = 1;                         //   build sorted list of output values
   // Ask layer to calculate its values
   Lptr->calc(out_layer);
   Lptr = Lptr->GetNext();                  // Move on to next layer
  }
}
int NETWORK::RequestNthOutNeuron(int neuron_num, NETRESULTS *results) {
 if(Alive == 0)                                    // If Alive = 0 then
   return 2;                                       //   declare failure
 NEURON *Nptr = OUTlayer->GetFirstNeuron();        // Start at 1st neuron in out lyr
 while(Nptr) {   // Traverse the list of neurons
   if(neuron_num == Nptr->GetId()) {
     results->index     = neuron_num;
     results->value     = Nptr->GetOut();
     results->character = '1';
     return 0;
    }
   Nptr = Nptr->GetNext();
  }
 // OUT OF BOUNDS ERROR on link-list traversal
 return 1;
}

double NETWORK::RequestTemp(void) {
 if(Alive == 0)                             // If Alive = 0 then no network data is avail and
   return -1.0;                             //   network is not in working condition
 else
   return netdata.GetTemp();                // Return the networks temperature value
}

double NETWORK::RequestThresh(void) {
 if(Alive == 0)
   return -1.0;
 else
   return netdata.GetThresh();  // Return the networks threshold value
}
int NETWORK::RequestLayerSize(int layer_num) {
 if(Alive == 0)
   return 0;
 else if(layer_num >= MAXLAYERS)            // If layer_num is an invalid
   return 0;                                //   layer number then return 0
 else
   return netdata.GetLayerSize(layer_num);  // return # of neurons in
}                                           //   layer number layer_num
```

4.3 利用 B-P 算法进行网络训练

在开始讨论网络训练之前，首先必须度量网络收敛到其期望值的程度。这个度量量即为网络误差。我们正在讨论有监督训练，所以对于给定训练集合来说，这个期望值是已知的。以后将看到，恰当地选择训练集合是成功的网络应用的关键因素。这个训练集合必须大小适当，并且能合理地描述问题空间。现在假设这样的训练集合存在，因此我们就可以讨论怎样利用

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

它来训练一个网络。

首先定义网络误差。对于反向传播训练算法[Rumelhart 等,1986],通常使用的网络误差是熟知的均方差。但事实上并不是必须使用均方差,可使用任何连续可微误差函数。但是选择其它误差函数增加了额外的复杂性,因而需十分小心处理。但要记住,不论选用哪种函数作为误差函数,必须在网络输出期望值与实际值间提供一个有意义的度量量——距离。均方差定义如下:

$$E_p = \frac{1}{2}\sum_{j=1}^{N}(t_{pi} - O_{pi})^2 \tag{4-5}$$

式中 E_p 是第 P 个表征矢量的误差,t_{pj} 是第 j 个输出神经元的期望值(即训练集值),O_{pj} 是第 j 个输出神经元的实际值。

式中每一项都反映单个输出神经元对整个误差的影响。取绝对误差(期望值和实际值之差)的平方,可以看出远离期望值的那些输出对总误差影响最大。增加幂指数影响更明显。

反向传播算法是熟知的梯度下降训练算法中比较简单的一种。它的中心思想是调整权值使网络总误差最小。梯度下降法,也称作最速下降法,便提供了这样一种方法。每个权值都是 N 维误差空间中的一个元素。在误差空间中权值作为独立的变量,并且相应误差表面的形状由训练集合和误差函数共同决定。

权值的误差函数负梯度指向误差函数减小最快的方向。如果在权值空间沿这个矢量移动,最终将达到极小值,在这点梯度为零。但这点可能是局部极小值,稍后将对此详细说明。图 4.8 给出了误差空间中一个二维横截面处的梯度情况。

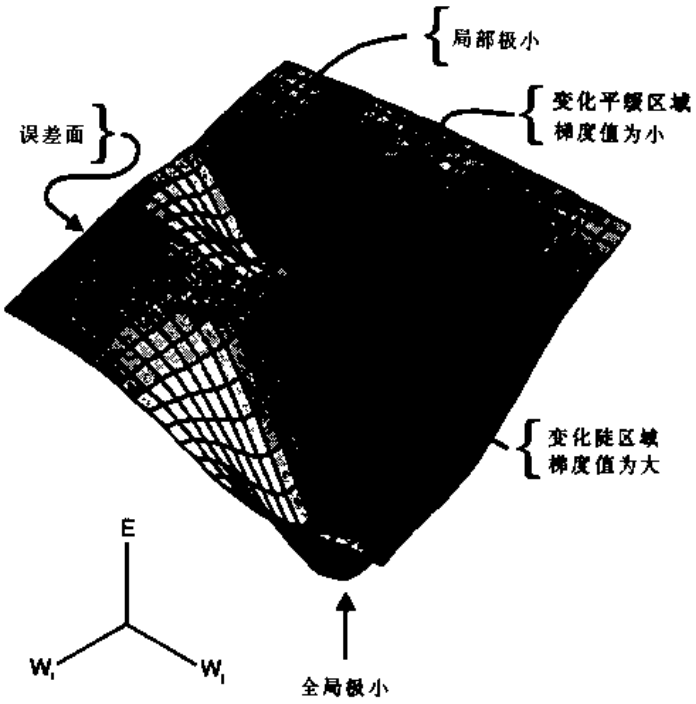


图 4.8 表面梯度图

上述结果可用数学方法描述如下:

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and cooperation: luzhenbo@yahoo.com.cn

$$\Delta_p W_{ji} \propto -\frac{\partial E_p}{\partial w_{ji}} \tag{4-6}$$

其中 $\Delta_p w_{ji}$ 表示连接 L－1 层的源神经元 i 和 L 层的目的神经元 j 权值的变化。权值的这个变化导致了权值空间中(示于图 4.8),梯度沿降低误差方向变化。

目标就是确定如何调整每个权值使网络收敛。式(4-6)说明每个权值 w_{ij} 将沿着局部误差表面最速下降的负梯度方向变化一步的关系式。

现在的任务是将式(4-6)转换成适于计算机实现的微分方程。为此,首先必须计算偏微分 $\partial E_p/\partial w_{ij}$,应用公式:

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial net_{pj}} \frac{\partial net_{pj}}{\partial w_{ji}} \tag{4-7}$$

然面 net_{pj} 由下式给出:

$$net_{pj} = \sum_i w_{ji} O_{pi} \tag{4-8}$$

式(4-8)中,对 L－1 层的所有神经元输出 O_{pi} 求和。因此,我们可以计算式(4-7)中的第二项 $\partial net_{pj}/\partial w_{ji}$,即:

$$\frac{\partial net_{pj}}{\partial w_{ji}} \frac{\partial}{\partial w_{ji}} \sum_i w_{ji} O_{pi} \tag{4-9}$$

展开式(4-9)得:

$$\frac{\partial net_{pj}}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \left(\sum_{i' \neq i} w_{ji'} O_{pi'} + w_{ji} O_{pi} \right) = O_{pi} \tag{4-10}$$

将式(4-10)代入式(4-7)得:

$$\frac{\partial E_p}{\partial w_{ji}} = O_{pi} \frac{\partial E_p}{\partial net_{pi}} \tag{4-11}$$

定义误差信号 δ_{pj} 为:

$$\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}} \tag{4-12}$$

合并式(4-11)和式(4-12)得出:

$$-\frac{\partial E_p}{\partial w_{ji}} = \delta_{pj} O_{pi} \tag{4-13}$$

将式(4-13)代入式(4-6)并且乘上一个比例常数 η,重写式(4-6)如下:

$$\Delta_p w_{ji} = \eta \delta_{pj} O_{pi} \tag{4-14}$$

常数 η 是学习速率,它控制在权值空间中权值对应每步沿负梯度方向变化的大小。

为了得到可用的微分方程,暂不考虑 δ_{pj} 的变化,运用微分公式:

$$\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}} = -\frac{\partial E_p}{\partial O_{pj}} = -\frac{\partial E_p}{\partial O_{pj}} \frac{\partial O_{pj}}{\partial net_{pj}} \tag{4-15}$$

已知,输出 O_{pj} 是 net_{pj} 的函数,其表示如下:

$$O_{pj} = f(net_{pj}) \tag{4-16}$$

$$\frac{\partial O_{pj}}{\partial net_{pj}} = f'(net_{pj}) \tag{4-17}$$

其中 f()是输出(作用)函数。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xaut.edu.cn

为了计算$\partial E_p/\partial O_{pj}$(式(4-15)中第一项),必须分别考虑下面两种情况:

1. 目的神经元 j 是一输出神经元。

2. 目的神经元 j 是一隐含层神经元。

对于输出层中的目的神经元 j,我们直接得到以 O_{pj} 为自变量的误差函数 E_p,因此可以得出:

$$\frac{\partial E_p}{\partial O_{pj}} = \frac{\partial}{\partial O_{pj}}\left[\frac{1}{2}\sum_{j'}(t_{pj'} - O_{pj'})^2\right] = -(t_{pj} - O_{pj}) \tag{4-18}$$

注意到,由式(4-18)对于特定的误差函数,我们研究了相应的算法。选择不同的误差函数会产生不同的微分方程。把式(4-17)和式(4-18)代入式(4-15),可以将 δ_{pj}(输出层的目的神经元)写成:

$$\delta_{pj} = (t_{pj} - O_{pj}) f'(net_{pj}) \tag{4-19}$$

对于隐含层中的目的神经元,不能直接对误差函数微分。因此,利用微分公式:

$$\frac{\partial E_p}{\partial O_{pj}} = \sum_k \frac{\partial E_p}{\partial net_{pk}} \frac{\partial net_{pk}}{\partial O_{pj}} \tag{4-20}$$

在式(4-20)中,对 L+1 层中的所有神经元求和。根据 net_{pk} 的定义,可以计算式(4-20)中的第二个因子:

$$\frac{\partial net_{pk}}{\partial O_{pj}} = \frac{\partial}{\partial O_{pj}}\left[\sum_l w_{kl} O_{pl}\right] = \frac{\partial}{\partial O_{pj}}\left[\sum_{l'\neq j} w_{kl'} O_{pl'} + w_{kj} O_{pj}\right] = w_{kj} \tag{4-21}$$

将式(4-21)代回式(4-20),得出:

$$\frac{\partial E_p}{\partial O_{pj}} = \sum_k \frac{\partial E_p}{\partial net_{pk}} w_{kj} \tag{4-22}$$

现在,由定义可以得到:

$$\delta_{pk} = p \frac{\partial E_p}{\partial net_{pk}} \tag{4-23}$$

将式(4-23)代入式(4-22),得出:

$$\frac{\partial E_p}{\partial O_{pj}} = \sum_k \delta_{pk} w_{kj} \tag{4-24}$$

最后,将式(4-15)、式(4-17)和式(4-24)合并,隐含层误差信号 d_{pj} 可表示为:

$$\delta_{pj} = f'(net_{pj}) \sum_k \delta_{pk} w_{kj} \tag{4-25}$$

概括上述结果,式(4-14)给出了关于 δ_{pj} 的微分方程。它对隐含层和输出层权值都有效。式(4-19)和式(4-25)分别是对应于输出层和隐层权值 δ_{pj} 的表达式。式(4-18)给出的是对应于式(4-5)的均方差的解。因此,如果使用其它的误差函数,则需要修正式(4-18)。为了得到适于数字计算机的微分方程,现仅计算 $f(net_{pj})$。为此,必须选择一个特定的输出函数 $f'(net_{pj})$ 并求出对应于这一函数的解。利用 sigmoid 函数,则得:

$$O_{pj} = f(net_{pj}) = \frac{1}{1 + e^{-net_{pj} + \theta}} \tag{4-26}$$

由式(4-17)和式(4-26),可以将 $f'(net_{pj})$ 写为:

$$f'(net_{pj}) = \frac{\partial}{\partial net_{pj}}\left(\frac{1}{1 + e^{-net_{pj} + \theta}}\right) \tag{4-27}$$

对式(4-27)求导,则得:

$$f'(net_{pj}) = \left(\frac{-1}{(1 + e^{-net_{pj} + \theta})^2}\right) \frac{\partial}{\partial net_{pj}} (1 + e^{-net_{pj} + \theta}) \tag{4-28}$$

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

下面继续计算 $f'(net_{pj})$：

$$f'(net_{pj}) = \left\{\frac{-1}{(1+e^{-net_{pj}+\theta})^2}\right\} e^{-net_{pj}+\theta} \frac{\partial}{\partial net_{pj}}(-net_{pj}+\theta) \tag{4-29}$$

$$= \left(\frac{1}{1+e^{-net_{pj}+\theta}}\right)\left(\frac{e^{-net_{pj}+\theta}}{1+e^{-net_{pj}+\theta}}\right) \tag{4-30}$$

$$= \left(\frac{1}{1+e^{-net_{pj}+\theta}}\right)\left(\frac{1+e^{-net_{pj}+\theta}}{1+e^{-net_{pj}+\theta}} - \frac{1}{1+e^{-net_{pj}+\theta}}\right) \tag{4-31}$$

$$= \left(\frac{1}{1+e^{-net_{pj}+\theta}}\right)\left(1 - \frac{1}{1+e^{-net_{pj}+\theta}}\right) \tag{4-32}$$

将式(4-26)代入式(4-32)可将 $f'(net_{pj})$表示为 O_{pj}的函数：

$$f'(net_{pj}) = O_{pj}(1-O_{pj}) \tag{4-33}$$

考虑式(4-14)、式(4-19)、式(4-25)和式(4-33)，可以写出在数字计算机上用 B－P 算法对网络训练所需的微分方程，其中误差函数是均方差函数且输出函数是 sigmoid 函数。从推导过程中我们可以看到，如选择另外的误差或兴奋函数，需要对上述公式进行修正。

综上所述，反向传播训练算法所需的微分方程是：

$$\Delta w_{ji} = \eta \delta_{pj} O_{pi} \tag{4-34}$$

其中，η 为学习速率，δ_{pj}为 L 层神经元 j 的误差信号，O_{pi}为 L－1 层神经元 i 的输出。

误差信号 δ_{pj}可表为：

对输出神经元　　$\delta_{pj} = (t_{pj} - O_{pj})O_{pj}(1-O_{pj})$　　(4-35)

对隐含层神经元　　$\delta_{pj} = O_{pj}(1-O_{pj})\sum_k \delta_{pk} w_{kj}$　　(4-36)

O_{pj}代表 L 层神经元 j 的输出，O_{pi}代表 L－1 层神经元 i 的输出，δ_{pk}代表 L＋1 层神经元 k 的误差信号。

真正的梯度下降是沿着梯度确定的方向以无穷小步长进行。很明显，这对于实现我们的目的是不切实际的，因此我们定义了学习速率 η(见式(4-34))。可以看到，式(4-34)确定了沿梯度方向的一个有限步长。这里 η 是常量，它相当于确定步长的增益。其中心思想就是选择足够大的 η，使得网络迅速收敛，而不会因调整过度面振荡。随后，我们将讨论共轭梯度技术，此技术实际上是利用梯度变化速率来确定步长。正确地理解学习速率的作用，能帮助我们合理选择它的大小。尽管如此，通常也需要一些实验调整以优化这一参数。

为清晰起见，式(4-34)、式(4-35)和式(4-36)的应用可见图 4.9。特别是当计算微分方程的各个元素时，该图有助于弄清网络的哪一层将参与计算。此图的上半部画出了输出层的训练，下半部描述了隐含层的训练。值得注意的是上面的微分方程仅对含有 sigmoid 兴奋函数(式(4-2)给出)的均方差有效。使用反向传播算法时，如使用另外的误差函数或兴奋函数时，就必须像推导中的那样，对相应微分方程进行修正。

实际上，在式(4-34)中经常加入一动量项，以便于在某些问题中加快网络的收敛速率。该动量考虑了过去权值变化的影响。动量常数 α 决定了这项的重要性。通过滤掉高频变量，该动量能使权值空间的误差表面平滑。当加入了动量项时，权值由下式调整：

$$\Delta w_{ji}(n+1) = \eta(\delta_{pj} O_{pi}) + \alpha \Delta w_{ji}(n) \tag{4-37}$$

有动量项的反向传播训练公式是纯梯度算法的变形之一，目的在于提高算法迅速收敛的能力。在这点上，我们注意到在反向传播算法中其变形有许多，并且绝大部分来自

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xidian.edu.cn

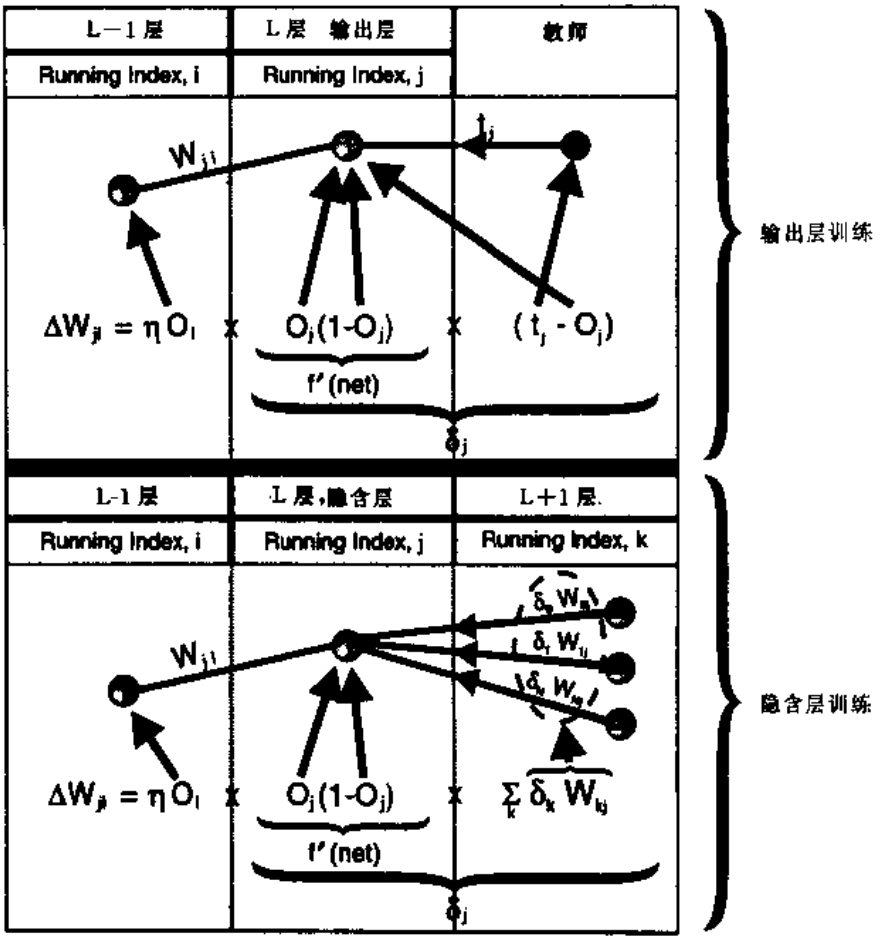


图 4.9 反向传播算法中的层间相互作用

于经验。

反向传播学习的全部过程,既包括它的前向路径也包括其反向路径,如图 4.10 所示。采用反向传播算法时,网络权值必须首先用一小随机值进行检查初始化。选择“小”初始权值是很重要的。若初始权值选择得太大,会导致网络不可训练。我们以后将讨论原因。初始化后,训练集矢量就可以被用于网络。使网络向前运行产生一实际值集合,利用反向传播可以建立一新权值集合,总误差经多次迭代后减小,如果不是这样,可以调整训练参数 α 和 η(矛盾数据,被与我们意图相反的期望数据所复制的训练矢量都将降低网络的收敛能力,在极困难的情况发生时,验证训练数据集合被证明是值得的)。

在定义训练集合中所有矢量的一个完整表达是一期间内。当权值接近某值使得网络总误差在这个期间低于以前建立的阈值,则称网络收敛。误差没有必要一致下降。网络总误差局部变化是正常的、需要的,特别在早期训练过程中。观察这一迭代函数误差曲线对得出收敛机理是很有用的。图 4.11 给出一个典型训练周期的收敛过程。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and cooperation: luzhenbo@yahoo.com.cn

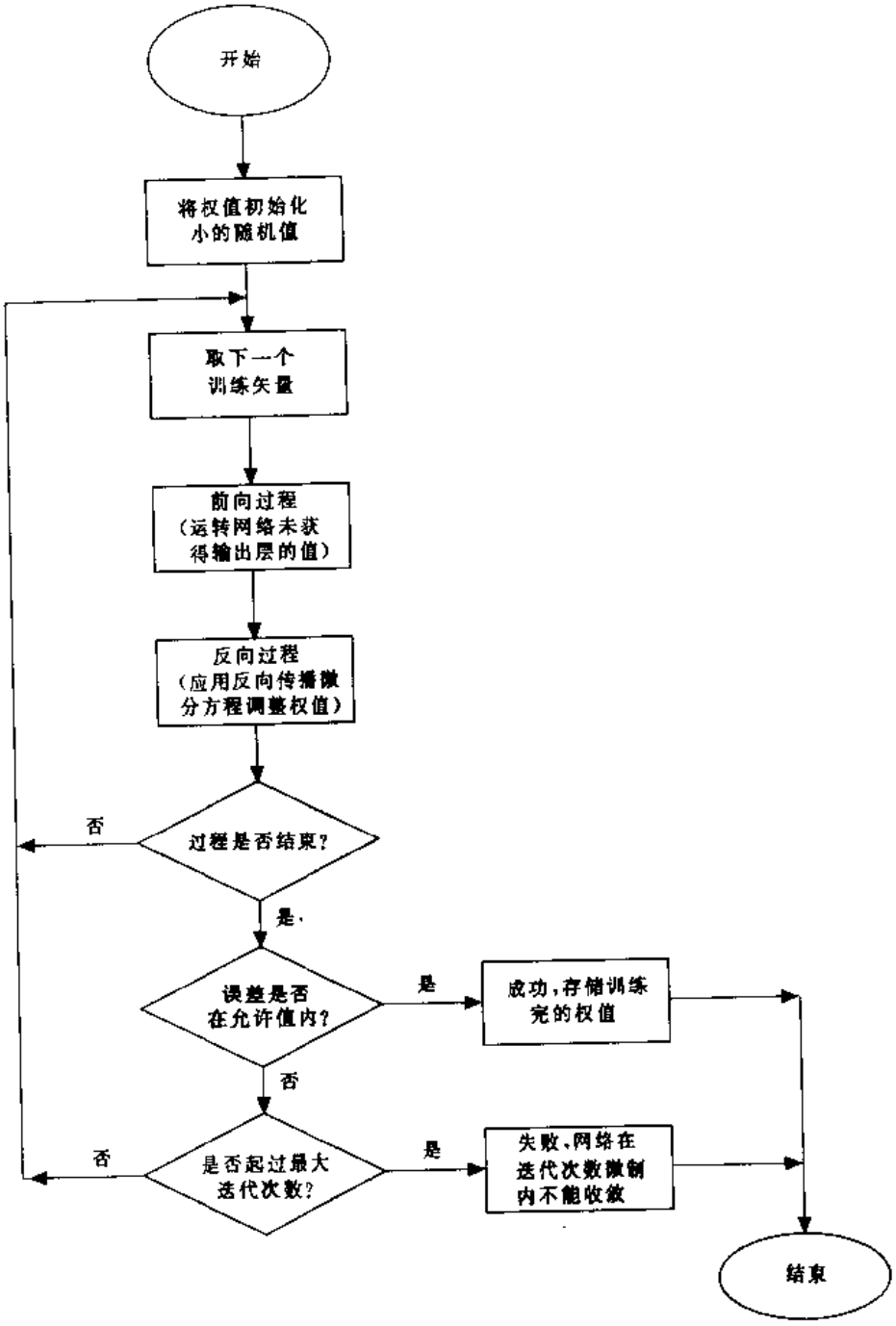


图 4.10 反向传播算法流程图

4.3.1 用 C++ 实现 B-P 算法

现在我们看一下实现反向传播算法的相关源程序。程序中对网络的描述比 4.2 节更为基本。这种简化可使我们集中精力于反向传播训练。细心的读者将观察到,以上网络需要在反向传播算法中有效地使用若干附加反向指针,这样,我们可避免干扰。由该程序计算的权值(清单 4.3),可以毫无困难地映射到 4.2 节中的网络上。

反向传播算法已被具体实现为其结构如下的单独一类。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

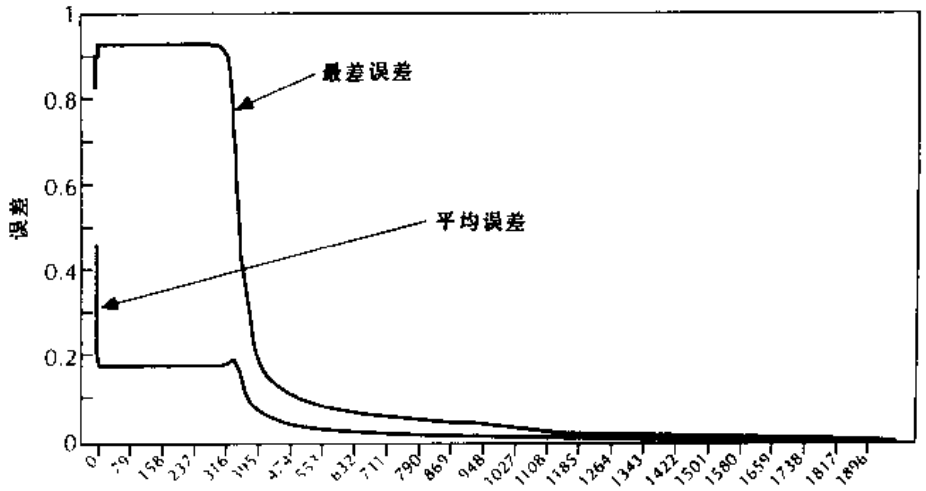


图 4.11 反向传播算法收敛曲线。最坏误差是整个时间轴中最大的误差。平均误差是整个时间轴的平均误差

清单 4.3

```
/**************************************************************************
*                                                                         *
*  0-0 BACK-PROP                                                          *
*                                                                         *
**************************************************************************/
// DEFINES

#define MAXLAYERS   4     // MAX NUMBER OF LAYERS IN NET (IN OUT & HIDDEN)
#define MAXNEURONS 120 // MAX NUMBER OF NEURONS PER LAYER
#define MAXPATTERNS 80  // MAX NUMBER OF PATTERNS IN A TRAINING SET
#define SIGMOID     0       // Choose Squashing fn
#define STEPFN      1       // Choose Squashing fn
#define ARCGRAN     1       // ARCHIVE GRANULARITY
#define NONE        0       // DO NOT ARCHIVE ERROR DATA
#define ALL         1       // ARCHIVE ALL ERROR DATA FOR EACH ITERATION
#define AVERAGE     2     // ARCHIVE AVERAGE ERROR DATA FOR EACH EPOCH
#define WORST       3       // ARCHIVE WORST ERROR FOR EACH EPOCH
#define TRUE        1
#define FALSE       0
#define SHUFFLE     0       // 0 TO DEFEAT RAND PRESENTATION ORDER
                          // 1 TO RANDOMIZE PRESENTATION OF INPUT VECTORS
//-------------------------------------------------------------------------

class BackProp {
private:
 double W[MAXLAYERS][MAXNEURONS][MAXNEURONS],       // WEIGHTS MATRIX
 double Wprev[MAXLAYERS][MAXNEURONS][MAXNEURONS]; // previous WEIGHTS
for momentum
 double Neuron[MAXLAYERS][MAXNEURONS];
 double DELTAj[MAXLAYERS][MAXNEURONS];
 double DELTA_Wij[MAXLAYERS][MAXNEURONS][MAXNEURONS];
 double ERROR[MAXNEURONS];
 double WorstErr;
 double AvgErr;
 double LastAvgErr;
 double LastWorstErr;
// Topology
 int NumLayers;                  // Number of layers
 int OutLayerIndx;               // array index of last layer
 int LayerSize[MAXLAYERS];  // Number of neurons in each layer
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
// Pattern data
 double InPattern[MAXPATTERNS][MAXNEURONS];   // Input values for each pattern
 double Desired[MAXPATTERNS][MAXNEURONS];     // desired value for each
                                              //  pattern/output
 unsigned int PatPresOrder[MAXPATTERNS];      // Order to present patterns to net
 unsigned long int CurrIter;     // Current iterations number.
 long int Epoch;
 int NumPatterns;                // Total patterns in training set
 int CurrPat;                    // Current pattern used in training
 int ConvCount;                  // The number of consecutive
                                 // patterns within tolerance
 int ConvergeFlg;                // Flag indicates convergence has
                                 // occurred
// PARMS
 double Temperature;             // For sigmoid
 double ETA;                     // Learning rate
 double ALPHA;                   // Momentum
 double ERRTOL;                  // min error tolerance required for convergence
 unsigned long int MAXITER;      // Max iterations to do before stopping

public:
 BackProp(void);
 void LoadTrainingSet(char *Fname);
 void  GetInputs(void);
 void  RunNet(void);
 void  GetParms(char *);
 void  GetWeights(void);
 double Sigmoid(double Net, double Tempr);
 void  SetRandomWeights(void);
 double HCalcDelta(int lyr,int j,double dNETj);
 int   CalcErrors(void);
 void  AdaptWeights(void);
 void  SaveWeights(char *);
 int   train(char *, char *);
 long int QueryEpoch(){return Epoch;}
 double QueryTemparature(){return Temperature;}
 double QueryEta(){return ETA;}
 double QueryAlpha(){return ALPHA;}
};
// MISC FLAGS
int ArchOn  = AVERAGE;          // Set to 1= ALL, NONE, AVERAGE or WORST to control
                                //  data sent to ARCHIVE file for graphical
                                //  analysis
FILE *ARCHIVE;                  // Archive training sequence
```

现在我们看一下这一算法的实现。只有 **train** 方法需要被用来训练网络。**RunNet** 完成前向路径。**AdaptWeights** 和 **HCalcDelfa** 共同完成了后向路径。注意，通过详细说明定义 **SHUFFLE**，我们能选择是采用随机模式，还是选择初始顺序模式。收敛判据要求，所有模式的均方差在一期间内小于容许的误差。

清单 4.4

```
//---------------------------------------------------------------------

// METHOD DEFINITIONS

BackProp::BackProp() {
 LastAvgErr= 1.0;
 LastWorstErr=1.0;
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xinhuanet.com.cn


```
 WorstErr = 0.0;
 AvgErr  = 0.0;
 CurrIter=0;
 Epoch=0;
 CurrPat = -1;                          // Current pattern used in training
 ConvCount=0;
 ConvergeFlg=FALSE;                     // Flag indicates convergence
}

void BackProp::LoadTrainingSet(char *Fname) {
FILE *PFILE;
int PGindx;
int x,mask;
int pat,i,j;
double inVal;
int NumPatBytes;

PFILE = fopen(Fname,"r");               // batch
if (PFILE==NULL){
  printf("\nUnable to open file \n");
  exit(0);
  }
fscanf(PFILE,"%d",&NumPatterns);
NumPatBytes= LayerSize[0] / 8;          // # of Input lyr neurons must be divisible by 8
for (pat=0; pat<NumPatterns; pat++) {
  PGindx=0;
  for (i=0; i<NumPatBytes; i++) {
    fscanf(PFILE,"%x",&x);
    mask = 0x80;
    for (j=0; j<8; j++) {
      if ((mask & x) > 0) {
        InPattern[pat][PGindx]=1.0;
        }
       else {
        InPattern[pat][PGindx]=0.0;
        } /* endif */
      mask=mask/2; //printf("(%x)",mask);
      PGindx++;
      } /* endfor */
    } /* endfor */
  // Now get desired / expected values
  for (i=0; i<LayerSize[OutLayerIndx]; i++) {
    fscanf(PFILE,"%lf",&inVal);
    Desired[pat][i]=inVal;
    } /* endfor */
  } /* endfor */
fclose(PFILE);
//init pattern presentation order
for (i=0; i<NumPatterns; i++) {
  PatPresOrder[i]=i;                    // Start with unmodified order
  } /* endfor */
}
/*****************************************************************************
* Function GetParms                                                          *
*    Loads topology from file spec'd file if avail.          *    otherwise tries
DEFAULT.PRM                             *
*
*****************************************************************************/
void BackProp::GetParms(char *PrmFileName){
FILE *PRMFILE;
PRMFILE = fopen(PrmFileName,"r");       // batch
if (PRMFILE==NULL){
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
  printf("\nUnable to open Parameter file: %s \n",PrmFileName);
  printf("\nAttempting to open default file: PARMS1");
  PRMFILE = fopen("PARMS1","r");
  if (PRMFILE==NULL){
    printf("\nUnable to open Parameter file\n");
     exit(0);
     }
  }
printf("\nLoading Parameters from file: %s\n",PrmFileName);
fscanf(PRMFILE,"%lf",&Temperature);
fscanf(PRMFILE,"%lf",&ETA);
fscanf(PRMFILE,"%lf",&ALPHA);
fscanf(PRMFILE,"%ld",&MAXITER);
fscanf(PRMFILE,"%lf",&ERRTOL);
fscanf(PRMFILE,"%d",&NumLayers);
printf("\nNumber of layers=%d\n",NumLayers);
for (int i=0; i<NumLayers; i++) {
  fscanf(PRMFILE,"%d",&LayerSize[i]);
  printf("Number of neurons in layer %d = %d\n",i,LayerSize[i]);
   }
OutLayerIndx = NumLayers-1;              // accommodate 0 org'd arrays
fclose(PRMFILE);
}

/*************************************************************************
* FUNCTION GetWeights                                                    *
*    Loads a set of previously stored weights from file                  *
     (Weights must match current net parms.  No advance                  *
     error checking or validation is done)                               *
*************************************************************************/
void BackProp::GetWeights(){
FILE *WTFILE;
char WtFileName[30];
int i,j,k;
double zWT;

WTFILE = fopen("DEFAULT.WGT","r");
if (WTFILE==NULL){
  printf("Unable to open weight file");
  exit(0);
  }
for (i=0; i<NumLayers-1; i++) {
  for (k=0; k<LayerSize[i+1]; k++) {
    for (j=0; j<=LayerSize[i]; j++) {          // One extra for bias neuron
        fscanf(WTFILE,"%lf",&zWT);             // read weights from file
        W[i][j][k] =zWT;
       }
    }
  }
fclose(WTFILE);
}

void BackProp::SetRandomWeights(){
int i,j,k;
double zWT;
  //randomize();
  srand(6);
  for (i=1; i<NumLayers; i++) {
    for (k=0; k<LayerSize[i]; k++) {
      for (j=0; j<=LayerSize[i-1]; j++) {      // One extra for bias neuron
        zWT=(double)rand();
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
            zWT=zWT/2.0;
            W[i][j][k] =zWT/65536.0;          // random weight normalized to [0,0.2]
            Wprev[i][j][k]=0;;
            }
          }
        }
}

/*****************************************************************************
* GetInputs                                                                  *
*  Loads training vectors into input layer neurons                           *
*  Keeps track of itsration and epoch                                        *
*****************************************************************************/
void BackProp::GetInputs(){
int i,j,k,a,b,c;
unsigned int PatID;
CurrIter++;
CurrPat++;                                // Update the current pattern
if (CurrPat>=NumPatterns){
  CurrPat=0;
  Epoch++;
  }
if ((CurrPat==0) &&(SHUFFLE>0)) {         // shuffle

  for (j=0; j<=5; j++) {
    a=NumPatterns*rand()/32767;
    b=NumPatterns*rand()/32767;
    c=PatPresOrder[a];
    PatPresOrder[a]=PatPresOrder[b];
    PatPresOrder[b]=c;
    } /* endfor */
    if ( 100*(Epoch/100) ==Epoch) {
      printf("Epoch:%d",Epoch);
      printf(" Worst=%lf Avg=%lf\n",LastWorstErr, LastAvgErr);
       }
  } /* endif */
PatID=PatPresOrder[CurrPat];
 for (i=0; i<LayerSize[0]; i++) {
   Neuron[0][i]=InPattern[PatID][i];          // Show it to the neurons
    }
}

/*****************************************************************************
*
* FUNCTION RunNet                                                            *
*   Back-Propagations forward pass
*    Calculates outputs for all network neurons                              *
*****************************************************************************/
void BackProp::RunNet(){
int lyr;   // layer to calculate
int dNeuron; // dest layer neuron
int sNeuron; // src layer neuron
double SumNet;
double out;
for (lyr=1; lyr<NumLayers; lyr++) {
  Neuron[lyr-1][LayerSize[lyr-1]]=1.0;       //force bias neuron output to 1.0
  for (dNeuron=0; dNeuron<LayerSize[lyr]; dNeuron++) {
     SumNet=0.0;
     for (sNeuron=0; sNeuron <= LayerSize[lyr-1]; sNeuron++) {              //add 1 for bias
      SumNet += Neuron[lyr-1][sNeuron] * W[lyr][sNeuron][dNeuron];
       }
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
    out=Sigmoid(SumNet,Temperature);
    Neuron[lyr][dNeuron] = out;
     }
   }
}

/*******************************************************************************
* HCalcDelta
*   Calculate backward error signal for hidden nodes
*******************************************************************************/

double BackProp::HCalcDelta(int lyr,int j,double dNETj){
int k;
double Delta, SUMk;
SUMk=0.0;
for (k=0; k<LayerSize[lyr+1]; k++) {
  SUMk += DELTAj[lyr+1][k] * W[lyr+1][j][k];
  } /* endfor */
Delta = dNETj * SUMk;
return Delta;
}

/*******************************************************************************
* CalcErrors                                                                   *
*   Determines convergence & archives convergence behavior                     *
*                                                                              *
*******************************************************************************/
int BackProp::CalcErrors(){
double dNETj;
double MOMENTUM;
int i, j;
int LocalConvFlg=TRUE;
unsigned int PatID;
PatID = PatPresOrder[CurrPat];
for (j=0; j<LayerSize[OutLayerIndx]; j++) {
  ERROR[j] = (Desired[PatID][j] - Neuron[OutLayerIndx][j]);
  if (fabs(ERROR[j])>=ERRTOL) LocalConvFlg=FALSE;   // Any Nonconverged error kills
  AvgErr +=fabs(ERROR[j]);
  if (fabs(ERROR[j])  > WorstErr) WorstErr= fabs(ERROR[j]);
  if (CurrPat==(NumPatterns-1)) {
    AvgErr=AvgErr/NumPatterns;
    if (ArchOn==AVERAGE) {
      if ((ARCGRAN*(Epoch/ARCGRAN)) == Epoch) {   // only save ARCGRANth epochs
        fprintf(ARCHIVE,"%ld %lf %lf\n",Epoch,AvgErr, fabs(WorstErr) );
        } /* endif */
      }
    if (ArchOn==WORST) {
      if ((ARCGRAN*(Epoch/ARCGRAN)) == Epoch) {         // only save alternate epochs
        fprintf(ARCHIVE,"%ld %lf\n",Epoch, fabs(WorstErr) );
        } /* endif */
      }
    LastAvgErr=  AvgErr;
    LastWorstErr=WorstErr;
    AvgErr=0.0;
    WorstErr=0.0;
    }
   else {
    } /* endif */
  if (ArchOn==ALL) fprintf(ARCHIVE,"%ld %lf\n",CurrIter,fabs(ERROR[j]) );
  } /* endfor */
 if (LocalConvFlg) {
  ConvCount++; //Record that another consec pattern is within ERRTOL
  if (ConvCount==2*NumPatterns) {
    ConvergeFlg=TRUE;
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xaut.edu.cn


```
    printf("Epoch:%d",Epoch);
    printf(" Worst=%lf Avg=%lf\n",LastWorstErr, LastAvgErr);
    } /* endif */
  }
  else {
  ConvCount=0; //Start over. This pattern had an error out of tolerance
  } /* endif */
return(ConvergeFlg);
}
/*****************************************************************************
* AdaptWeights
*    Back-propagations backward pass
*****************************************************************************/
void BackProp::AdaptWeights(){
double dNETj;
double MOMENTUM;
int lyr, i, j;
unsigned int PatID;
PatID = PatPresOrder[CurrPat];
for (lyr=OutLayerIndx; lyr>0; lyr--) {
  for (j=0; j<LayerSize[lyr]; j++) {
    dNETj=Neuron[lyr][j] * (1 - Neuron[lyr][j]);
    if (lyr==OutLayerIndx) {
      ERROR[j] = (Desired[PatID][j] - Neuron[lyr][j]);
      DELTAj[lyr][j] = ERROR[j] * dNETj;
      }
     else {
      DELTAj[lyr][j] = HCalcDelta(lyr,j,dNETj);
      }
    } /* endfor */
  } /* endfor */
for (lyr=OutLayerIndx; lyr>0; lyr--) {
  for (j=0; j<LayerSize[lyr]; j++) {
    for (i=0; i<=LayerSize[lyr-1]; i++) {  // include bias
      DELTA_Wij[lyr][i][j] = ETA * DELTAj[lyr][j] * Neuron[lyr-1][i];
      MOMENTUM= ALPHA*(W[lyr][i][j] - Wprev[lyr][i][j]);
      Wprev[lyr][i][j]=W[lyr][i][j];
      W[lyr][i][j] = W[lyr][i][j]+ DELTA_Wij[lyr][i][j] + MOMENTUM;
      } /* endfor */
    } /* endfor */
  } /* endfor */
}

/*****************************************************************************
*                                                                            *
* SaveWeights                                                                *
*                                                                            *
*****************************************************************************/
void  BackProp::SaveWeights(char *WgtName) {
int lyr,s,d;
double zWT;
FILE *WEIGHTFILE;
  WEIGHTFILE = fopen(WgtName,"w");
  if (WEIGHTFILE==NULL){
    printf("Unable to open weight file for output:%s\n",WgtName);
    exit(0);
    }
  printf("SAVING CALCULATED WEIGHTS:\n\n");
  fprintf(WEIGHTFILE,"0.00\n");                    // Threshold always 0
  fprintf(WEIGHTFILE,"%lf\n",Temperature);         // Temperature
  fprintf(WEIGHTFILE,"%d\n",NumLayers);            // Number of layers
  for (lyr=0; lyr<NumLayers; lyr++) {              // Save topology
    fprintf(WEIGHTFILE,"%d\n",LayerSize[lyr]);     // Number of neurons/layer
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
    }
  for (lyr=1; lyr<NumLayers; lyr++) {                    // Start at 1st hidden
    for (d=0; d<LayerSize[lyr]; d++) {
      for (s=0; s<=LayerSize[lyr-1]; s++) {              // One extra for bias
        zWT=W[lyr][s][d];
        fprintf(WEIGHTFILE,"%lf\n",zWT);
        }
      }
    }
  fclose(WEIGHTFILE);
}

/*****************************************************************************
* Sigmoid                                                                    *
*   This is the activation function                                          *
*                                                                            *
*****************************************************************************/
double BackProp::Sigmoid(double Net, double Tempr){
return 1.0/(1.0 + exp(-Net/Tempr));
}
/*****************************************************************************
* train                                                                      *
*   Top level control to train the network with back-propagation             *
*                                                                            *
*****************************************************************************/
int   BackProp::train(char *TrnFname, char *ParmFname){
if (ArchOn) ARCHIVE=fopen("archive.lst","w");
GetParms(ParmFname);
LoadTrainingSet(TrnFname);
SetRandomWeights();
int Converged=0;
while ((!Converged) && (CurrIter < MAXITER) ) {
  GetInputs();
  RunNet();
  Converged=CalcErrors();
  AdaptWeights();
  }
if (ArchOn) fclose(ARCHIVE);
return Converged;
}
```

最后，下述程序给出了类 **BackProp** 在一个模式程序中的应用。

清单 4.5

```
/*****************************************************************************
*                                                                            *
* FUNCTION ShowResults                                                       **
*                                                                            *
*****************************************************************************/

void ShowResults(int ConvergeFlg, long int CurrEpoch,
          double temp, double alpha, double eta){
printf("\n-------------------------------------------------------------\n");
if (ConvergeFlg) {
  printf("SUCCESS: Convergence has occured at iteration %ld\n",CurrEpoch);
  }
 else {
  printf("FAILURE: Convergence has NOT occured!!\n");
  } // endif
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
printf("Temperature = %lf\n",temp);
printf(" ETA  = %lf\n Alpha= %lf\n",eta,alpha);
printf("\n-----------------------------------------------------------\n");
}

BackProp BPnet;

int main(int argc, char *argv[]) {
int Converged;
if (argc>3) {
  Converged = BPnet.train(argv[1],argv[2]);
  }
 else {
  printf("USAGE: BPROP TRAINING_FILE PARMS_FILE WEIGHT_FILE\n");
  exit(0);
  }
// Show how we did
ShowResults(Converged,BPnet.QueryEpoch(),
       BPnet.QueryTemperature(),BPnet.QueryAlpha(),BPnet.QueryEta());
if (Converged)
  BPnet.SaveWeights(argv[3]);                    // Save the weights for later use
return 0;
}
```

对于被训练的网络,我们已建立起一套方法来获得权值。现在,只剩下如何使用这些方法。在下一节中,基于目前为止我们所做的工作,将提出一非常基本的相互作用的字符识别系统。更典型的是可用数据能被分成用于反向传播算法(或另一个训练算法)的训练集合和一个不参加训练过程的测试集合。总的来说,对于测试集合,最突出的是它具有识别器性能。在十三章中,我们将讨论另外的方法(特别是可信和混淆矩阵)由此而来判断一个神经识别系统的全部优点。现在,一个简单直观的量度是精确度:

$$\text{精确度} = \text{正确的分类数}/\text{总模式数} \tag{4-38}$$

与我们努力获得的权值有关的其它问题,要通过图形方式来研究它们。一个最常用的以图形表达训练权值的方法是利用 Hinton 图。为了弄清怎样获得和解释这些图,让我们考虑一下图 4.12 中给出的网络。

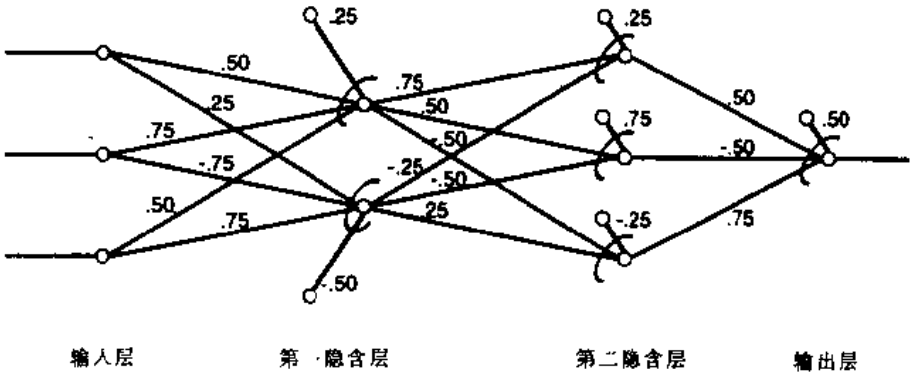


图 4.12 Hinton 网络

与此对应的 Hinton 图如图 4.13。

图中的权值幅度大小,对应于正方形的大小;权值的符号对应于阴影。黑色正方形代表正

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and cooperation: luzhenbo@yahoo.com.cn

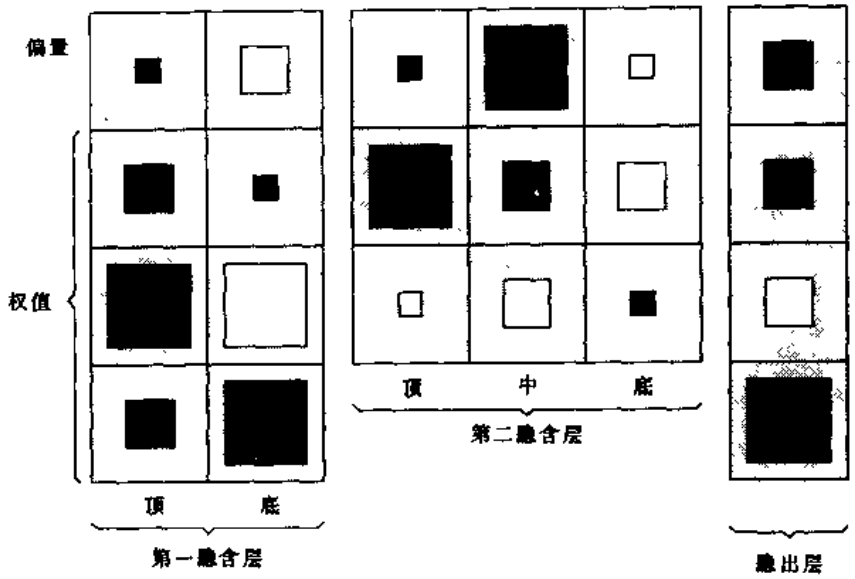


图 4.13 Hinton 图

权值,白色正方形代表负权值。注意,图中每一个神经元表示为一列,权值是加在神经元输入上的。这样,输入神经元不出现在图中,因为它们没有这样的输入权值。

4.4 一个基本例子

对模式识别问题的最简单和最明显的方法,可能是提供一个前馈网络。用带有图像的扫描点映射组成的矢量,由反向传播算法训练这个前馈网络。在随后的特征提取讨论中(见第六章、第七章),我们将描述几个更复杂的技术。这些技术用于把变换点输入空间映射到另一类描述空间,这类描述空间对于神经识别系统具有更加满意的特征。然而,尽管存在其它的或许更有效的技术,原始点映射输入已被成功地采用。对于某些给定的应用,当计算太复繁时,也可考虑使用该技术。网络中的点映射图像描述,见图 4.14。

这些直接使用点映射作为输入量的方法,可见有关神经网络的文献[Baker and McCartor, 1991]。然而更经常的是把它们作为一个基准与其它更复杂的方法相比较。

为了得到感性认识,我们给出一种方法[Zurada 等,1991]。在其应用中,用前馈神经网络对打印字符进行分类。更具体地说,字符用 7×10 象素点(来自一标准的点阵打印机)来描述。因为直接将 7×10 象素点映射输入到网络,所以输入层需要 70 个神经元。在训练集上,从训练的有效性和精确度角度考虑。对于被给问题域,这一研究评价了若干网络拓扑结构。对于独立的测试集目前仍没有结果。但评估了输出层的两个变形:

1. 96 个神经元,每一个代表一个独立的数字,大写和小写 α 及标点符号
2. 共同表示字符的 ASCII 码编码的 8 个神经元

考虑了有一个或 2 个隐层的网络。在隐层数范围从 25 到 80。对于每层都含有 70 个神经元的 2 隐层拓扑,得到了最好的分类结果。

对于这部分内容的 C 语言实现,我们采取不同的方法,即识别器系统对于它识别的 10 个数字的每一个都有一个网络。也就是,我们使用了较少的隐神经元,但利用了更多的网络。所以,我们能了解在这种结构下网络是如何工作的。让我们先看一下识别器类定义。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

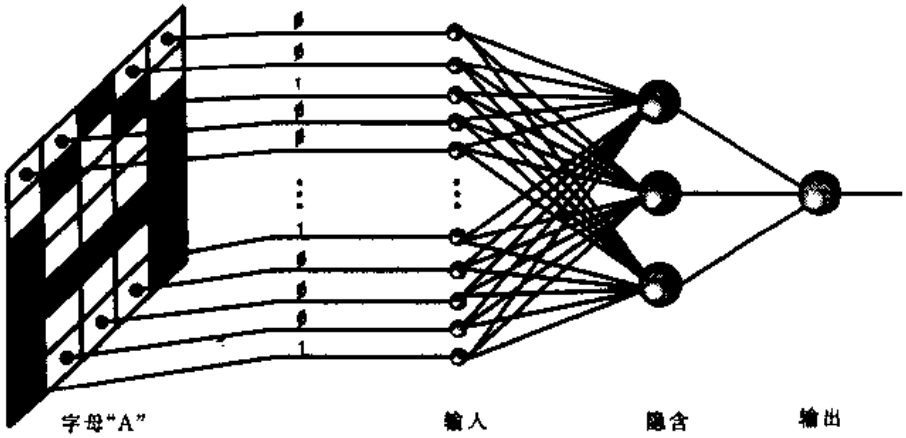


图 4.14 字符位图"A"馈入网络

清单 4.6

```
//*************************************************************************
// THE RECOLIST CLASS                                                     *
//*************************************************************************
struct RecoDat {
double Val;
int    NetID;
char   digit;
RecoDat *next;
};

class RecoList {
private:
 RecoDat     *head;
public:
 RecoList(){head = (RecoDat *)NULL;}
 void kill();                              //delete entire list
 void AddSorted(double, int);              //Add value + net id to list
 RecoDat QueryNth(int);                    //Get nth element in sorted list
 ~RecoList();                              //destructor
};

//*************************************************************************
// THE RECOSYS CLASS                                                      *
//*************************************************************************
class RECOSYS {
private:

  NETWORK     *net;
  RecoList    rList;
public:
 RECOSYS(void);
 int   Setup(char *);                      // parm is filename of file with
                                           // weight filenames for all networks
 void  ApplyVector(unsigned char *, int);  // same vect for -> all nets
 void  RunReco(void);                      // run all nets, sort results
 int   QueryNth(int, NETRESULTS *);        // results from Nth net
 int   QueryNthBest(int, NETRESULTS *);    // results from Nth best net
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
double QueryTemp() {return net->RequestTemp();}
double QueryThresh() {return net->RequestThresh();}
int   QueryLayerSize(int l){return net->RequestLayerSize(l);}
int   QueryNetCount() { return 10; }
int   QueryAlive() { return net->GetAlive(); }
};
```

从清单 4.6 中看出 **RECOSYS** 类由两部分组成。第一并且最重要的是一个 **NETWORK** 目标连接表(见图 4.15)。表中的每一个 **NETWORK** 目标专门识别一个特定数字,因此仅有单个输出。**ApplyVector** 把相同的输入矢量装入每一个 **NETWORK** 目标。RunReco 用表中每个 **NETWORK** 目标去计算输入模式。**RECOSYS** 的第二部分是 **RecoList** 目标,它的作用是为每个被存储在激活区域的 **NETWORK** 结果提供连接。

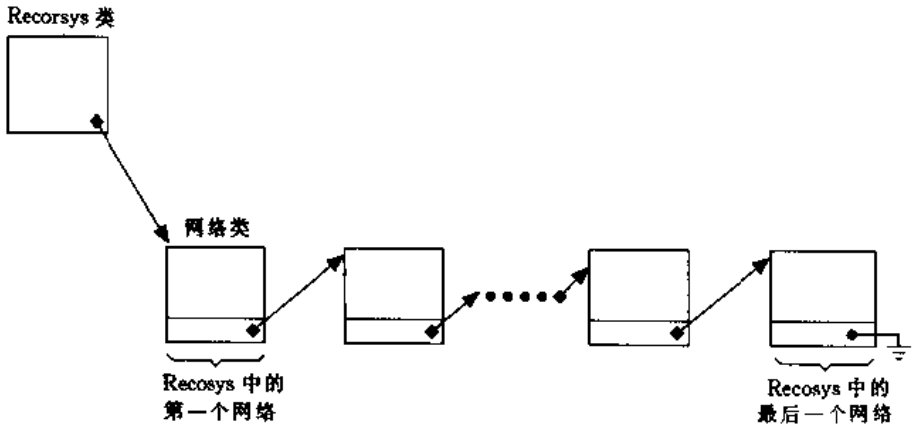


图 4.15 识别器的类

类 **Recosys** 和 **RecoList** 方法的实现见清单 4.7。

清单 4.7

```
//*************************************************************************
// METHODS FOR CLASS RECOLIST                                             *
//*************************************************************************
void RecoList::kill() {                          //delete entire liet
RecoDat *p1;
RecoDat *p2;
p1=head;
while (p1) {
  p2=p1;
  delete p1;
  p1=p2->next;
  } /* endwhile */
head = (RecoDat *)NULL;
}

void RecoList::AddSorted(double V, int id) {     //Add value + net id tp list
  RecoDat *Itm;
  RecoDat *cur;
  RecoDat *prev;
  int got1;
Itm = new RecoDat;
Itm->Val=V;
Itm->NetID=id;
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo2@163.com


```
ltm->next=(RecoDat *)NULL;
ltm->digit=id+0x30;
if (head) {
  if (V > head->Val) {
    ltm->next=head;                          // add as 1st item on list
    head = ltm;
    }
   else {
    cur=head->next;
    prev=head;
    got1 =0;
    while (cur && !got1) {
      // if we find 1 here its in the middle
      if (V > cur->Val) {
        got1=1;                              // found the spot...Add BEFORE cur
        prev->next=ltm;
        ltm->next=cur;
        }
       else {
        prev=cur;
        cur=cur->next;
        } /* endif */
      } /* endwhile */
      if (!got1) {
        // add at end using prev
        prev->next=ltm;
        } /* endif */
    } /* endif */
  }
   else {
    head=ltm;                                // add as only item on list
    } /* endif */
  }

  RecoDat RecoList::QueryNth(int n) {        //Get nth element in sorted list
    RecoDat *cur;
    RecoDat rv;
    int found,cnt;
  rv.Val=0;
  rv.NetID=-1;
  rv.digit='*';
  cur=head;
  found =0;
  cnt=0;
  while (cur && !found) {
    if (n==cnt) {
      found=1;
      rv.Val=  cur->Val;
      rv.NetID= cur->NetID;
      rv.digit= cur->digit; }
    else {
      cnt++;
      cur=cur->next;
      } /* endif */
    } /* endwhile */
  return rv;
  }

  RecoList::~RecoList() {                    //destructor
  kill();
  }
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and cooperation: luzhenbo@yahoo.com.cn


```
//*************************************************************************
// METHODS FOR CLASS RECOSYS                                              *
//*************************************************************************
RECOSYS::RECOSYS() {
  int i;
  NETWORK *N;
for (i=0; i<10; i++) {                          // create 10 generic networks
N = new NETWORK;
if (net) {
  N->SetNext(net);                              // point to old 1st net
  net=N;
  }
 else {
  net=N;
  } /* endif */
 } /* endfor */
}

int RECOSYS::Setup(char *Zname) {
  FILE *WFL;                                    //weight file list pointer
  char Wname[40];
  int i;
  NETWORK *N;
  int rv=0;
if((WFL = fopen(Zname, "r")) == NULL) return 1;
N=net;                                          // set equal to head
for (i=0; i<10; i++) {                          //init each of the nets
  fscanf(WFL,"%s",Wname);
  if (N->Setup(Wname)) rv=1;
  N=N->GetNext();
  } /* endfor */
return rv;
}

void RECOSYS::ApplyVector(unsigned char *Vect, int Sz) {
  int i;
  NETWORK *N;
N=net;                                          // set equal to head
for (i=0; i<10; i++) {                          // send same vector to each of the nets
  N->ApplyVector(Vect, Sz);
  N=N->GetNext();
  } /* endfor */
}

void RECOSYS::RunReco(void) {
  int i;
  NETWORK *N;
  NETRESULTS results;
N=net;                                          // set equal to head
for (i=0; i<10; i++) {                          // run each of the nets successively
  N->RunNetwork();
  N=N->GetNext();
  } /* endfor */
//BUILD THE SORTED LIST HERE!!!!
//BUT 1st INIT LIST TO EMPTY
rList.kill();
N=net;                                          // reset to head
for (i=0; i<10; i++) {
 N->RequestNthOutNeuron( 0, &results);
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
 rList.AddSorted(results.value, i);
  N=N->GetNext();
  } /* endfor */
}
int RECOSYS::QueryNth(int n, NETRESULTS *rv)  {
   int i;
   NETWORK *Nptr;
   NETRESULTS results;
rv->value   = 0.0;
rv->index   = -1;
rv->character= '*';
if (n>9) return 1;                                  // out of bounds
Nptr=net;                                           // set equal to head
for (i=0; i<n; i++) {
  Nptr=Nptr->GetNext();
  } /* endfor */
if (!Nptr->RequestNthOutNeuron(0, &results)) {
  rv->value   = results.value;
  rv->index   = n;
  rv->character= n+0x30;
   return 0;
   }
return 2;                                           // Nth net not found
}

int RECOSYS::QueryNthBest(int n, NETRESULTS *rv) {
  RecoDat RD=rList.QueryNth(n);
  rv->value = RD.Val;
  rv->index = RD.NetID;
  rv->character = RD.digit;
return 0;
}
```

Recosys方法 **ApplyVector** 产生一输入矢量，它被装到所有网络成员的输入层。对于recosys 方法的一个参照 **RunReco** 使所有成员网络的运行。网络对输入矢量的运行结果，按兴奋水平进行分类。使用 **QueryNthBest** 方法可以得到这些结果。兴奋水平最高的将取胜。对于识别器系统和对视图的用户接口完成了源码，磁盘上提供(GUI)程序。此外，磁盘上还提供用于 8 隐单元和 16 隐单元的两套预训练权值。

附盘提供了带有图形用户界面的识别器类程序。这个程序可帮助我们用前面介绍过的网络实现，来做识别器类实验。现在我们简单描述一下这个程序界面的操作。要执行此程序，只要在任何 OS/2 提示符下键入 GRID1 便可。图 4.16 给出了完成所识别一个字符，所需所有的步骤之后的屏幕显示。

活动菜单选择具有自我解释特点。选择 **LOAD.WEIGHTS**，网络权值便从已存好的权值集合中装载。**LOAD.WEIGHTS** 给出一包含当前目录下存在权值文件的对话框。程序完成后，**NETWORK PARAMETES** 窗口将显示网络拓扑和温度系数。

一旦权值按如上所述建立起来，下一步装入要识别图像的点映射描述。从 ***CHARACTER GRID MAP*** 窗口中的 **EDIT**，可以装载所有的大小写字符及数字样本图像。这些字符图像与在训练集中提供的字符图像是一样的。用鼠标的左上钮击网络中的某一区域，可以关闭或打开某一象素。以这种方法，样本字符可被修改成任何所需要的形式。EDIT 下的另一操作是清除网格，建立与样本无关的模式。***CHARATER BRID MAP*** 窗口选项，还提供网络前景和背景颜色的选择。

即然我们已建立了可用的权值集合和被识别的点映射，所有余下的就是通知识别系统去

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

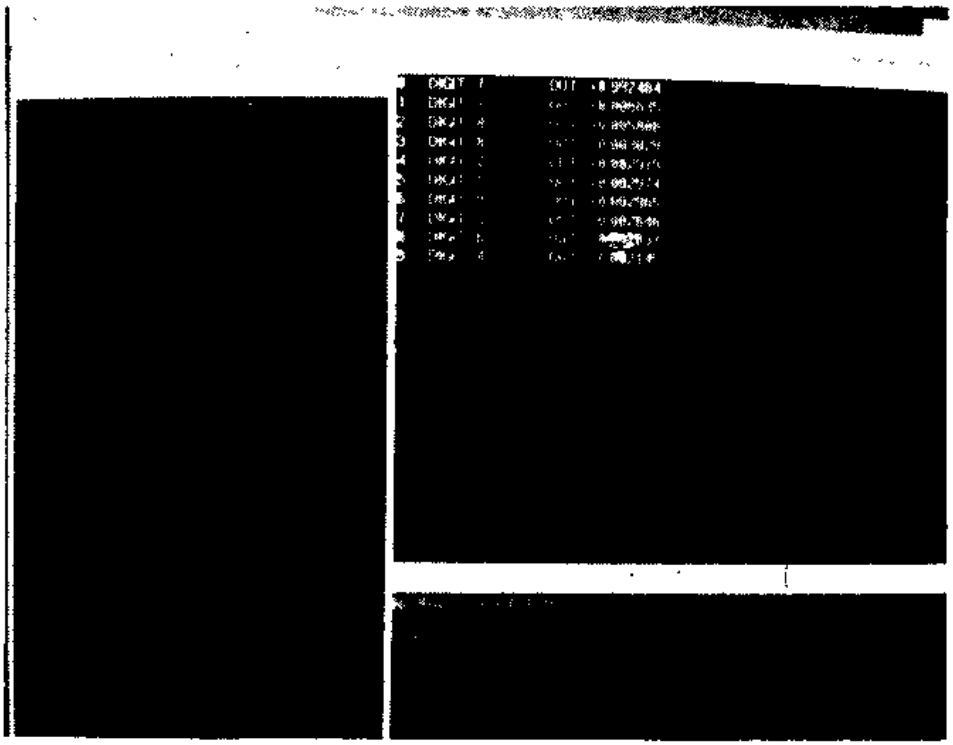


图 4.16 识别器识别后的屏幕

识别它。这将从上滚条选择 RUN RECOGNIZER 来完成。识别结果呈现在 *NETWORK-ALL OUTPUTS* 窗口上,并由激活水平存储。

文献中讨论大量的同多层感知网络相关用反向传播算法训练网络的问题。尽管这些问题对模式识别广泛研究课题来说不是很重要,然而,它们对其在真正意义下的实现仍很重要。这样的问题包括:

1. 网络训练参数和策略的选择
2. 合适网络拓扑的选择
3. 重叠训练问题
4. 训练集合的特征和大小

4.5 训练策略和避免局部最小

训练参数的选择和避免局部最小,在某种情况是精细的技术。目前如何选择最优参数还没有标准。收敛缓慢或发散是反向传播算法面临的问题之一。除了局部最小的问题,另外一个困难有时称错误最小问题。当误差空间曲面包含一扩展的近零梯度平坦平面时(见图 4.8),这一问题可能发生。当这个平面足够长时,收敛可能非常缓慢。如果误差曲面足够平坦,舍入误差将是主要问题,它可能把梯度下降算法引向一个完全谬误的方向。

未成熟饱和是反向传播算法易产生的另一问题。当神经元超出了它们的线性范围工作时,就会发生未成熟饱和。因为这些神经元在整个训练过程中不能被有效地训练,网络误差将保持很大。其原因为,当与给定目的神经元相关的权值增加到足够大时,神经元工作在兴奋函数渐近它的极限值(sigmoid=1 或 0 时)的区域内。在这个区域内,微分 $f'(net)$ 值非常小。由

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

式(4-14)、式(4-19)和式(4-25),很明显看出,当 $f'(net)$ 趋于零时,通过反向传播算法完成的权值调整几乎也趋于零,因此导致训练无效。我们说这样的神经元处于饱和状态。如果初始权值值选择太大,也可发生未成熟饱和。同时,训练选择参数 α 和 β 选择不当,也能导致这种饱和。我们也注意了像 ALOPEX 训练算法(见 4.2 节),它不依赖于对兴奋函数求导数,所以不会出现这个问题。

在训练技术中有一类方式,其重点放在把训练数据提供给网络。这样的训练被称为 Snowball 训练[Wang and Jean,1993]。它包括首先用有正样本表示网络,以这个样本建立一有利于它们的权值集合。逐渐地,随着训练的进行,把负的或相反的样本加到训练集合中。其它技术包括,在训练中及当确认对所提供全部类训练矢量数几乎相等时,以随机顺序表示训练模式。

对于管理使反向传播算法收敛得更迅速的训练数据描述,是较复杂的技术。此方法包含削减被称为"多余"的训练模式。多余模式是那些在学习过程期间不能提供给网络最新权值的模式。输入数据被分成两类:即学习集合和空闲集合(每一个这样的数据都被称为过剩数据)。一旦注明了一个空闲集成员,那么删减就用每隔 N 个时间延迟提供给网络对应矢量来完成。已有报道,利用此技术,可减小 2/3 的训练时间[Tsay 等,1992]。

另一类训练策略包含网络学习速率 η 的使用。前馈网络中的后层有较大的局部梯度趋势,这样它也就具有成为更快的学习者的趋势。对于某给定层,神经元或权值确定学习速率是容易做到的。把学习速率嵌入层内,神经元或权值在本区域是必要的。因此,很自然的是让早出现层(离输入层靠近的层)具有一较大学习速率。另外,对多输入神经元,减少学习速率更有利[Haykin,1994]。其实现是相同的。

这样我们看到学习速率的修改有益于网络训练。让我们沿着这一主线进一步研究和考虑学习速率的调整。

学习速率的适应调整,是从一个递归到下一个递归的递推过程。反向传播算法收敛减慢的可能原因之一,是在一给定的误差表面区域中,给定的权值维数不能接受单一的学习速率。对误差表面上一已知点,某一个权值维数可能急剧下降,同时,另一个却保持某一相当的值。当然,所有这些在下一次递归中要变化。为了处理这些问题,常用下面的概念。如果误差函数对于已知权值的微分在连续的递归过程中符号不变,我们可试着增加它的学习速率。相反,如果符号变化,则应减少学习速率。

为了利用包扩学习速率的适应调整这样一些方法,在每一递归过程中,需要有决定学习速率的适当算法。这两个算法分别是 Jacobs[1988]的 delta-bar-delta 算法和 Souncek[1992]去耦合的动量算法。delta-bar-delta 算法在每一次递归过程中更新学习速率。其数学表达式如下:

$$\eta_{ij}(n+1)=\eta_{ij}(n)+\Delta_{ij}(n) \tag{4-39}$$

其中 η_{ij} 表示连结第 i 和第 j 神经元权值的学习速率,学习速率的变化率 $\Delta\varphi_{ij}(n)$ 可按下式计算:

$$\Delta\eta_{ij}(n)=\begin{cases}k,\text{如果 } \bar{\delta}(-1)\delta_{ij}(n)>0\\ -\phi\eta_{ij}(n),\text{如果 } \bar{\delta}(-1)\delta_{ij}(n)<0\\ 0,\text{其它情况}\end{cases} \tag{4-40}$$

其中

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and cooperation: luzhenbo@yahoo.com.cn

$$\delta_{ij}(n)=\frac{\partial E(n)}{\partial w_{ij}} \tag{4-41}$$

和

$$\delta_{ij}(n)=(1-\theta)\delta_{ij}(n)+\theta\delta_{ij}(t-1) \tag{4-42}$$

这里 κ,ϕ 和 θ 是恒定参数。

delta-bar-delta 算法详细的描述和推导,见文献[Haykin,1994]。去耦合角动量算法的讨论见[Soucek,1992]。

随着递归过程的进行,学习速率的修正也以基本方式修改了反向传播算法。这种修改可能导致反向传播算法变化到仍能保持某种程度识别能力的边缘。无论如何,在此类问题方面有许多有用的变型方法,他们大都是以经验为依据,最后的决断依据网络权值的收敛集合。

先前提到,如何避免局部最小是反向传播算法训练中的一个问题。有两种避免局部最小的方法,它们分别是模拟退火算法和遗传算法。兴奋函数列入温度项后所构造的模拟退火函数,根据某一规律随着训练的进行而减小。遗传算法(Gas)同时考虑搜索空间上的许多点,并使用概率法则来引导其搜索。因此,避免局部最小的可能性很大。详细讨论避免局部最小的应用问题可见[Masters,1993]。避免局部最小的另一项技术是在训练模式中注入噪声。在训练早期,由随机信号发生器产生的噪声可能很大,随着训练的进行,噪声渐渐减小,最终减至为零,因而算法收敛。训练早期存在噪声在某种意义上同模拟退火方法相似。

4.6 梯度下降中的变量

梯度下降问题中,变量可按频率调整(块适应和数据适应技术)和方向调整(一阶和二阶技术)分成两类。

4.6.1 块适应和数据适应梯度下降方法的比较

当每个训练矢量被实验后,用数据适应技术可更新网络权值。也就是说,每一单个模式矢量产生一梯度。算法在一段时间的运行,要沿着这些梯度中的每一个来进行。其结果类似于通过最终趋向直接减小遍及整个训练模式误差函数的权值空间上的无规行走。在此意义上,数据适应方法可看成具有随机性质。很明显,数据适应的最新结果将趋向于对单个模式中噪声影响更敏感。

块适应技术保持权值调整值不变,直到整个数据块产生的效果被累加完毕为止。一般说来,尽管有存在较小块的可能性,但该数据块的大小就是全部训练集合。在某种意义上,块适应方法是更纯正的梯度下降算法形式,在全部训练集合矢量的集合里,更新方向的梯度下降真正是负梯度。其结果是,由于训练步长在块内所有训练模式上要取平均,因此块适应技术在数值方面更稳健。所以二阶技术倾向于使用块适应方法。没必要从上面就得出块适应调整技术优于其它方法这一结论。数据适应技术的随机性质实际上在避免局部最小问题上是有用的,该问题在反向传播算法中是最经常讨论的问题之一。

反向传播技术可用于任一种更新方式。对于反向传播算法,我们早期考查的编码是具有数据适应性的。然而,为充分利用数据适应更新的随机性优点,表示训练矢量的阶数在训练调整期间应是随机的。构造编码块需很小的工作量,全部所需就是存贮一段时间内变化累积的空间。当训练开始时,清这块存储空间。在训练期间,对每个训练矢量的权值调整总和进行累

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@hotmail.com

加,在训练结束时,网络权值已调整完毕。

4.6.2 一阶和二阶梯度下降方法的比较

一阶技术仅依赖于梯度。二阶技术利用二阶梯度。在这两种情况中反向传播算法对于计算梯度仍然很有用。二阶梯度对于数值计算很敏感,但总的来说,当使用在块适应方法时二阶技术具有更优越的数值性能。一个特别普通的二阶技术既共轭梯度方法将在 4.11 节中讨论。

4.7 拓 扑

层数和每层节点数将影响决策曲面。对一给定任务到底需要多少隐层单元没有简单规则可循。希望得到在隐层中具有最少神经元这样的网络。除了训练中和训练后的性能问题外,过多的隐层神经元数会产生所谓过拟合现象。网络有过多的信息处理能力时,过似合就可能产生。它将学习训练集合不重要的方面。在图 4.17 中可见过拟合现象。

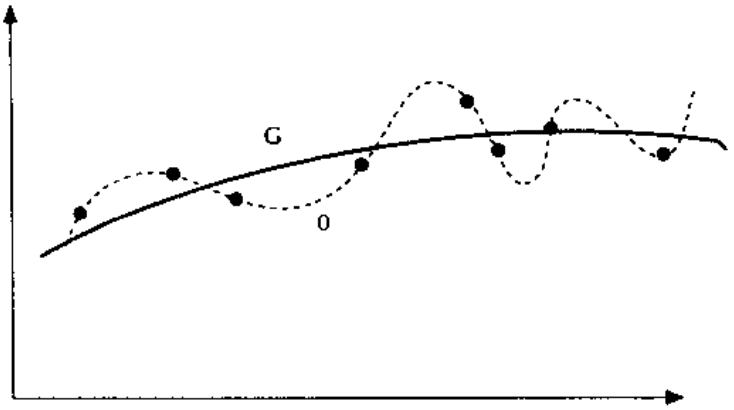


图 4.17 拟合图

一般建议由较少的隐层节点数开始,最好是低于解决此问题所需要的节点数。如果网络没有令人满意地收敛,确认网络缺少分化数据的能力,于是可增加隐层大小。这个过程重复进行,直到获得解决问题的最小两络为止。

删除两络隐节点,也是定义网络拓扑的另一方法。Lee 等[1992]提出一种删除方法,在他的方法中可被删除的节点是:

1. 对于所有训练模式来说输出是常量(在限度内)。
2. 对于权值矢量的辐度比其它隐节点和输出层的模相对小。

4.8 ACON 和 OCON 的比较

另一拓扑问题是使用带有多个输出量的单个两络(ACON,一个网络所有类),还是使用带有单个输出的多重网络(OCON,一个网络一类)。在 ACON 网络中,只有一个神经网络负责识别所有字符,而且,只有运行它来获取字符的识别。例如,一个用于识别数字的 ACON 网络,将有十个输出神经元,每个数字对应一个。与之相对照,OCON 的实现将由十个网络和一

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

个输出组成。这十个网络的每一个,专门识别特定的数字,一个后端结果在图 4.18 中给出。作为专门网络,单个 OCON 网络可能更简单、更快、更容易训练。Kung[1993]称 OCON 甚至需要更少的全部隐层节点。在一个 side-by-side 测试中,OCON 圆满达到了对收敛速度和精确度方面的要求。这种测试,对 36 个字母和数字字符使用了 432 个训练矢量和 396 个测试矢量,其结果总结在表 4.1 中。

表 4.1 ACON 与 OCON 模式在手写字数识别方面的性能比较

	训练精度	普通精度	训练时间
ACON(BP)	405/432=94%	324/396=82%	正常=1.00
OCON(BP)	430/432=99.5%	344/396=87%	大约=0.25

Tsay 等[1992]也对一个基于 OCON 的网络识别器在手写数字方面进行了测试。他们在训练集合上获得了 100% 的识别率。测试集合上获得 84.3% 的识别率。训练集合包含来自 70 个书写者的 700 个数字,同时,测试集合大小是 500。识别过程中的最后阶段,单个 OCON 网络的输出被输入到一个称为 MAXNET 的最后分类器中,见图 4.18。

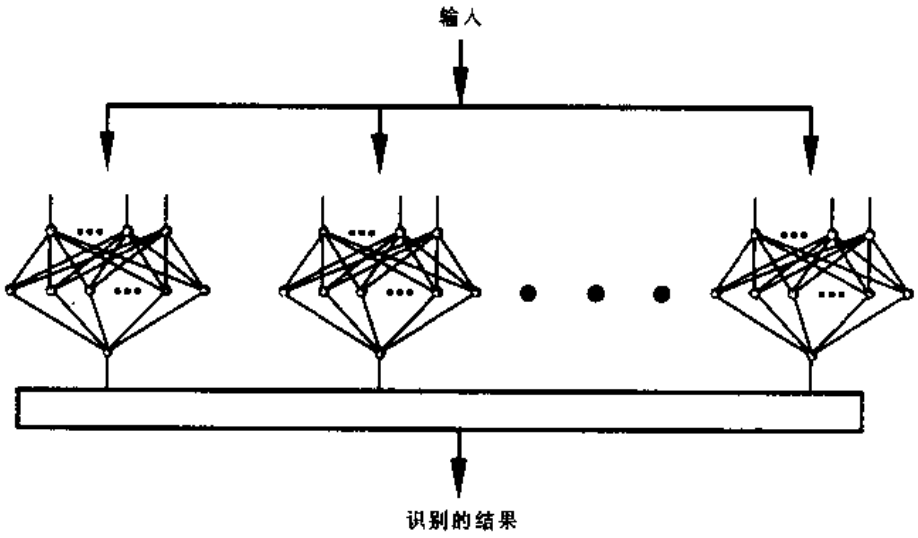


图 4.18 MAXNET 分类器

上述结构的新变形如下[Bebbis 等,1992]。它即不是纯粹的 OCON 也不是 ACON,而是介于两者之间。这一过程由 ACON 网络开始,经训练和测试后,求出混淆矩阵(对于混淆矩阵我们将在十二章中讨论)。基于这样的结果,我们确知该网络容易混淆哪些字符。然后整个网络被分成较小的网络,有理由认为它们能做得更好。重复此过程,直到获得可接受的结果。实际上,网络从 ACON 进化到了 OCON。值得强调的问题是手写数字的识别。利用刚描述的过程,定义四个子网络,这些子网络分别指定为:{4,1,7,2},{3,5},{6,9},{0,8}。这些网络组合的方式,几乎是 MAXNET 的反向方法。一个前端分类器被用于选择特殊的子网,该子网络将用于分类的最后阶段,见图 4.19。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

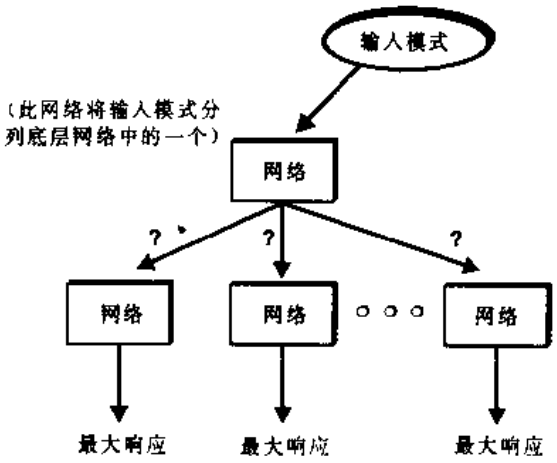


图 4.19 反向 MAXNET 分类器

4.9 过训练和推广

图 4.20 说明对于过低的误差容忍度，训练能导致网络出现全面的低性能[Hammerstrom, 1993]。实际上，网络开始记忆训练集合细节的同时，已失去了它的推广能力。这在本质上同过拟合思想是相似的，见图 4.21。

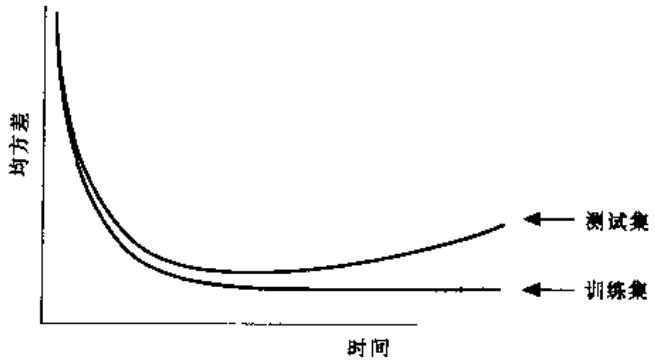


图 4.20 过度训练曲线

测试集合误差开始升高的点，即便在此点训练集合误差减少，先前也并不知道。这时问题变为：如何知道有效训练已停止，而过训练正在出现？

传统的收敛测试要求在所有训练矢量集合上的误差 E_t，小于某些特定的极小值。假设提供了一独立测试集集合，周期计算这些测试集集合向量的误差 E_{test}。例如，对每一段时间或每 k 段时间，如果 $E_{test}(n)<E_{test}(n-k)$，则测试集误差已经减小。这意味着应该贮存目前的值作为权值 W_{test_min}。但我们不进行这种尝试，并且贮存时间段数 n_{test_min}。注意，用反向传播网络去训练权值的误差项仅依赖训练集合矢量，所以，除了交换权值本身外，我们什么也没做。修正的目的仅在于提供一个尺度，它指出何时拓扑结构或训练集合出现问题。如果 n_{test_min} 比收敛发生时的时间段小，那么，下面可能有一个是真的：

1. 网络中有太多的隐单元，应减小隐层大小。

2. 训练集合并没充分地描述决策类，应当增加该集合。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and cooperation: luzhenbo@yahoo.com.cn

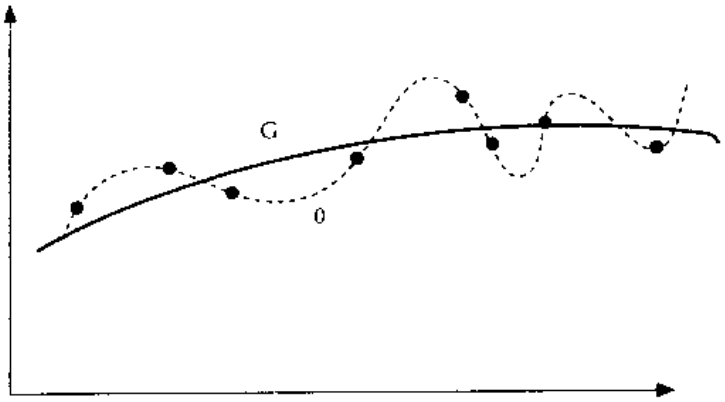


图 4.21 过拟合曲线

能改进反向传播训练网络推广能力的技术，包括带有噪声和权值衰减的训练。它用噪声注入作为一手段来避免出现局部最小。

另一个改进推广能力的方法就是熟知的权值衰减，它给误差函数引入一个额外的项，表达式如下：

$$E_{new} = E_{MSE} + E_{WD} = E_{MSE} + \alpha \parallel W \parallel^2 \tag{4-43}$$

这里 α 是一常数，$\parallel W \parallel^2$ 表示为

$$\parallel W \parallel^2 = \sum_{i \in C_{total}} W_i^2 \tag{4-44}$$

上式中，C_{total} 代表所有网络中所有的突触权值集合，E_{MSE} 表示熟知的均方差，E_{WD} 为新增加的权值衰减误差项。新项 E_{WD} 的作用是减小系统中全部权值的大小。即使权值大大地减少了均方差项，E_{MSE} 仍趋于增长。由于权值衰减项引起所有权值的降低，对误差 E_{MSE} 影响极小的额外权值将趋于消失(假设其值接近于 0)。没有权值衰减误差项，额外权值可导致不良的推广。它们对接纳任意值都不会受到惩罚。为了在训练误差方面得到相对小的改进，它们也能引起网络对数据的过拟合。出现于权值衰减和权值平滑(对此我们将简略讨论)中，这一增加额外误差项的方法是常用于错误姿态稳定问题的调节技术。

总的来说，模式在相邻点间将具有相关度。对一完全联接的前馈神经网络，在神经网络结构中近邻信息是不明显的。这样图像中近邻点的相关性都严格，是作了足够训练后的人为结果。在此意义上，向网络挑战，让它去识别大量杂乱或无联系的模式样本。为了达到足够的训练，所获得的训练时间和模式集合的大小因此增加了。分离出近邻点间的相关先验信息的网络(在训练算法影响下)能自由提取它所能获得的任何关系。很明显，一个被限定在某些方式下的网络，在模式识别问题领域中的推广方面将会更成功。

可采用几个方法去解决这类问题。新识别器(见十一章)和特征映射(见第六章)描述了针对这类问题的结构性方法。作用在反映位置空间关系的输入数据上的变换，也用来处理这类问题(见第六章)。在本节我们将讨论已知的权值平滑方法，这种方法通过某种调整把空间相关性嵌入反向传播训练算法中。

用于模式分类的神经网络，一般包含许多自由权值。网络训练是在权值空间中寻找一个良好的解。因为权值空间一般总是很大，存在许多解，但不是所有的解都反映空间的相互关系。下面称作权值平滑的技术，描述了训练网络的一种强迫考虑空间相关性的方法。类似权值衰减，权值平滑技术在常用于反向传播训练算法的典型均方误差中，引入一个额外误差项。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

让 I 和 H 分别代表输入层和隐层中的神经元数。如果 $\mathbf{W}_{ji}$ 表示连接第 j 隐层神经元和第 I 输入神经元的权值，则第 j 隐层神经元的权值矢量是 $(\mathbf{W}_{j1}, \mathbf{W}_{j2}, \cdots, \mathbf{W}_{jI})^T$。下面的公式能作为衡量第 j 隐层神经元的权值矢量平滑程度的一个尺度：

$$\sum_{i=2}^{I} [\mathbf{W}_{ji} - \mathbf{W}_{j(i-1)}]^2 \tag{4-45}$$

权值平滑算法使用的误差项如下：

$$E_{new} = E_{MSE} + E_{WS} = E_{MSE} + \xi \sum_{j=1}^{H} \sum_{i=2}^{I} [W_{ji} - W_{j(i-1)}]^2 \tag{4-46}$$

其中 ξ 是常量，E_{ws} 是衡量所有隐层神经元平滑程度的误差项。

注意，尽管式(4-46)很明显地可应用在一维模式中，但它很容易扩展到二维或高维模式。进一步注意到，假设相邻模式点输入到相邻输入神经元，式(4-46)通过正比于相邻权值矢量平均距离的平方误差项(这项与权值衰减误差项相对比已作了标明)施加了一种惩罚。

回忆 4.3 节，利用反向传播算法改变误差函数产生新的微分公式。由于改变了误差函数，所以为满足使用权值平滑误差函数的要求，我们必须确定新的微分公式。对式(4-4)微分得到：

$$-\frac{\partial E_{NEW}}{\partial W_{ji}} = \frac{\partial W_{MSE}}{\partial W_{ji}} + 2\xi[2W_{ji} - W_{j(i-1)} - W_{j(i+1)}] \tag{4-47}$$

其中 $(2W_{ji} - W_{j(i-1)} - W_{j(i+1)})$ 项代表了权值 $\mathbf{W}_{ji}$ 和它的两个近邻(输入层)间差的和。当使用权值平滑误差函数时，对反向传播算法由式(4-47)可导出了下面的微分公式：

$$\Delta W_{ji}(t+1) = \alpha \Delta W_{ji}(t) + \eta\{\Delta W_{ji}^{s}(s+1) - 2\xi[2W_{ji}(t) - W_{j(i-1)}(t) - W_{j(i+1)}(t)]\} \tag{4-48}$$

这里

$$\Delta W_{ji}^{s}(t+1) = -\frac{\partial E_{MSE}}{\partial w_{ji}} \tag{4-49}$$

两个进一步增强手段可被应用到权值平滑算法中。它们是(1)过放松，(2)引入一个退火平滑面子。过放松技术逼近 $\Delta W_{ji}(t+1)$ 时，采用下边两项操作：

$$\Delta W_{ij}^{bp} = \eta \Delta W_{ji}^{s}(t+1) + \alpha \Delta W_{ji}(t) \tag{4-50}$$

$$\Delta W_{ji}^{sm}(t+1) = W_{ji}^{bp}(t+1) - 2\eta\xi[2W_{ji}^{bp}(t+1) - W_{ji}^{bp}(t+1)W_{j(i+1)}^{bp}(t+1)] \tag{4-51}$$

注意式(4-50)与带有均方差的反向传播算法没有什么差别。式(4-51)提供了对基于由式(4-50)获得的新权值的平滑操作。基于这些新权值的算法更新技术被称为过放松，它常用于递归优化中。重写式(4-50)如下：

$$\Delta W_{ji}^{sm}(t+1) = \gamma W_{ji}^{bp}(t+1) + \frac{(1-\gamma)}{2}[2W_{j(i-1)}^{bp}(t+1) - W_{j(i+1)}^{bp}(t+1)] \tag{4-52}$$

其中

$$\gamma = 1 - 4\eta\xi \tag{4-53}$$

式(4-52)阐明这样的思想：平滑操作视为在权值 W_{ji} 和它的两个近邻权值间求加权平均，常数 γ 作为权值因子，常量 γ 小于或等于 1。较小的值会产生最强的平滑效果。当 $\gamma = 1$ 时，平滑项不起作用。注意，通过平均过程，平滑效果传播给整个网络权值。

平滑常量可被退火，既在训练早期，它具有较强的作用，但在训练后期阶段它减小且接近于 1。结果在训练后期阶段它的影响可以被忽略，在此阶段它还破坏解的精确性。训练早期

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and cooperation: luzhenbo@yahoo.com.cn

阶段 γ 增强了平滑效果,因此在解上加了一个过程结构因子。

对此我们可以选择一个单调增加的函数来表示 γ:

$$\gamma = \gamma(t) = 1 - (1 - \gamma_0)e^{-t/T} \tag{4-54}$$

其中 γ_0 和 T 是常量,常量 T 决定平滑效果的衰减速率。这样,它的大小可根据训练集合的大小而定。常量 γ_0 的选值接近 1,以避免引起同基于均方差的梯度下降法相比太强的平滑效果。

4.10 训练集合和网络大小

对于一给定带有指定维数输入的网络拓扑,将有一些能成功训练网络权值的最小训练样本数。这个所需样本数随输入空间的维数按指数规律增长[Kneer 等,1992],这个现象有时被称为"维度灾难"。它和其它原因一起将促进特征提取技术的应用。特征提取提供了一方法来减小输入矢量的维数,同时对于识别来说保持重要的信息。

前馈网络当然能同许多各种各样的特征提取技术联合作用,并且可通过各种各样的方法对它进行训练。关于对那些识别特征来自前端特征提取的神经元或其它网络的讨论,见第六章。

训练集合必须足够大而又多种多样,以便充分地描述问题域。尽管有时其内容不一定很准确,但被识别的每类必须是存在的。不太明显的是一些不能被识别的反例也可能需要,当涉及到 ACON 网络时这一点是非常正确的。如果训练一个 OCON 网络让它去识别数字 2,它也应当由一些不是 2 的其它数字去训练。

在每一类型中,必须存在大量的例子以反映此类型的现实世界变化。还回到数字识别例子,人们以不同方式有时甚至令人吃惊的方式写数字 2。这样的变化是区域性的,但几乎可以肯定它是具有国家特点的。

我们早期研究过,网络变得越大,它越可能过拟合。解决此问题在于,提供大量丰富的训练数据。而网络并不能学习训练矢量中出现的所有细节,因此需要加以推广。这样,对于一给定的拓扑,人们不仅要问:取多少训练集合矢量才是足够的? 假如在第 1 隐层有 L_0 个输入单元和 L_1 个隐单元,则输入层和第 1 个隐层间的权值数是 $N = (L_0 + 1) \times L_1$,这里包括偏畸权值。其中输入层同其它层相比较大,这些层构成了系统中的主要自由参量。这种情况下,至少使用 2N,或最好使用 4N 个训练矢量,作为粗略选择的训练矢量数量。

总之,以上把大量的注意力用于处理反向传播算法的明显缺欠,特别是在梯度下降法方面。要想实际应用这些技术还存在很多问题。如果读者得出此印象,作者也完全是无意的。梯度下降法是普遍存在而又很有用的。大量的关于改进它们的工作就足以表明它们很实用。在结束梯度下降这个主题前,我们将简单讨论更重要的二阶技术之一——共轭梯度方法。

4.11 共轭梯度方法

共轭梯度方法首先在基于一阶梯度的权值空间中建立一方向矢量,然后决定基于二阶梯度沿着这个矢量算法可运行多久。用反向传播算法,有许多共轭梯度算法的变形。下面的讨论可见 Kung[1993]。另一些共轭梯度技术的非严格数学讨论,可见 Masters[1993]。

注释:$\vec{W}$ 是由所有权值组成的矢量

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

$E'_{\overline{W}}$是梯度矢量

$E''_{\overline{W}}$是 Hessian 矩阵

计算梯度矢量后(利用反向传播算法),方向矢量 $\mathbf{d}_k$ 更新为:

$$\mathbf{d}_k = -E'_{\overline{W}}(k) + \beta_{k-1}\mathbf{d}_{k-1} \tag{4-55}$$

$$\mathbf{d}_0 = E'_{\overline{W}}(0) \tag{4-56}$$

其中 β_{k-1}计算如下:

$$\beta_{k-1} = \frac{E'_{\overline{W}}(k)^T E'_{\overline{W}}(k)}{E'_{\overline{W}}(k-1)^T E'_{\overline{W}}(k-1)} \tag{4-57}$$

更新的步长大小由下式定:

$$\eta_k = \frac{E'_{\overline{W}}(k)^T E'_{\overline{W}}(k)}{\mathbf{d}_k^T E''_{\overline{W}}(k)\mathbf{d}_k} \tag{4-58}$$

最后,权值矢量更新为:

$$W_{k+1} = \overline{W}_k + \eta_k \mathbf{d}_k \tag{4-59}$$

从式(4-55)我们看到更新的方向矢量是两项矢量和。第一项是权值空间误差函数的梯度,它正是我们在反向传播算法中所见到的;第二项来自反向传播算法的动量项。在反向传播算法中,动量趋向于保持一基于它自己过去的特殊权值的运动。$\beta_{k-1}\mathbf{d}_{k-1}$项进行更全局性地操作。它的趋势是保持前一次更新的所有方向。不管是深刻还是大致,当我们回忆反向传播算法选择自由参数的技术,都会注意到在某种程度上 β_{k-1}在反向传播算法中起到于动量常量 α 相似的作用,但它并不是一自由参数。式(4-57)精确告诉我们如何选择它作为当前梯度幅值平方对前一周期梯度平方之比。

根据式(4-59)我们观察到 η_k 项起到反向传播算法学习速率的作用,用其可建立步长大小。步长是权值空间中沿方向矢量运行的距离。不同于反向传播算法,它并不是一自由参量,必须通过直觉和经验来优化。考查一下式(4-58)所表达的内容——如何计算步长大小。特别地,我们要注意分母中的二阶梯度。当误差区面斜率急剧变化时,倾向于选择较小的步长,反之,误差表面的斜率变化不大时,增加步长。

共轭梯度技术的一个主要优点(同反向传播算法相比)是消除通过实验和误差来发现学习参数的必要性,这对大的训练集合来说是非常困难的任务。另一个是同反向传播算法相比它能在很短的时间中趋于收敛。我们应该知道,每个共轭梯度训练周期中的计算与反向传播算法的训练周期中的计算相比更复杂。总的来说,二阶方法提供了更优越的数值计算性能,但应该应用在块适应方法中。因为二阶梯度对数值计算很敏感。

梯度下降技术是训练多重感知神经网络强有力和普遍的方法。然而,它们远不是能被使用方法的唯一类型。特别当我们希望改变误差函数或激活函数时,不需计算梯度的训练算法更方便。例如,如果我们选择了一不可微的误差函数,将会发生什么情况呢?随着各种特征识别工具的讨论,将遇到若干不同的训练算法。其中的一个算法,ALOPEX 算法,在网络结构、兴奋函数、误差函数以及神经元类型方面都具广阔的应用性,使我们对它产生了特别的兴趣。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

4.12 ALOPEX

ALOPEX 是把神经网络学习过程,看作最优化问题的随机并行算法[Pandya 和 Szabo, 1991]; Venugopal 和 Pansya,1991]。

ALOPEX 是模式提取算法的首字母缩写。ALOPEX 程序首先被用于描述确定视觉接收场形状的有关问题[Harth 和 Tzanakou,174;Tzanakou 等,1979]。Harth[1976]最初提出了视觉感知模型为一随机过程,在某一给定的水平上,修正接收的感觉信息去最大化中心模式分析器的响应。随后使用单个标量响应产生的反馈进行计算机模拟。视觉路径上简单神经回路被证明执行了 ALOPEX 算法[Harth 等,1987]。Harth 和 Pandya[1988]证明这个过程描述了一个逼近经典数学问题的新方法。这个问题是优化具有 N 个参数 xi(i = 1…N)的标量函数 f (x1,…xn),其中 N 是一大正整数。在模式分类中,Herman 等,[出版中]特别讨论了在 ALOPEX 中使用分段线性分类器问题。Rosenfeld[1987]在图形处理算法中的综述中,讨论了这一问题。

在过去的几年中,反向传播算法已成为神经网络中最普遍的学习算法。正如 Hinton [1989]指出,尽管它在小问题方面展现显著性能及其广泛应用前景,对一个域结构下提取的稳健算法,当学习任务变得复杂时反向传播算法便表现得不尽人意。我们也很有趣地注意到,这种普遍的学习算法(反向传播算法)在生物学方面并非可行。没有迹象表明,生物突触能反向传递误差或者对神经元能求导数[Hinton,1989]。梯度下降方法的缺欠也被反向传播算法采纳。此算法对初始权值分布是很敏感的[Kolen 和 Pollack ,1990],而且可能收敛到局部最小点而不是全局最小点[Rumelhart 等,1986;Ackley 等,1985]。另外,反向传播算法需要非线性神经元的变换是单调增加和可微分的函数[Rumelhart 等,1986]。微分的存在性是必要的。由反向传播算法来训练网络需要一个条件,其中的微分项会导致某种困难。而使用 ALOPEX 则不会出现这种情况(见 4.5 节)。

ALOPEXE 有一优点就是计算简单,并且可以用高速大规模集成电路来实现。这是因为并不需要更新权值处理过程元素间的相互关系。遍过反向传播算法,误差信息递归计算梯度再计算新的权值。ALOPEX 通过对网络中所有权值处理器预测全局性能的一个量度(一个标量损耗函数)来进行工作。权值的改变由每个权值处理器随机产生,这种作法基于同它自身输出变化和这个标量损耗函数变化的相关反馈。每个权处理器仅依赖于全局损耗函数,不需要处理器间的相互作用,而且算法的硬件实现更加容易。

Pandya 和 Venugopal[1994]的研究证明:由 ALOPEX 训练的网络与由反向传播算法训练的网络相比,在大范围的信噪比情况下,有更好的抗噪声能力。

Monk 的问题源于 Thrun 等[1991],它可用来评价各种学习技术的性能。当把 ALOPEX 应用于 Monk 问题时,特别在整个噪声数据集合上,发现它比其它算法更好推广(包括反向传播算法)。注意到 Monk 问题的训练集合比测试集合小得多。因此,它提供了良好的测试和推广能力。

我们现在描述一下用于训练神经网络的 ALOPEX 算法。到第 i 神经元的网络输入最:

$$net_i = \sum_j w_{ji} out_j + \theta_i \tag{4-60}$$

假设一 sigmoid 兴奋函数,第 j 神经元的输出是

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

$$out_i = \frac{1}{1 + e^{-net_i/Q_0}} \tag{4-61}$$

(注意,ALOPEX 并不需要对兴奋函数做任何特殊选择。)

ALOPEX 算法确定了网络权值在第 n 次递归期间的新值:

$$w_{ji}(n) = w_{ji}(n-1) + \delta(n) \tag{4-62}$$

$$\delta_{ji}(n) = \begin{cases} -\delta, \text{概率为 } P_{ij}(n) \\ +\delta, \text{概率为 } P_{ij}(n) \end{cases} \tag{4-63}$$

δ 是小正常数,概率 $P_{ij}(n)$由下式给出:

$$P_{ij}(n) = \frac{1}{1 + e^{\frac{\Delta_{ij}(n)}{T}}} \tag{4-64}$$

$$\Delta_{ij}(n) = \Delta w_{ij}(n)\Delta E(n) \tag{4-65}$$

其中

$$\Delta w_{ij} = [w_{ij}(n-1) - w_{ij}(n-2)] \tag{4-66}$$

$$\Delta E(n) = [E(n-1) - E(n-2)] \tag{4-67}$$

如果 e_k 是输出神经元 k 的误差,那么总的取自所有输出神经元的误差 E 为:

$$E = \sum_k e_k \tag{4-68}$$

此算法沿着减少误差 E 的方向上采取有偏随机行走。步长 δ 是一常量,温度 T 决定随机行走的效果。建议 T 的初始值选得大一些,然后,随训练过程的进行下降到相关平均值 Δ_{ij}

$$T = |\Delta| \tag{4-69}$$

容易验证,概率的含义总是向总误差 E 降低的方向倾斜。从图 4.22 中,可以更清楚地看到所取概率的意义。

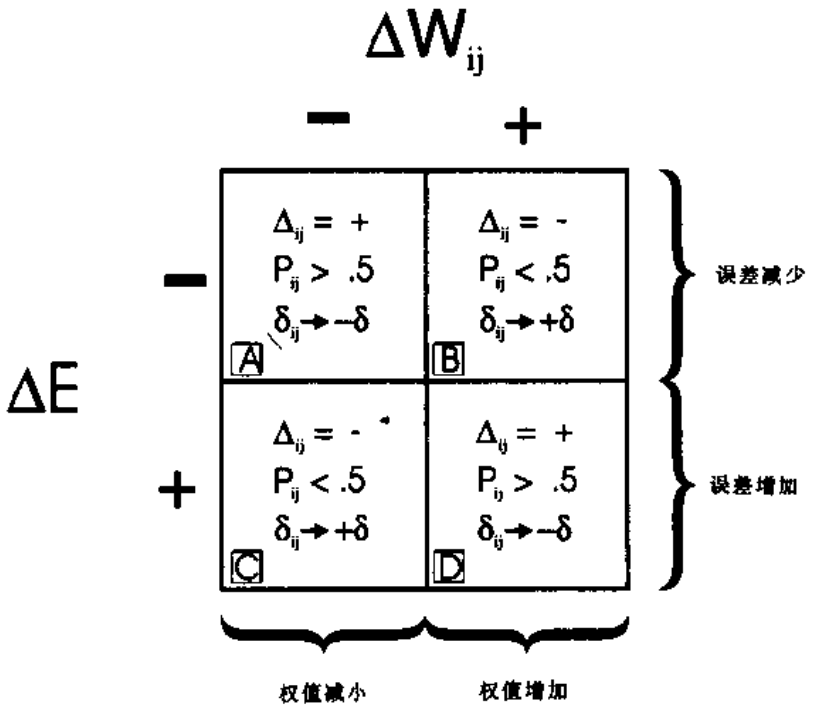


图 4.22 ALOPEX 组合

把 ALOPEX 训练算法的收敛特征同反向传播算法的收敛特征比较很有趣。已经指出,对

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

缺少局部最小但有强全局最小这样一类问题，ALOPEX 不能像反向传播算法那样迅速地收敛。在特定的环境下，ALOPEX 要预防的问题并不一定出现。同时，在特定的情况下，存在许多反向传播算法是件好事，并不是什么缺点。预先知道这些条件的存在是没什么困难的。对于其它类问题，ALOPEX 算法完成得也非常令人满意。尽管在这些情况下 ALOPEX 不能像反向传播算法那样迅速收敛，但其它的优点为它的使用提供了有力的依据。

对于那些 ALOPEX 能发挥其作用的可能问题，它将是一强有力的候选者。概括一下，选择 ALOPEX 可能是因为：

1. 网络拓扑误差函数和神经元类型的选择的灵活性。
2. 它可避免由于随机性质造成局部最小的倾向性。
3. 它具有标明梯度下降的微分独立性。
4. 它良好的推广能力(特别在噪声数据上)。
5. 它对高速并行实现的适用性。

由于它具有普适性，人们在加强和推广反向传播算法方面作了大量的努力。相比较而言，ALOPEX 算法不太显著且不太被注意。改进 ALOPEX 算法包括：同模拟退火相似的退火策略，避免局部最小的温度扰动机制和增加了动量项等。

注意，在我们离开主题之前应关注一下 ALPOLEX 的实现。很明显，ALOPEX 算法依赖于产生 $P_{ij}(n)$概率的随机数。在绝大多数编译器中，随机数发生器使用一个线性的求余算法，因此它受到一些严格限制[Masters,1993]。设 p_n 在伪随机序列中是一个非负的整数。由线性求余算法，下一个伪随机数按 $p_{n+1}=(ap_n+c)$模 m 产生。许多 C 编译器使用 $m=2^{15}=32768$，因此有许多可用的伪随机数。这个问题由于算法是周期性这一事实被恶化，一旦某值出现，同样的序列将随它而产生。这意味着不能保证 m 个序列就有 m 个。结果，应选择一个更合适的随机数发生器，在与 ALPOLEX 的联接中使用这个发生器。

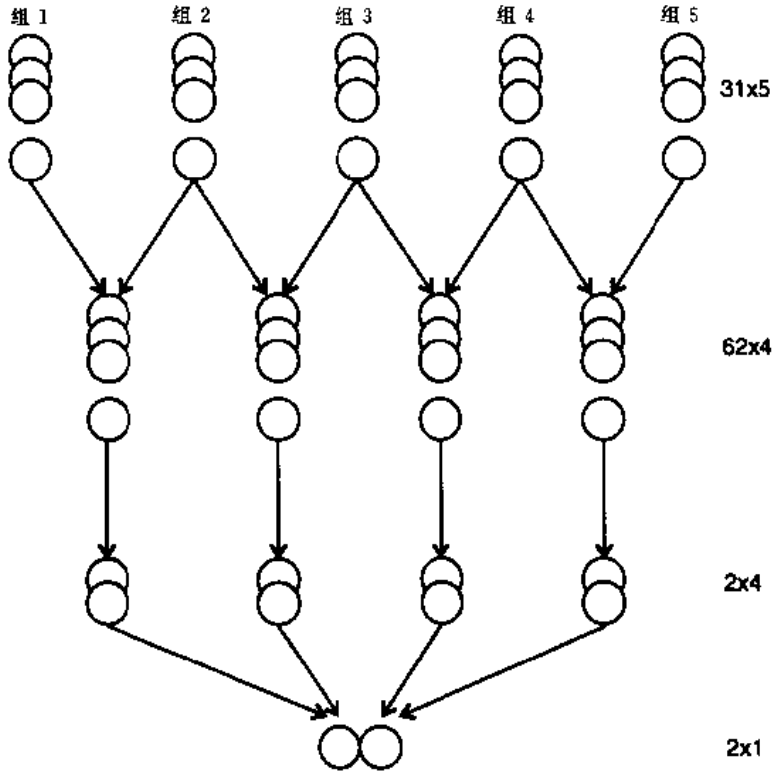
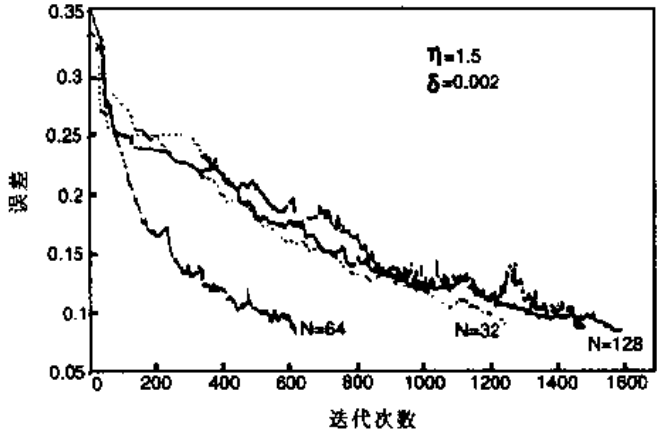


图 4.23 用于声纳目标识别的神经网络结构

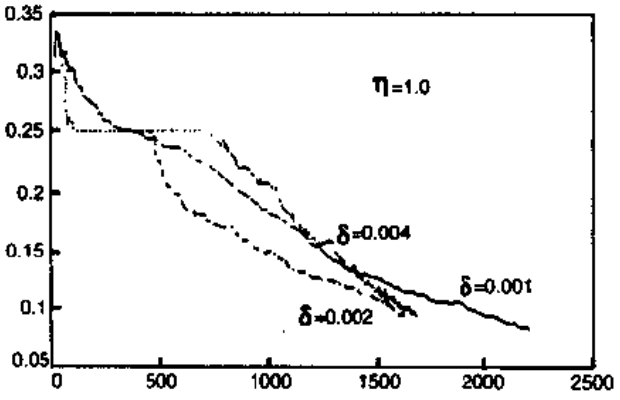
More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@hotmail.com

Pandya 和 Venugopal(1994)研究了用侧扫描声纳回波利用 ALOPEX 算法对海下目标连续识别的有效性。这是一个探索海底和避免障碍方面的实际问题(水下船舶导航)。由于声纳回波的复杂性和噪声的干扰,需要人们投入大量精力。ALOPEX 算法可被用于训练网络结构,包括对顺序声纳回波的依赖关系的谱时学习。图 4.23 给出了用于这一识别问题的网络结构。声纳回波的能量密度作为神经网络的输入,输入层的每一组有 31 个神经元。第一隐层有四组,每组由 62 个神经元组成;第二隐层由四组,每组由 2 个神经元组成,这样可描述 5 个顺序回波。

用 ALOPEX 算法对应两个已知目标的侧扫描声纳回波进行了仿真,两个目标是木盒和金属鼓。训练表现为每一类都可以提供 52 个回波。图 4.24(a)给出了对于隐层神经元数的不同时误差收敛特征。发现,隐层神经元数的增加并不提高收敛速率。在实验中得出,当步长改变时也可得到相似特性(见图 4.24(b))。收敛的网络由 12 个测试回波描述,且能对它们正确地进行分类。



(a)



(b)

图 4.24　误差收敛的变化(a)第一隐含层的神经元的数目改变时(b)步长大小改变时

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

ALOPEX 算法的实现在清单 4.8 中给出。

清单 4.8

```
/*************************************************************************
*                                                                        *
* ALOPEX                                                                 *
*                                                                        *
*************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <math.h>

// FUNCTION PROTOTYPES
void  ShowResults(int, long int,double temp, double alpha, double eta);

// DEFINES

#define MAXLAYERS 4      // MAX NUMBER OF LAYERS IN NET (IN OUT & HIDDEN)
#define MAXNEURONS 30 // MAX NUMBER OF NEURONS PER LAYER
#define MAXPATTERNS 10 // MAX NUMBER OF PATTERNS IN A TRAINING SET
#define MAXPREV    3      // NUMBER OF PREV EPOCH FOR WHICH WEIGHTS
                          ARE MAINTAINED
#define ARCGRAN   250    // ARCHIVE GRANULARITY
#define NONE       0      // DO NOT ARCHIVE ERROR DATA
#define ALL        1      // ARCHIVE ALL ERROR DATA FOR EACH ITERATION
#define AVERAGE 2         // ARCHIVE AVERAGE ERROR DATA FOR EACH
                          EPOCH
#define TRUE       1
#define FALSE      0

//-------------------------------------------------------------------


class ALOPEX
{
private:
 double W[MAXLAYERS][MAXNEURONS][MAXNEURONS];       // WEIGHTS MATRIX
 double Wprev1[MAXLAYERS][MAXNEURONS][MAXNEURONS]; // previous WEIGHTS
for correlation
 double Neuron[MAXLAYERS][MAXNEURONS];              // Output of all net neurons
 double ERROR[MAXPATTERNS][MAXNEURONS];  // Stored error over epoch/outlayer
 double AvgErr;

// Alopex specific data
 double delta;                                  // The ALOPEX step size
 double T;                                      // Temperature for ALOPEX probability fn
 double Ep1;                                    // Sum error over iteration n-1
 int AnnealCount;
 double SumDELTAc;

// Topology
 int NumLayers;                         // Number of layers
 int OutLayerIndx;                      // array index of last layer
 int LayerSize[MAXLAYERS];              // Number of neurons in the each layer
 int NumWeights;                        // Tot number of weights

// Pattern data
 double inPattern[MAXPATTERNS][MAXNEURONS];   // Input values for each pattern
 double Desired[MAXPATTERNS][MAXNEURONS];     // desired value for each
                                              //  pattern/output
 unsigned long int CurrIter;                 // Current  iterations number.
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
long int Epoch;                                  // Counts number of epochs
int EpochFlag;                                   // True at end of each epoch
int NumPatterns;                                 // Total patterns in training set
int CurrPat;                                     // Current pattern used in training
int ConvCount;                                   // The number of consecutive
                                                 // patterns within tolerance
int ConvergeFlg;                                 // Flag indicates convergance has
                                                 // occurred

// PARMS
 double Qzero;                        // For Neuron activation function
 double ERRTOL;                       // min Error tolerance required for convergence
 double ETA;                          // ALOPEX Version of learning rate
 unsigned long int MAXITER;           // Max iterations to do before stopping

// PRIVATE METHODS
 double ErrFunc(double Target, double Actual);          // Calc the log error
 double CalcSumErr();                 // Calc tot err over all patterns/outlayer neurons
 void  RecordErrors();                // At each iter, record error results
 void  ArchivePut();                  // Keep a record on disk of net progress
 public:
  ALOPEX(void);
  void  GetTrnSet(char *Fname);                  // Load training set from file
  void  GetInputs(void);                         // Load input layer w/ curr pattern
  void  RunNet(void);                            // Run the networks forward pass
  void  GetParms(char *);                        // Load parameters from parm file
  double Sigmoid(double Net, double Tempr);      // The non-linear squashing function
  void  SetRandomWeights(void);                  // Init weights to random values
  int   IsConverged(void);                       // Check for convergence
  void  AdaptWeights(void);                      // Modify weights by ALOPEX method
  void  SaveWeights(char *);                     // Save trained weights to file
  int   train(char *, char *);                   // Top level control of training the net
  long int QueryEpoch(){return Epoch;}
  double QueryQzero(){return Qzero;}
 };


 // MISC FLAGS
 int ArchOn  = AVERAGE;       // Set to 1= ALL, NONE, AVERAGE or WORST to control
                              //  data sent to ARCHIVE file for graphical analysis
 FILE *ARCHIVE;               // Archive Training sequence
 //-------------------------------------------------------------------------
 // METHOD DEFINITIONS

 /*************************************************************************
 * Function GetTrnSet                                                     *
 *   Class constructor                                                    *
 *************************************************************************/

 ALOPEX::ALOPEX() {
  AvgErr  = 0.0;
  CurrIter=0;
  Epoch=0;
  AnnealCount=0;
  CurrPat = -1;                          // Current pattern used in training
  ConvCount=0;
  ConvergeFlg=FALSE;                     // Flag indicates convergence
  SumDELTAc=0.0;
  ETA=2.0;
 }
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
/*************************************************************************
* Function GetTrnSet                                                     *
*   Load training set                                                    *
*************************************************************************/

void ALOPEX::GetTrnSet(char *Fname) {        // for small training sets
FILE  *PFILE;
int   x;
int   pat,i,j;
double inVal;

PFILE = fopen(Fname,"r");
if (PFILE==NULL){
  printf("\nUnable to open file \n");
  exit(0);
  }
fscanf(PFILE,"%d",&NumPatterns);

for (pat=0; pat<NumPatterns; pat++) {
  for (i=0; i<LayerSize[0]; i++) {
     fscanf(PFILE,"%lf",&inVal);
     InPattern[pat][i]=inVal;
     } /* endfor */

  // Now get desired / expected values
  for (i=0; i<LayerSize[OutLayerIndx]; i++) {
     fscanf(PFILE,"%lf",&inVal);
     Desired[pat][i]=inVal;
     } /* endfor */
  } /* endfor */
fclose(PFILE);
}
/*************************************************************************
*                                                                        *
* Function GetParms                                                      *
*   Loads topology from file spec'd file if avail.                       *
*   otherwise tries DEFAULT.PRM                                          *
*                                                                        *
*************************************************************************/

void ALOPEX::GetParms(char *PrmFileName){
FILE *PRMFILE;
PRMFILE = fopen(PrmFileName,"r");
if (PRMFILE==NULL){
  printf("\nUnable to open Parameter file: %s \n",PrmFileName);
  printf("\nAttempting to open default file: PARMS1");
  PRMFILE = fopen("PARMS1","r");
  if (PRMFILE==NULL){
    printf("\nUnable to open Parameter file\n");
     exit(0);
     }
  }
printf("\nLoading Parameters from file: %s\n",PrmFileName);
fscanf(PRMFILE,"%lf",&Qzero);
fscanf(PRMFILE,"%lf",&ETA);
fscanf(PRMFILE,"%lf",&delta);
fscanf(PRMFILE,"%lf",&T);
fscanf(PRMFILE,"%ld",&MAXITER);
fscanf(PRMFILE,"%lf",&ERRTOL);
fscanf(PRMFILE,"%d",&NumLayers);

printf("&Qzero=%lf",Qzero);
printf("&delta=%lf",delta);
printf("T=%lf",T);
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
printf("MAXITER=%d",MAXITER);
printf("ERRTOL=%lf",ERRTOL);

printf("\nNumber of layers=%d\n",NumLayers);
for (int i=0; i<NumLayers; i++) {
  fscanf(PRMFILE,"%d",&LayerSize[i]);
  printf("Number of neurons in layer %d = %d\n",i,LayerSize[i]);
   }
OutLayerIndx = NumLayers-1;                    // accommodate 0 org'd arrays
fclose(PRMFILE);

NumWeights=0;
for (int lyr=OutLayerIndx; lyr>0; lyr---)      //Visit each layer
  NumWeights += LayerSize[lyr] *  LayerSize[lyr-1];
printf("\n%d\n",NumWeights);
}

/*****************************************************************************
* FUNCTION SetRandomWeights                                                  *
*                                                                            *
*   Initialize weights between 0 and 1                                       *
*****************************************************************************/
void ALOPEX::SetRandomWeights(){
int i,j,k;
double zWT;
  srand(6);
  for (i=1; i<NumLayers; i++) {
    for (k=0; k<LayerSize[i]; k++) {
      for (j=0; j<=LayerSize[i-1]; j++) {          // One extra for bias neuron
        zWT=(double)rand();
        W[i][j][k] =zWT/32767.0;
        }
      }
    }
}


/*****************************************************************************
* FUNCTION GetInputs                                                         *
*                                                                            *
*   Loads input layer neurons with current pattern                           *
*****************************************************************************/

void ALOPEX::GetInputs(){
int i,j,k;
EpochFlag=FALSE;
CurrIter++;
CurrPat++;                                     // Update the current pattern
if (CurrPat>=NumPatterns){
  EpochFlag=TRUE;
  CurrPat=0;
  Epoch++;
  }

for (i=0; i<LayerSize[0]; i++) {
  Neuron[0][i]=InPattern[CurrPat][i];          // Show it to the neurons
   }
}


/*****************************************************************************
* FUNCTION RunNet                                                            *
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and Cooperation: luzhenbo@yahoo.com.cn


```
*                                                                          *
*   Calculates outputs for all network neurons                             *
*****************************************************************************/

void ALOPEX::RunNet(){
int lyr;   // layer to calculate
int dNeuron; // dest layer neuron
int sNeuron; // src layer neuron
double SumNet;
double out;

for (lyr=1; lyr<NumLayers; lyr++) {
  Neuron[lyr-1][LayerSize[lyr-1]]=1.0; //force bias neuron output to 1.0
  for (dNeuron=0; dNeuron<LayerSize[lyr]; dNeuron++) {
     SumNet=0.0;
     for (sNeuron=0; sNeuron <= LayerSize[lyr-1]; sNeuron++) {          //add 1 for bias
       SumNet += Neuron[lyr-1][sNeuron] * W[lyr][sNeuron][dNeuron];
        }
     out=Sigmoid(SumNet,Qzero);
     Neuron[lyr][dNeuron] = out;
      }
   }
}


/*****************************************************************************
* Function RecordErrors ( )                                                  *
*   Save error results by neuron & pattern for later use                     *
*****************************************************************************/

void ALOPEX::RecordErrors(){
   int j;
for (j=0; j<LayerSize[OutLayerIndx]; j++) {
  ERROR[CurrPat][j] = ErrFunc(Desired[CurrPat][j], Neuron[OutLayerIndx][j]);
  } /* endfor */
}

/*****************************************************************************
* IsConverged                                                                *
*   Calculates error for each neuron and each pattern                        *
*****************************************************************************/


int  ALOPEX::IsConverged(){
double dNETj;
double SumErr;                                          //Cumulative error
int  i, j;
int LocalConvFlg=TRUE;

SumErr=CalcSumErr();
if (SumErr>ERRTOL) {
   return FALSE;
 } else {
   return TRUE;
   } /* endif */
}


/*****************************************************************************
* Function ArchivePut()                                                      *
*   Creats a record of training results                                      *
*****************************************************************************/
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
void  ALOPEX::ArchivePut(){
int  i, j;

AvgErr =CalcSumErr()/NumPatterns;

if (ArchOn==AVERAGE) {
  if ((ARCGRAN*(Epoch/ARCGRAN)) == Epoch) {        // only save ARCGRANth epochs
     fprintf(ARCHIVE,"%ld %lf\n",Epoch,AvgErr );
     printf("\nEpoch:%d ",Epoch);
     printf("Avg=%lf\n",AvgErr);
     } /* endif */
   }
}


/*****************************************************************************
* Function ErrFunc                                                           *
*    Calculates the error for a given output neuron & target value           *
*****************************************************************************/

double ALOPEX::ErrFunc(double Target, double Actual) {
  double E;
if (Target>=.99) {
   E=log(1/Actual);
   }
  else {
   E=log(1/(1-Actual));
   } /* endif */
return E;
}


/*****************************************************************************
* Function CalcSumErr()                                                      *
*    Calculates total error over all (ouput neurons) and patterns            *
*****************************************************************************/

double ALOPEX::CalcSumErr(){
  int j,Pat;
  double E,desire;
E=0.0;
for (Pat=0;Pat<NumPatterns ; Pat++) {
  for (j=0; j<LayerSize[OutLayerIndx]; j++) {
     E += ERROR[Pat][j];
     }
  } /* endfor */
return E;
}


/*****************************************************************************
* Function AdaptWeights()                                                    *
*    Calculates new weights based on the ALOPEX algorithm                    *
*****************************************************************************/

void ALOPEX::AdaptWeights(){
double E,R,P, DELTAc;
double DELTAWEIGHT, DELTAERROR;
int lyr, i, j,k;

E=CalcSumErr();
DELTAERROR=E-Ep1;
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication or cooperation: luzhenbo@yahoo.com.cn


```
Ep1=E;                                          //Prev iter errors
for (lyr=OutLayerIndx; lyr>0; lyr—) {           // Visit each layer
  for (j=0; j<LayerSize[lyr]; j++) {            // Visit each neuron
    for (k=0; k<=LayerSize[lyr-1];k++ ) {       // Visit all Weights (incl. bias)
      DELTAWEIGHT= W[lyr][j][k]-Wprev1[lyr][j][k];
      DELTAc= DELTAWEIGHT*DELTAERROR;   //Calculate the correlation
      SumDELTAc+=fabs(DELTAc);          //Running sum of correlation over all weights
      Wprev1[lyr][j][k]=W[lyr][j][k];
      if (Epoch > 1) {                          //Wait till system is primed
        P = 1/(1 + exp(ETA*DELTAc/T));          // Probability fn once it is
        }
       else
        P= 0.5;                                 //50% till then
      R=(double)rand();
      R=R/32767.0;
      if (R>P) {
        W[lyr][j][k]=W[lyr][j][k] - delta;
        }
       else {
        W[lyr][j][k]=W[lyr][j][k] + delta;
        } /* endif */
      } /* endfor */
    } /* endfor */
  } /* endfor */

//Establish a new temp for the alopex probability fn base on avg correlation
if ((AnnealCount>=10)&&(SumDELTAc>0.0)) {
  T=SumDELTAc/ AnnealCount;
  AnnealCount=0;
  SumDELTAc=0.0;
  } else {
  AnnealCount++;
  } /* endif */
}


/*************************************************************************
* FUNCTION SaveWeights                                                   *
*                                                                        *
*    Output weights to default file DFLT.WGT                             *
*************************************************************************/
void  ALOPEX::SaveWeights(char *WgtName) {
int lyr,s,d;
double zWT;
FILE *WEIGHTFILE;

  WEIGHTFILE = fopen(WgtName,"w");
  if (WEIGHTFILE==NULL){
    printf("Unable to open weight file for output:%s\n",WgtName);
    exit(0);
    }
  printf("SAVING CALCULATED WEIGHTS:\n\n");
  fprintf(WEIGHTFILE,"0.00\n");                 // Threshold always 0
  fprintf(WEIGHTFILE,"%lf\n",Qzero);            // Temperature
  fprintf(WEIGHTFILE,"%d\n",NumLayers);         // Number of layers
  for (lyr=0; lyr<NumLayers; lyr++) {           // Save topology
    fprintf(WEIGHTFILE,"%d\n",LayerSize[lyr]);  // Number of neurons/layer
    }
  for (lyr=1; lyr<NumLayers; lyr++) {           // Start at 1st hidden
    for (d=0; d<LayerSize[lyr]; d++) {
      for (s=0; s<=LayerSize[lyr-1]; s++) {     // One extra for bias
        zWT=W[lyr][s][d];
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
            fprintf(WEIGHTFILE,"%lf\n",zWT);
            }
          }
        }
    fclose(WEIGHTFILE);
}


/*************************************************************************
* FUNCTION Sigmoid                                                       *
*                                                                        *
*                                                                        *
*   Output non-linear squashing function                                 *
*************************************************************************/

double ALOPEX::Sigmoid(double Net, double Tempr){
  double x;
x = Net/Tempr;
if ( x >100.0)
  return 1.0;
if ( x < -100.0)
  return 0.0;
return 1.0/(1.0 + exp(-x));
}


/*************************************************************************
* FUNCTION train                                                         *
*    Trains the network                                                  *
*     Input: Training data file name and Parameter file name             *
*     Output:non-linear squashing function                               *
*************************************************************************/

int   ALOPEX::train(char *TrnFname, char *ParmFname){
if (ArchOn) ARCHIVE=fopen("archive.lst","w");
GetParms(ParmFname);
GetTrnSet(TrnFname);
SetRandomWeights();
int Converged=0;
while ((!Converged) && (CurrIter < MAXITER) ) {
  GetInputs();
  RunNet();
  RecordErrors();
  if (CurrPat==0)                                  // Test for full epoch
    ArchivePut();
   Converged=IsConverged();
    AdaptWeights();
   }
 if (ArchOn) fclose(ARCHIVE);
 return Converged;
 }

//-------------------------------------------------------------------------


/*************************************************************************
* FUNCTION ShowResults                                                   *
*                                                                        *
*   Output Summary specific to the parity-3 problem                      *
*************************************************************************/
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication E-mail: luzhenbo@yahoo.com.cn


```
void ShowResults(int ConvergeFlg, long int CurrEpoch, double temp){
printf("\n-------------------------------------------------------------\n");
if (ConvergeFlg) {
  printf("SUCCESS: Convergance has occured at iteration %ld\n",CurrEpoch);
  }
 else {
  printf("FAILURE: Convergance has NOT occured!!\n");
  } // endif
printf("Temperature = %lf\n",temp);
printf("\n-------------------------------------------------------------\n");
}

ALOPEX Alopex;

/*****************************************************************************
*  MAIN                                                                      *
*****************************************************************************/


int main(int argc, char *argv[])
{
int Converged ;

if (argc>3) {
  Converged = Alopex.train(argv[1],argv[2]);
  }
 else {
  printf("USAGE: ALOPEX TRAINING_FILE PARMS_FILE WEIGHT_FILE\n");
  exit(0);
  }

// Show how we did
ShowResults(Converged, Alopex.QueryEpoch(), Alopex.QueryQzero());
if (Converged)
  Alopex.SaveWeights(argv[3]);  // Save the weights for later use

return 0;
}
```

参考书与文献

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J., "Learning algorithm for Boltzman machines," *Cognit. Sci.*, vol. 9, pp. 147–169, 1985.

Baker, T. and McCartor, H., "A comparison of neural network classifiers for optical character recognition," *Proc. SPIE, vol. 1661, pp. 191–202, 1991.*

Bebbis, G. N., Georgiopoulis, M., Papadourakis, G. M., and Heilman, G. L., "Increasing classification accuracy using multiple neural network schemes," *Proc. SPIE Applications of Neural Networks III,* vol. 1709, pp. 221–231, 1992.

Cybenko, G., "Approximation by superpositions of a sigmoidal function," *Math. Control, Signals, Syst.*, vol. 2, pp. 303–314, 1989.

Funahashi, K., "On the approximate realization of continuous mappings by neural networks," *Neural Networks,* vol. 2, no. 3, pp 183–192, 1989.

Hammerstrom, D., "Working with neural networks," *IEEE Spectrum,* July 1993, pp. 46–53, 1993.

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

Harth, E., "Visual perception: a dynamic theory," *Biol. Cybern.*, vol. 22, pp. 169–180, 1976.

Harth, E. and Tzanakou, E., "ALOPEX: a stochastic method for determining visual receptive fields," *Vision Res.*, vol. 14, pp. 1475–1482, 1974.

Harth, E., Unnikrishnan, K. P., and Pandya, A. S., "The inversion of sensory processing by feedback pathways: a model of visual cognitive functions," *Science*, vol. 237, pp. 187–189, 1987.

Harth, E. and Pandya, A. S., "Dynamics of the ALOPEX Process: Applications to the Optimization Problem," in *Biomathematics and Related Computational Problems*, L. M. Ricciardi (Ed.), Reidel Publ., Amsterdam, pp. 459–471, 1988.

Hartman, K., Keeler, J. D., and Kowalski, J. M., "Layered neural networks with Gaussian hidden units as universal approximations," *Neural Computation*, vol. 2, pp. 210–215, 1990.

Haykin, S., *Neural Networks, A Comprehensive Foundation*, IEEE Society Press, Macmillan College Publishing, New York, 1994.

Herman, G. T., Odhner, D., and Yeung, K. T. D., "Optimization for pattern classification using biased random search techniques," *Ann. Operations Res.*, (in press).

Hinton, G. E., "Connectionist learning procedures," *Artif. Intell.*, vol. 40, pp. 184–234, 1989.

Hornik, K., Stinchcombe, M., and White, H., "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

Jacobs, R. A., "Increased rates of convergence through learning rate adaptation," *Neural Networks*, vol. 1, pp. 295–307, 1988.

Jean, J. S. N. and Wang, J., "Weight smoothing to improve network generalization," *IEEE Trans. Neural Networks*, vol. 5, no. 5, 1994.

Kneer, S., Personnaz, L., and Dreyfus, G., "Handwritten digit recognition by neural networks with single-layer training," *IEEE Trans. Neural Networks*, vol. 3, no. 6, pp. 962–968, 1992.

Kolen, J. F. and Pollack, J. B., "Backpropagation is sensitive to initial coaditions," *Complex Syst.*, vol. 4, pp. 269–280, 1990.

Kung, S. Y., *Digital Neural Networks*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

Lee, Y., Oh, S. H., Song, H. K., and Kim, M. W., "Design rules of multilayer perceptron," *Proc. SPIE, Science of Artificial Neural Networks*, vol. 1710, pp. 329–339, 1992.

Masters, T., *Practical Neural Network Recipes in C++*, Academic Press, Boston, San Diego, and New York, 1993.

Pandya, A. S. and Szabo, R., "A fast learning algorithm for neural network applications," *Conf. Proc. 1991 IEEE Int. Conf. Syst. Man Cybern.*, vol. 3, pp. 1569–1573, 1991.

Pandya, A. S. and Venugopal, P., "A stochastic parallel algorithm for supervised learning in neural networks," *IEEE Trans. Inf. Syst.*, E77-D, no. 4, pp 376–384, 1994.

Rosenfeld, A., "Picture processing: 1986," *Comput. Vision, Graphics, Image Process.*, vol. 38, pp. 147–225, 1987.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing, Vol. 1: Foundations*, D. E. Rumelhart and J. L. McClelland (Eds.), MIT Press, Cambridge, MA, pp. 318–362, 1986.

Soucek, B. *Fast Learning and Invariant Object Recognition*, John Wiley & Sons, New York, 1992.

Thrun, S. B., Bala, J., et al., *The MONK's Problems: A Performance Comparison of Different Learning Algorithms*, Carnegie-Mellon University, Pittsburgh, CMU-CS-91–197, 1991.

Tsay, S., Hong, P., and Chieu, B., "Handwritten digit recognition via OCON neural network by selective pruning," *IEEE Proc. 11th Int. Conf. Pattern Recognition*, pp. 656–659, 1992.

Tzanakou, E., Michalak, R., and Harth, E., "The ALOPEX process visual receptive fields by response feedback," *Biol. Cybern.*, vol. 35, pp. 161–174, 1979.

Venugopal, K. P. and Pandya, A. S., "ALOPEX algorithm for training multilayer neural networks," *Proc. IJCNN, Singapore*, pp. 182–190, 1991.

Wang, J. and Jean, J., "Resolving multifont confusion with neural networks," *Pattern Recognition*, vol. 26, no. 1, pp. 175–187, 1993.

Zurada, J. M., Zigoris, D. M., Arohime, P. B., and Desai, M., "Classification of printed characters using multi-layer feedforward neural networks," *IEEE Proc. 34th Midwest Symp. Circuits Syst.*, vol. 2, pp. 191–202, 1991.

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

第五章 其它类型的神经网络

5.1 概 述

本章将讨论另外两种重要的前馈网络。它们使用的神经元，不同于我们熟知的、含有sigmoid 兴奋函数的感知器所采用的神经元。

5.2 径向基函数网络

用于模式分类的神经网络设计，可看成超空间中的曲线拟合问题。其中权值的学习过程相当于寻找某一超曲面，该曲面可对训练数据提供最佳拟合。在这种情况下，该问题的推广就是利用超曲面对测试数据进行插值。Cover[1965]证明了，在低维空间中不可分的复杂的模式分类器问题，有可能在高维空间中变得线性可分。径向基函数(RBF)方法是在高维空间进行插值的一种技术。Broomhead 和 Love[1998]最率先使用 RBF 技术的二个人，他们提供了神经网络学习的一种新手段。

径向基函数网络包含有一维数足够高的隐含层，此层对输入空间进行非线性变换。在这些网络中，输出层提供了从隐单元空间到输出空间的一种线性变换。这样，基本的 RBF 网络仅有单隐层，面多层感知器(MLP)有一层或更多的隐含层。

5.2.1 网络结构

图 5.1 给出了 RBF 网络中每一层对输入矢量所做的变换。隐神经元相当于输入矢量(未画出)的兴奋函数，而分类器的输出只是这些兴奋函数的加权线性和。注意，不像 MLP 中的神经元不管处于哪一层都是同一种神经元模型，面 RBF 网络中的隐含层神经元根本不同于输出层中的神经元。

应用 RBF 网络，使得变换在隐含层进行，这就允许隐含单元空间具有更高维数。RBF 这种兴奋函数，只对固定位置“c”视野范围内的输入产生响应。因此当 $x=c$ 时 $f(x)$值最大，随着 $|x-c|$ 的值增加，$f(x)\to 0$(见图 5.1)，高斯势函数就是这样一种函数。

当使用径向基函数神经元时，模式的类可被可看成模式空间中点的高斯分布。当 RBF 神经元的输入距高斯函数的中心点足够近时，那么神经元就被激活了。多层感知器(MLP)中的每一隐含层的神经元计算其输入的加权和；面 RBF 网络的隐含层神经元是通过计算它们距感受野的远近程度(例如，输入矢量和对应的神经元的质心间的距高)来对输入进行编码。因此，RBF 分类器属于感受野分类器这一类。利用隐含层神经元的兴奋函数所确定的叠代局部区域，输出节点建立了复杂的判决区域。中心和宽度是与兴奋函数有关的两个重要参量。

对于一组输入量 x_j 和权值 w_{ij}，径向基函数神经元的输出为：

$$out = g\left[\sum_{j}^{n}(x_j - w_{ij})^2\right] \tag{5-1}$$

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and cooperation: luzhenbo@yahoo.com.cn

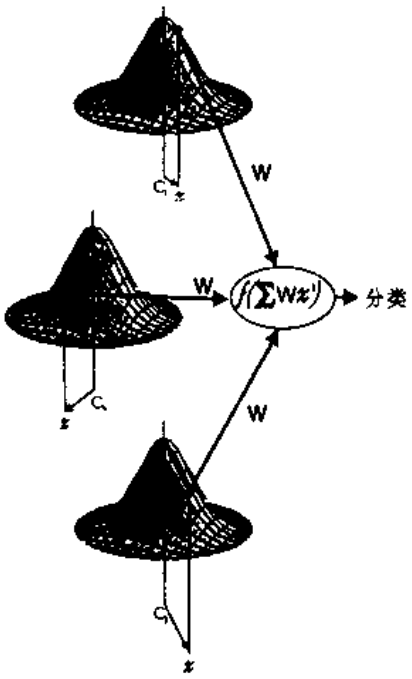


图 5.1 RBF 网络,输入空间的变换。隐含层神经元的响应为 x 与 c_i 的远近程度的函数

这里 $g()$是高斯函数:

$$g(r^2) = ce^{-r^2/\sigma^2} \tag{5-2}$$

距离函数可取欧几里德距离,因为它是最常用的一个。此例中,兴奋函数采用高斯函数。对于某种函数能否用作 RBF 的较详细讨论,请参阅 Micchelli[1986]。

图 5.2 给出了 MLP 和 RBF 网络的比较。应特别注意的是 RBF 网络中的异类神经元。MLP 对于非线性的输入-输出映射采用全局非线性函数。与此相反,RBF 采用了指数递减的局部非线性函数(见公式 5-2)。这样,RBF 比通常的感知器(MLP)有更强的局部聚类能力。

Wong[1991]证明:由于高斯函数的特性,RBF 网络难以学习映射的高频部分。这可能是由于这样一个事实,为了以所需的平滑度来描述一个输入——输出映射,遍历整个输入空间所需的 RBF 数量可能非常大。Wong 给出了利用反向传播算法训练 RBF 网络。Kung[1993]指出了在这种情况下反向传播算法效果减弱了。

5.2.2 RBF 训练

RBF 网络训练,包括确定函数的中心、宽度和联结隐含层和输出层神经元的权值。聚类算法,如 K-近邻算法,可用于解决这些问题。表示一个聚类的 RBF 神经元的相应中心,可看成聚类平均。对聚类算法的讨论参看 Duda 和 Hart[1973],Fukunaga[1990]。

这里,我们描述一个迭代聚类算法,它由 Musavi 等 [1991] 作为一个可行的方案最早提出。此算法特别有用,因为它能使隐含层神经元的数目最小。此聚类算法定义如下:

1. 将每一训练点分配给一个聚类来进行初始化。
2. 随机标记每一聚类(L=1,2……,C)。
3. 选择第一个聚类(L=1)。
4. 确定同一类中任一聚类作为被选聚类。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

5. 合并两个聚类，并计算新的平均值。

6. 找出新的均值和相反类中最近聚类的均值之间距离 d_{app}。

7. 计算从新的均值到属于它自己聚类的最远点之间的距离，此距离定义为聚类的半径 R。

8. 定义 α 为聚类参数：

a. 如果 $d_{app} > \alpha R$，接受第 5 步的结果（现在的 L 值和最新构造的聚类相联系，C 被增加到 $C+1$）。

b. 否则，放弃合并。恢复两个最初的聚类并回到第 5 步，L 增加到 $L+1$，C 不变。

9. 重复第 4 步到第 8 步，直到 $L = C$ 为止。

在此算法中，聚类参数 α 是一控制聚类范围的常量。α 值越大，减少的节点就愈少。但由于减少了重叠，就增加了训练集合的精确度。

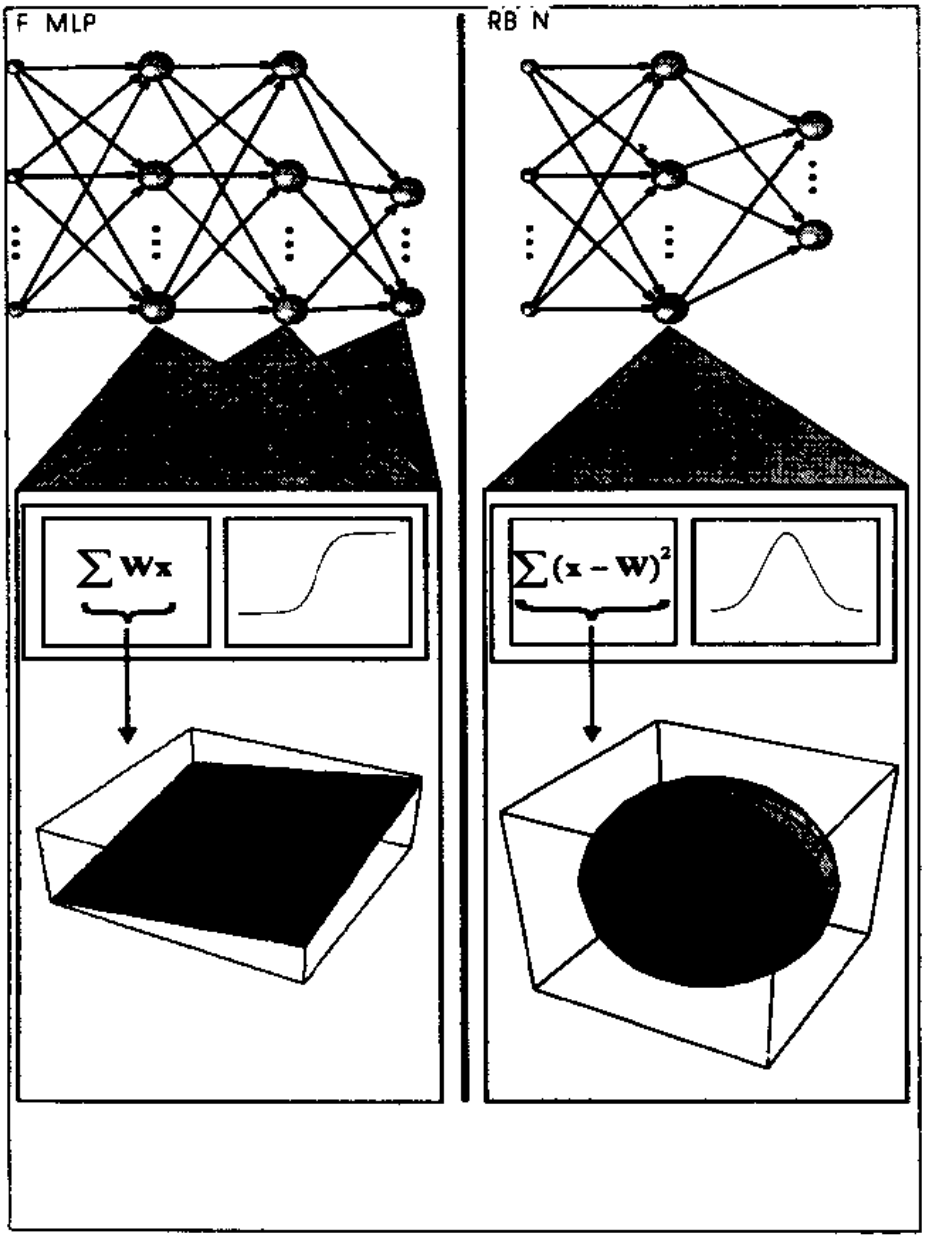


图 5.2　RBF 和 MLP 网络的结构

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and cooperation: luzhenbo@yahoo.com.cn

注意这个算法只解决二类问题。使用多个单类网络就可解决多类问题。对于某一类,根据成员函数可能有几种聚类。训练结果就是对每个聚类指定一个隐含层神经元。Musavi 等[1991]给出了一例子,此例中由高斯随机矢量产生的 400 个点,组成的二个数据集被分成两大类。图 5.3 画出了这两类的所有模式的模式空间。使用上面讨论的聚类算法训练网络后,模式分为 86 个聚类,因此就确定了 RBF 网络神经元的中心和半径参数。图 5.3 也画出了每一聚类的中心。之后使用一简单的 delta 规则以训练连接到输出神经元的权值,期望训练完成后对于类一的模式样本输出等于 0,对于类二的输出等于 1。图 5.4 给出了用图 5.3 所示的样本训练后的网络输出曲面。

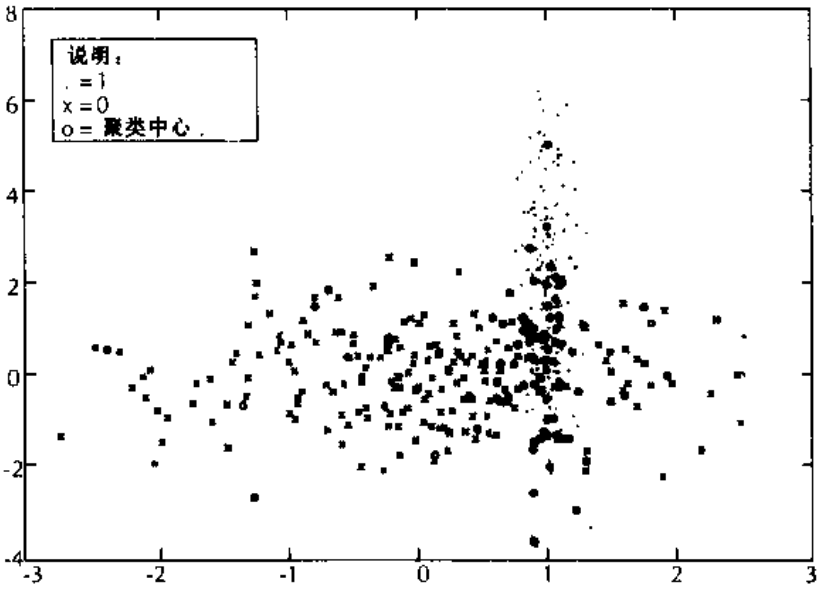


图 5.3 测试样本以及聚类中心(摘于 Musavi, M. T. 等 ACMANNA-91 ,Anal. Neural Network Appl. ,1991. 经过授权)

5.2.3 RBF 网络的应用

RBF 网络也被称作局部网络,它学习速率快并且对训练模式的表示阶数有较低的敏感性。Moody 和 Darken[1989]证明了使用 RBF 两络能获得较快的学习速率;同时 Poggio 和 Giorsi[1994]应用正则法则来提高 RBF 网络推广能力。对于研究 RBF 两络感兴趣的读者,Haykin[1994]进行了深入的探讨。

Renals[1989]将 RBF 网络应用在语音的模式分类方面。Lee[1991]已将 RBF 网络应用到手写数字识别问题。在此实验中,没有前端的特征提取。网络的输入是一原始位图格式。通常的感知器产生了 5.15%误差率,面径向基函数的误差率为 4.77%,两者的差别太小,得不出任何结论。然面,这足够证明 RBF 能够用来解决此类问题。

RBF 网络已被应用在广泛的领域,如图像处理[Poggio 和 Edelman,1990;Saah 等,1991];医疗诊断[Lowe 和 Webb,1990];时间序列分析[He 和 Lapedes,1991;Broomhead 和 Lowe ,1988;Kadirkamanathan 等,1991];语音识别[Ng 和 Lippman,1991;Niranjan 和 Fallside,1990]。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

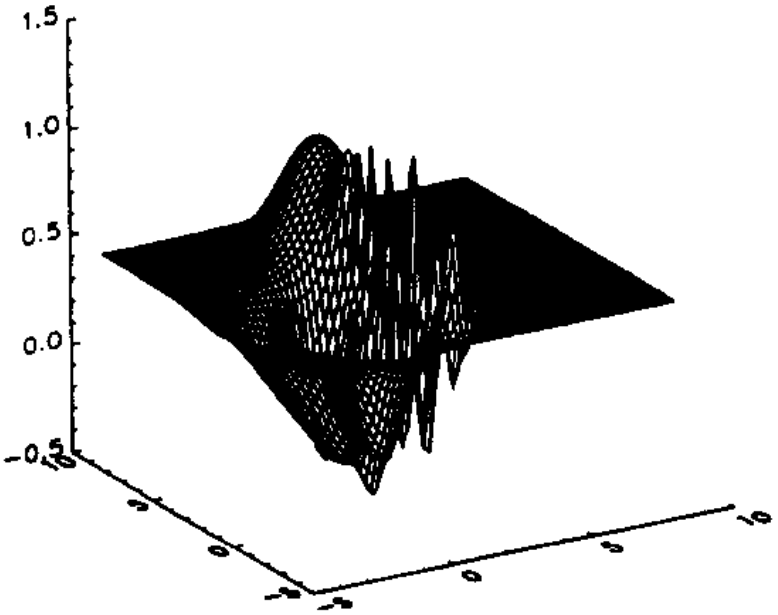


图 5.4 用图 5.3 所示的样本训练后的网络输出曲面(摘于 Musavi, M. T. 等 ACMANNA-91, Anal. Neural Network Appl., 1991. 经过授权)

5.3 高阶神经网络

随后第六章我们将讨论特征提取。讨论过程中,我们将看到可以通过两个阶段得到不同形式的不变量。首先,构造一不变特征集;然后将它们输入到识别器。另一种方法是将所需的不变量构造入网络本身结构中去,此节中讨论的高阶神经网络便可做到这点。

5.3.1 引言

对象识别的目标是不管输入图像的位置、大小、角度的方向怎样,如图 5.5 所示,它都必须被识别出来。多层感知器(MLPs)、神经认知机(在十一章讨论)以及高阶神经网络(HONN)[Giles 和 Maxwell, 1987]是能够达到这个目标的三个较成功的神经网络结构。HONN 的最重要优点是,能够将几何变换的不变性构造入网络中,因此不必通过叠代训练以更新权值。

对于 MLP 网络必须学习才能进行这样的识别,因此训练集合就必须包含一将识别对象进行变换后的大子集[Rumelhart, 1989; Troxel 等, 1988]。通常需要大量训练步骤,才能将几何变换中的这些概念进行归纳。神经认知机是含有简单神经元和复杂神经元的分级结构。此结构使之对于要识别的对象,不管其在输入区域的位置、大小上的微小变化或轻微畸变,都能对其进行识别[Fukushima, 1992]。它的局限是,模型里的神经元的数目几乎随要识别的对象的数量线性增加。

同这两种方法相比,HONN 是利用输入模式中的已知关系,将所需的不变量直接构造入网络结构。因此网络被"预训练",并且不需要学习几何变换的不变量,HONN 只需基于每个要识别对象,而不是基于它的各种变形来进行训练。

通过恰当地调整网络权值 HONN,可设计成对图像的二维坐标变换具有不变性。此

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and cooperation: luzhenbo@yahoo.com.cn

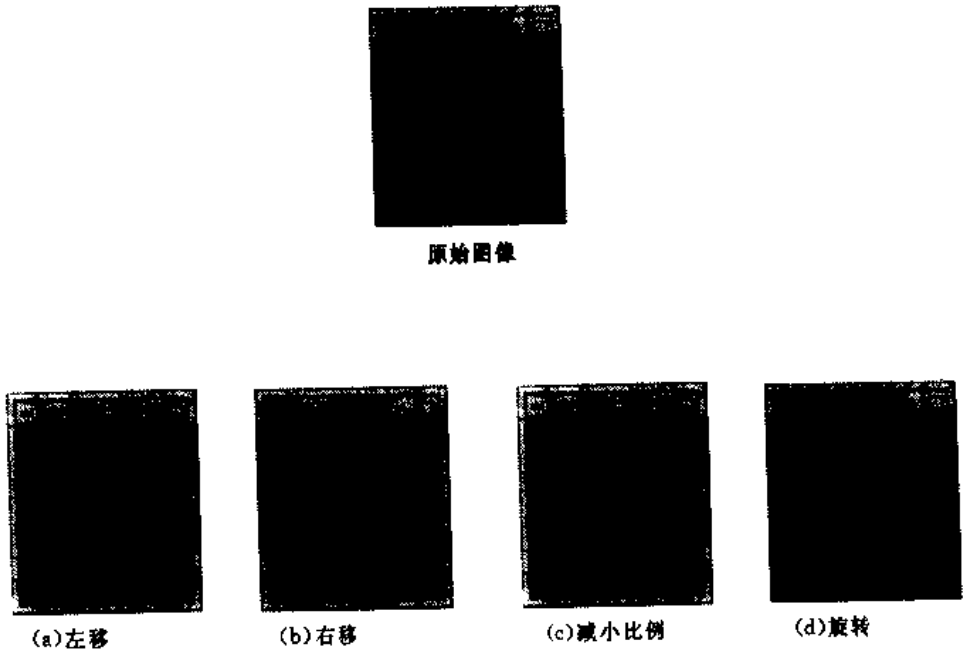


图 5.5 对样本图像进行平移、比例调整以及旋转后的图像

网络的二阶形式，能够对平移及比例畸变不敏感。一个三阶的神经网络能被用来进行平移、比例和旋转不变的对象识别，并且训练时间比其它的神经网络，如 MLP，有很大程度上的减少。

5.3.2 结构

通常，HONN 的一个节点 i 的输出 y_i，由下式给出：

$$O = f\Big(\sum_{i=0}^{N-1} w_i x_i + \sum_{j=0}^{N-1}\sum_{k=0}^{N-1} w_{jk} x_j x_k + \sum_{j=0}^{N-1}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} w_{jkl} x_j x_k x_l + \cdots\Big) \tag{5-3}$$

式中，f 是一非线性变换函数；x_j、x_k 和 x_l 是来自其它节点的输入；w_{ij}、w_{ijk} 和 w_{ijkl} 确定求和项中指定的每个输入的权值(见图 5.6(a)和图 5.7)。

HONN 可由反向传播算法进行训练，其它的高阶项被看作另外的输入。

图 5.6(a)和(b)给出了简单的一、二阶网络的例子。二阶网络图中的输入和输出层中间的阴影方块(见图 5.7)代表了乘积单元。此单元计算输入信号的乘积，并将结果输入到下一单元。

在一个 N 阶网络中，输入是一些乘积项，每一次最多为 N 项 (见公式 5-3)。严格的 N 阶网络中，每一次输入为第 N 项乘积，并且低阶项将丢失。对严格的三阶神经网络，其输出 y_i 由下式给出(见图 5.7)：

$$y_i(x) = S\Big[\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N} w_{ijk} x_j x_k x_l\Big] \tag{5-4}$$

注意，一阶或二阶项在此式中并没出现。也应注意，首先输入是三个一组地进行组合，然后使用这些乘积的加权和确定输出值。S 是非线性变换，所有其它的变量表示意义同公式(5-3)中所述一样。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

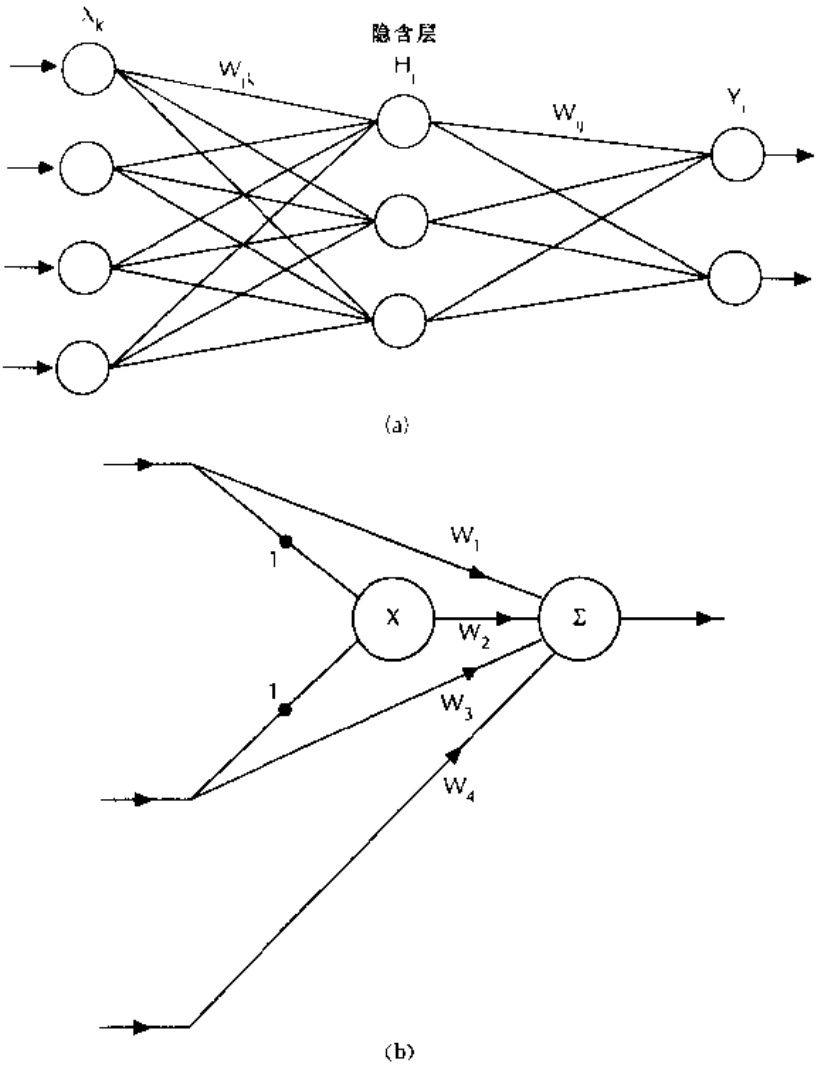


图 5.6 (a)中的多层感知器是一阶网络,每一权值修改一个输入 (b)中的二阶网络的前两个输入相乘(乘法器处)后在取合之前与权值再相乘。其它的连接与 MLP 类似

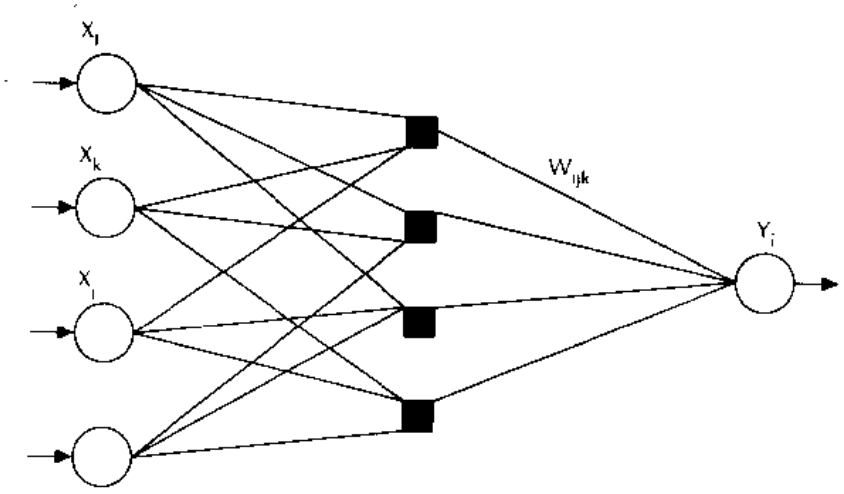


图 5.7 严格的三阶网络,在加权之前把三个输入相乘。然后对四个乘积项(每个方块一乘积项)进行求和

使用一个简单的 delta 规则训练三阶网络,双曲线正切函数作为兴奋函数。之所以采用正

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

切函数是因为它的双极性。它具有这样一个特性,即已证明在减少各种网络训练时间方面具有很好的效果。

5.3.3 几何变换的不变性

为了建立对三种几何变换的不变性,神经网络需要学习输入象素间的关系。HONN的权值受到这样的限制,能够组成相似三角形的三个象素组合与输出以同样的权值进行连接。使用的神经网结构可以是单层的前馈网络,每一类别对应一输出神经元。在上面方法中,所有相似的三角形被映射为同一权值。在这例中,"相似"意味着"当三角形沿同一方向被横截时,有相同集合和顺序的内角"。图 5.8 给出了相似三角形和不相似三角形的例子。

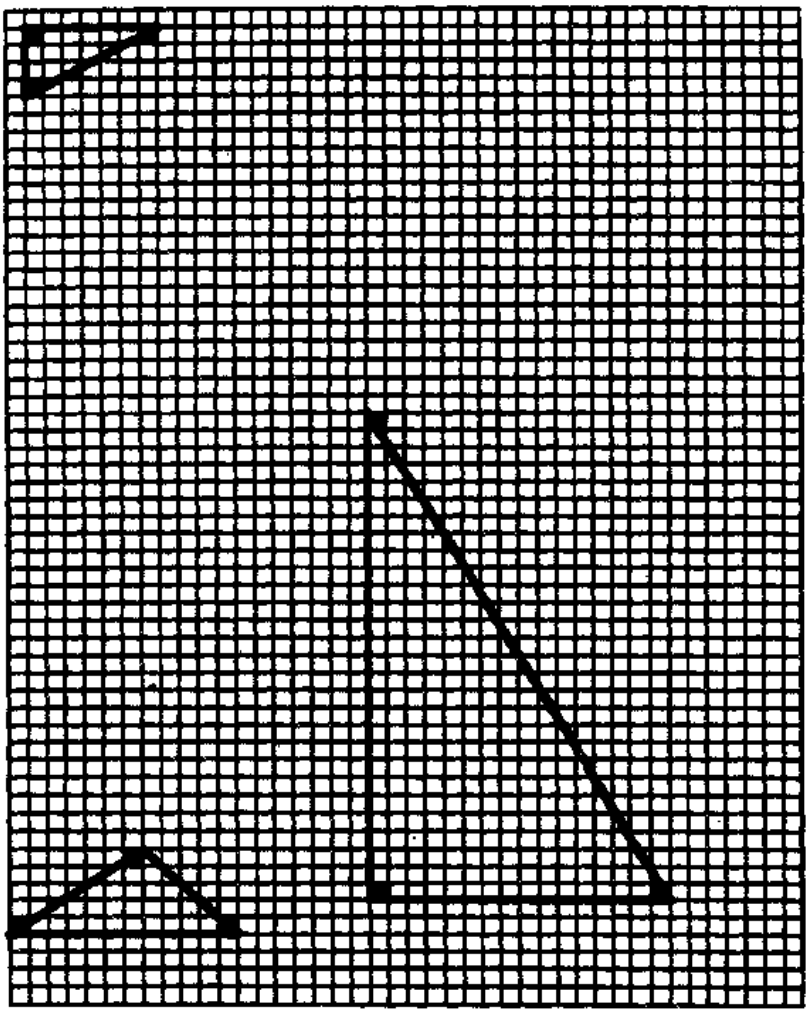


图 5.8 左上角和右下角的三角形是相似三角形。另一三角形与这两个三角形都不相似

当计算角度时,三角形总是沿同一方向被横截。因为内角的顺序是一很重要的变量,它允许我们区分一个三角形和它侧向翻转所产生的三角形。此例中,这两个三角形实际是不同的,因为(使用平移、比例改变或旋转的组合)不可能将三角形映射到它侧向翻转所产生的三角形上。对图像中每个可能的三个一组的内角必须计算,并且进行贮存。

HONN 的一个内在困难是权值数量随网络的阶数和输入的维数增加得非常快。在 N×N 分辨率的图像中,有"N^2 选 3"种方式来选择三象素的组合;这样一个 10×10 象素输入需要

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

"100 选 3"=161,700 组合;若分辨率增加到 128×128 象素点,则可得的选择性增加到 7.3×10^{11},这样大的组合数量实在是太大了,以至于在大多数机器中都不能存贮。在大部分实际应用中,必须采用一些方法来限制这种情况(例如,见 Spirkovaska 和 Reid [1993];Lisboa 和 Perantonis [1991])。

5.3.4 范例

考虑设计一个严格的三阶网络,以区分二个简单的几何图形(一个四边形,一个三角形),如图 5.9 所示。

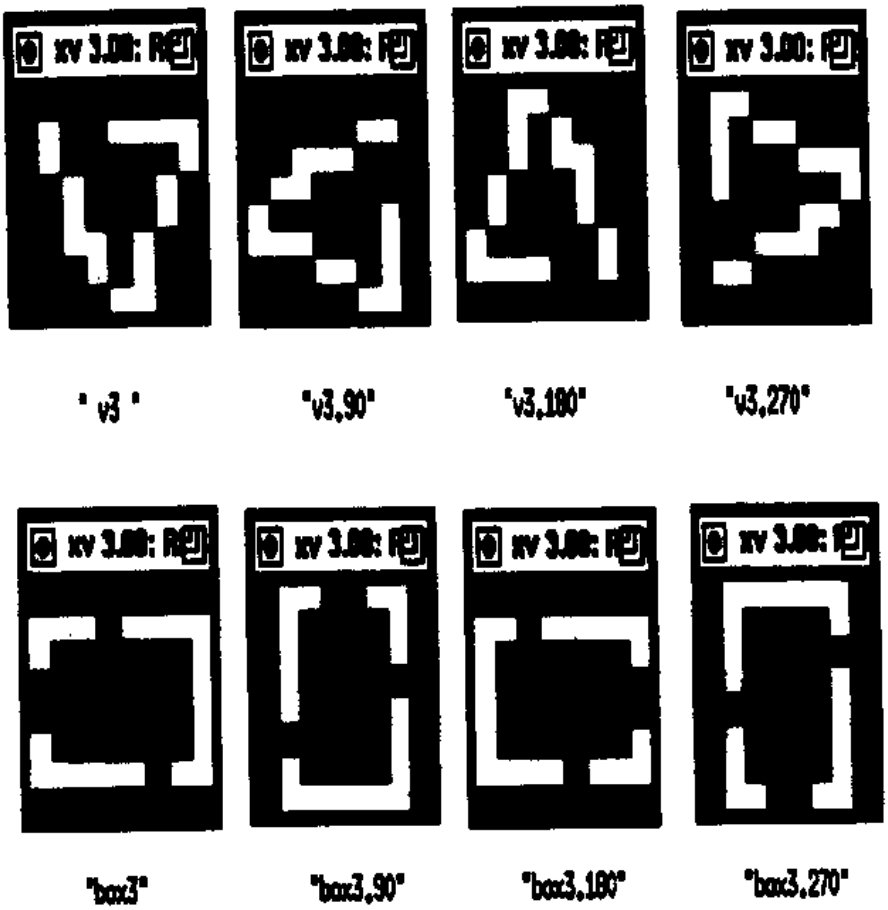


图 5.9 所使用的基本图形为四边形和三角形。图形的数值部分(扩展部分)表示为产生相应图形所需旋转(顺时针)的角度(单位为度)

第一步是定义合适的通过三角形一顶点的平面平分线(见图 5.10)。内角是由顶点和以平分线为参考点来计算。图 5.10 给出了四种可能情况/顶点位置和平分线位置的组合。确定了三个角度后,就能够计算出三角形的质心,并由此确定顺时针方向转动所需的顶点的次序(图 5.11)。实际上,以内角为参考点就足够了,但用质心为参考点更容易。

实际上,所计算出的角度能被量化成与之最近的 4 度范围,以便允许图像数据中可能的噪声分布。实验已经证明,三阶网络能区分两个具有不同数量噪声的简单的几何单色图(一个四边形,一个三角形)(见图 5.12,5.13)。对于四边形,训练网络产生一个"+1"的输出量,对于三角形,产生"-1"的输出量。对训练完的网络进行测试所使用的一些图像中加了许多象素点(例如,"雪花点"),其它的测试图像中故意丢失了一些象素。我们发现,实际上加上雪花点的图像比那些故意丢失了象素的图像更易被错误分类。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

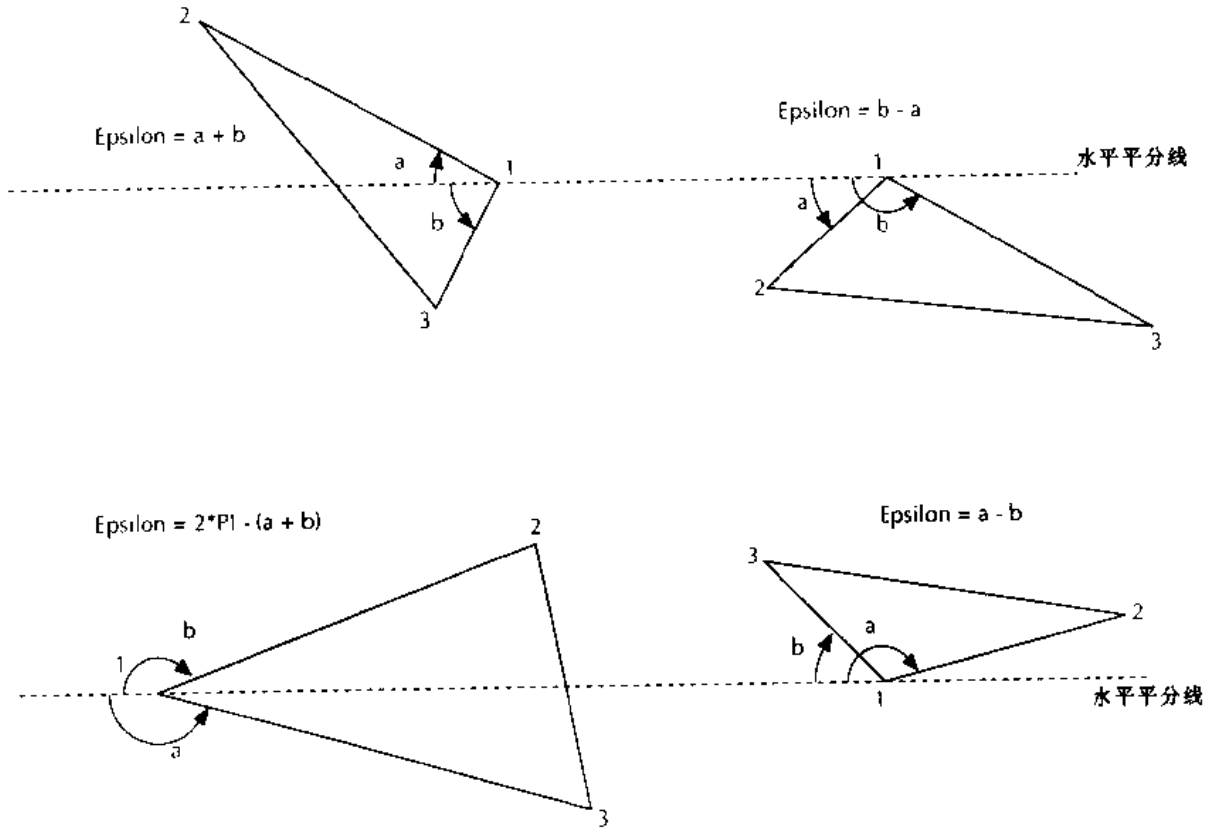


图 5.10 顶点位置和平分线位置的组合

图 5.14 给出了对一对简单几何图形——"box2"和"V"的模拟结果。正如所看到的，所有的用于测试网络的三角形都能被正确分类(输出为 -1)。然而，"box"中的两个图形——"box"和"box.270"被错误分类。

5.3.5 实际应用

Duren 和 Peikari[1991]提出一个平移和方向不变的二阶神经网络，并且此神经元的特性不同于感知器。对于二阶网络(见图 5.6(b))更为特殊的是，神经元的输出变为：

$$O = f\left(\sum_{i=0}^{N-1} w_i x_i + \sum_{j=0}^{N-1}\sum_{k=0}^{N-1} w_{jk} x_j x_k\right) \tag{5-5}$$

通过将权值限制为输入之间距离的函数已达到平移不变。对于手写数字识别的一个实验，据报道它的成功识别率是 94.8%。

Lisboa 和 Perantonis[1991]用三阶神经网络来识别手写体数字；因此，此例中是每三个象素一组进行处理。我们所需的是旋转和比例不变。使用的模式是 20×20 象素位图格式的手写数字。旋转和比例的测试结果都很显著。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

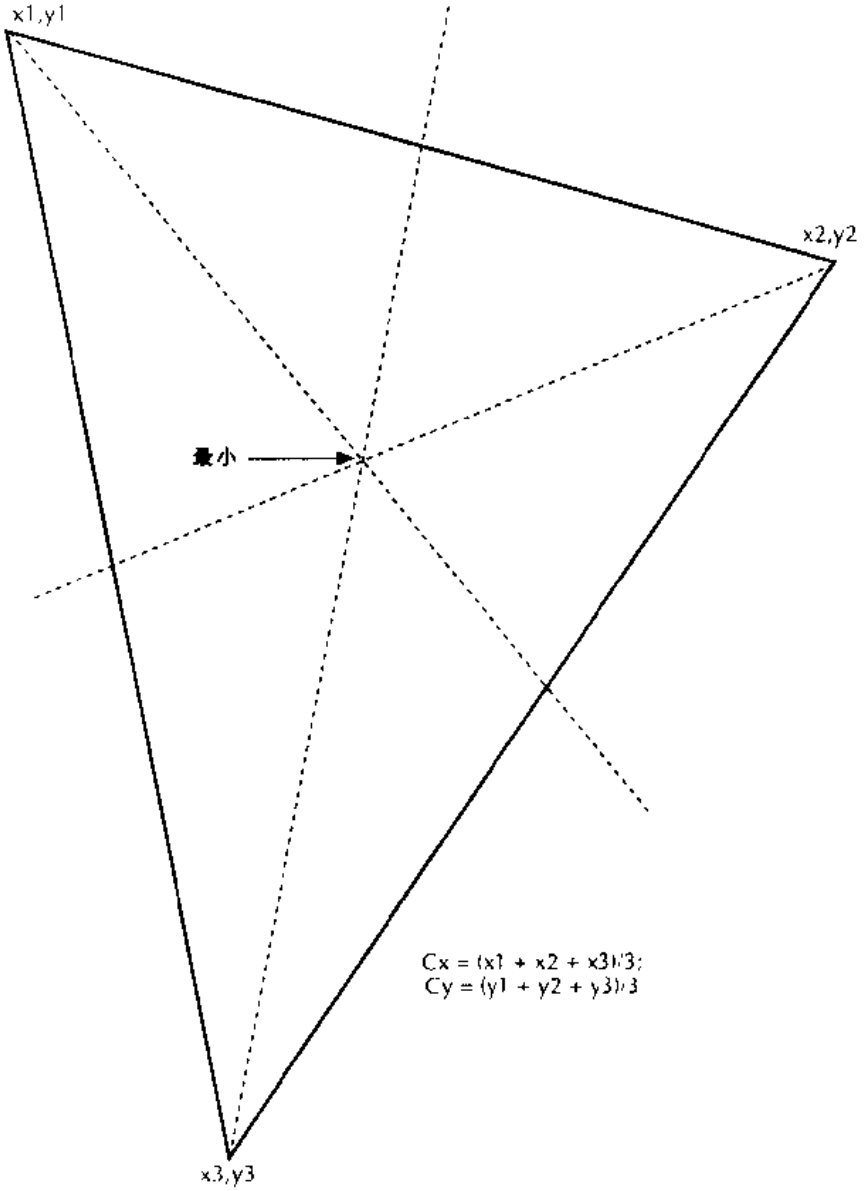


图 5.11 三角形的质心

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

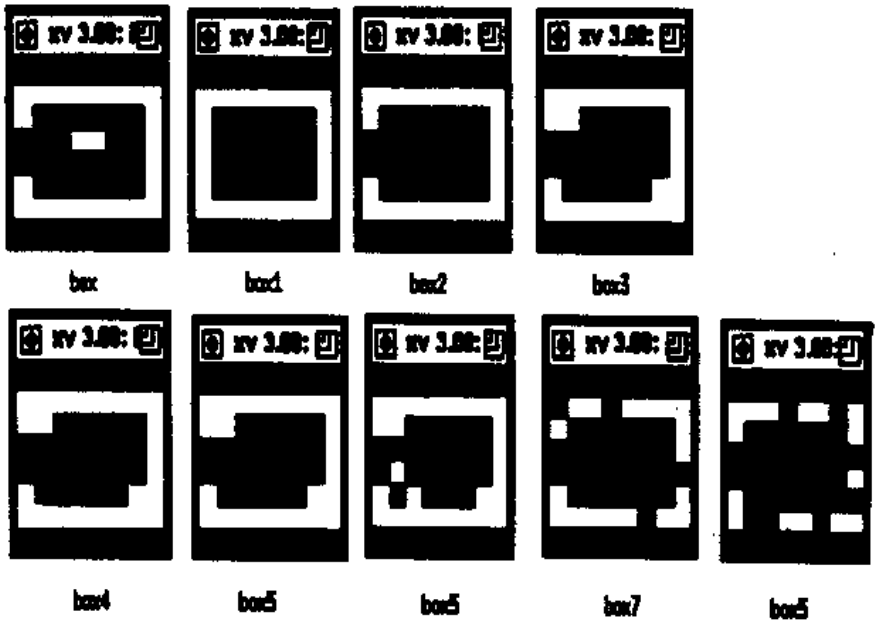


图 5.12 含有噪声的四边形

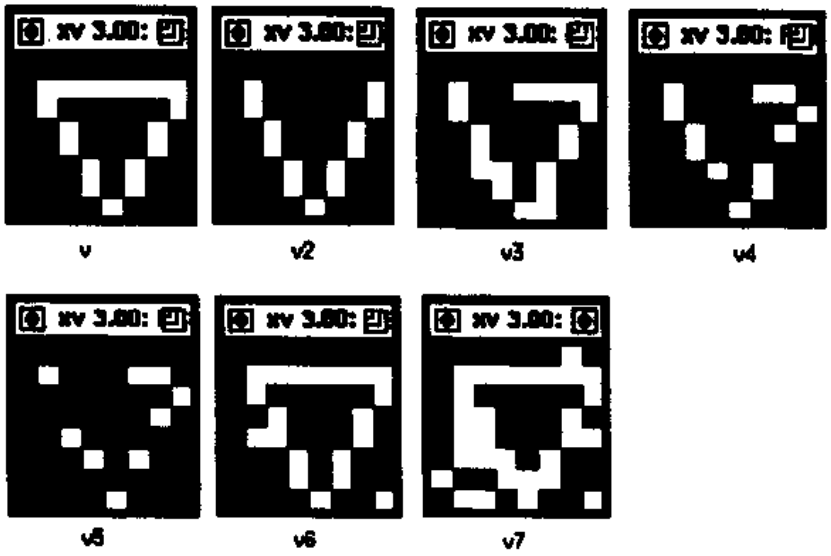


图 5.13 含有噪声的三角形

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

angle resolution : 4 degrees
weight initialization : RANDOM
activation function : HYPERBOLIC TANGENT
target ontputs : +1 , -1
error threshold : 0 .0000001
learning rate : 0 . 1
training image 1 : box2 (o/p : +1)
training image 2 : v (o/p : -1)

Image Name	Network Output	Image Name	Network Output
box	-1	v	-1
box.270	-1	v.left	-1
box1	1	v.90	-1
box1.90	1	v.180	-1
box2	1	v.270	-1
box2.90	1	v2	-1
box2.180	1	v2.90	-1
box3	1	v2.180	-1
box3.90	1	v2.270	-1
box3.180	1	v3	-1
box3.270	1	v3.90	-1
box4	1	v3.180	-1
box4.90	1	v3.270	-1
box4.180	1	v4	-1
box5	1	v4.90	-1
box5.90	1	v4.180	-1
box5.180	1	v5	-1
box6	-1	v5.90	-1
box6.90	-1	v5.180	-1
box6.180	1	v6	-1
box7	1	v6.90	-1
box7.90	1	v6.180	-1
box7.180	1	v7	-1
box8	1	v7.90	-1
box8.90	1	v7.180	-1
box8.180	1		

图 5.14 对于简单图形的模拟结果

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
communication & cooperation: luzhenbo@yahoo.com.cn

参考书与文献

Broomhead, D. S. and Lowe, D., "Multivariable functional interpolation and adaptive networks," *Complex Syst.*, vol. 2, pp. 321–355, 1988.

Cover, T. M., "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," *IEEE Trans. Electron. Comput.*, EC-14, pp. 326-334, 1965.

Duda, R. O. and Hart, P. E., *Pattern Classification and Scene Analysis*," John Wiley & Sons, New York, 1973.

Duren, R. and Peikari, B., "A comparison of second order neural networks to transform based methods for translation and orientation invariant object recognition," Neural Networks for Signal Processing, Proc. of the IEEE Workshop, pp. 236–245, 1991.

Fukushima, K., "Character recognition with neural networks," *Neurocomputing*, vol. 4, pp. 221-233, 1992.

Giles, G. L. and Maxwell, T., "Learning, invariances and generalizations in high-order neural networks," *Appl. Opt.*, vol. 26, pp. 2972-2978, 1987.

Haykin, S., *Neural Networks, A Comprehensive Foundation*, MacMillan College Publishing, New York, 1994.

He, X. and Lapedes, A., "Nonlinear modeling and prediction by successive approximation using radial basis functions," Technical report LA-UR-91–1375, Los Alamos National Laboratory, NM, 1991.

Kadirkamanathan, V., Niranjan, M., and Fallside, F., "Neural networks and radial basis functions in classifying static speech patterns," *Comput. Speech Language*, vol. 4, pp. 275–289, 1990.

Kadirkamanathan, V., Niranjan, M., and Fallside, F., "Sequential adaptation radial basis function neural networks." *Adv. Neural Inf. Process. Syst.*, vol. 3, pp. 721–727, 1991.

Kung, S. Y., *Digital Neural Networks*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

Lee, Y., "Handwritten digit recognition using K nearest-neighbor, radial basis function, and backpropagation neural networks," *Neural Computation*, no. 3, pp. 440–449, 1991.

Lippman, R. P., "Pattern classification using neural networks," *IEEE Commun. Mag.*, vol. 27, no. 11, pp. 47–64, 1992.

Lisboa, P. J. G. and Perantonis, S. J., "Invariant pattern recognition using third-order networks and zernike moments," *IEEE Int. Joint Conf. Neural Networks (IJCNN) 91*, pp. 1421–1425, 1991.

Lowe, B. and Webb, A. R., "Exploiting prior knowledge in network optimization: an illustration from medical prognosis," *Network*, vol. 1, pp. 291–323, 1990.

Minsky, M. L. and Papert, S., *Perceptrons*, MIT Press, Cambridge, MA, 1969.

Moody, J. E. and Darken, C. J., "Fast learning in networks of locally-tuned processing units," *Neural Comput.*, Vol. 1, pp. 281-294, 1991.

Musavi, M. T., Faris, K. B., Chan, K. H., and Ahmed, W., "On the implementation of RBF technique in neural networks," ACM ANNA — 91, *Anal. Neural Network Appl.*, pp. 110–115, 1991.

Nirangan, M. and Fallside, F., "Neural networks and radial basis functions in classifying static speech patterns," *Comput. Speech Lang.*, vol. 4, pp. 275-289, 1990.

Ng, K. and Lippman, R. P., "Practical characteristics of neural networks and conventional pattern classifiers, *Adv. Neural Inf. Process. Syst.*, vol. 3, pp. 970–976, 1991.

Poggio, T. and Giorgi, F., "Regularization algorithm for learning that are equivalent to multilayer networks, *Science*, Vol. 247, pp. 978- 982, 1990.

Poggio, T. and Edelman, S., "A network that learns to recognize three-dimensional objects," *Nature*, vol. 343, pp. 263–266, 1990.

Renals, S., "Radial basis function network for speech pattern classification," *Electron. Lett.*, vol. 25, pp. 437-439, 1989.

Rosenblatt, F., Principles of Neurodynamics, Spartan Books, New York, 1962.

Rumelhart, D. E., *Neural Networks*, vol. 2, pp. 348–352, 1989.

Saha, A., Christian, J., Tang, B. S., and Wu, C. L., "Oriented non-radial basis functions for image coding and analysis," in ...analysis," in *Advances in Neural Information Processing Systems, Vol. 3*, R. P. Lippmann, J. E. Moody, and D. S. Touretzeky (Eds.), Morgan Kaufmann Press, San Mateo, CA, pp. 728–734, 1991.

Spirkovska, L. and Reid, M.B., "Coarse-coded higher order neural networks for PSRI optic recognition," *IEEE Trans. Neural Networks*, vol. 4, no. 2, pp. 276-283, 1993.

Troxel, S. E., Rogers, S. K., and Kabrisky, M., "The use of neural networks in PSRI recognition," in Proc. Joint Int. Conf. Neural Networks, San Diego, CA, July 24–27, pp. 593–600, 1988.

Wong, Y., "How radical basis functions work," Proc. Int. Joint Conf. Neural Networks, Vol. 2, Seattle, WA, pp. 133-138, 1991.

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

第六章 特征提取 I:几何特征和变换

6.1 概 述

在某种意义上说,许多神经网络能自动提取特征。例如,使用反向传播算法训练后的前馈网络的隐含层节点就可看作是提取特征,这些特征最终将在输出层被分类。如果有多个隐含层,随着层数的增加,神经元所提取的特征就越为复杂。而且,我们不选择网络需进行训练的特征,因为这些特征可能有或可能没有所希望的特性。它们是一些抽象的特征,不同于线段、环、圆弧这些人们容易识别的特征。因此,我们现在希望选取显而易见的特征及特征提取来完成特定的目的。第一种想法就是,特征提取是为了降低维数。

当输入是原始的位图格式时,有足够强的辨别能力能对实际问题进行识别的全互连网络,可能有许多参数要推广。当希望保存它的突出特征时,慎重地选择特征来降低输入矢量的维数是提高推广能力的一种方法。因此,应按第四章讨论的那样减小所需训练集合的大小。本章将讨论几个提取和使用这样特征的方法。我们首先对直观的几何特征进行提取。然后,阐述它们在前馈多层感知器网络中的使用方法。另外一种方法就是在特征映射中使用特征。这种方法包括将网络的头几层限制为局部的,由此来确保这些神经元中能检测到局部的特征(Le Cun 等,1990)。特征映射在 6.3 节中讨论。

慎重选择特征的另外一个优点是可以获得所需的一些形式的不变量。利用不变量,输入可以以一些特定的方式改变,而不影响识别结果。我们常常需要对于旋转、平移和比例变化都具有不变性的表示方法。我们将阐述怎样选取这样的特征和利用变换技术来降低维数。

很明显,特征的选取和特征的质量,将对识别系统的特性有较深远的影响。在特征识别问题中,选取不同的特征集合能使识别器对不同的字符产生二意性[Wang 和 Jean,1993]。随后在第十二章,当我们讨论多识别系统和混淆矩阵时,再讨论这个问题。

由于以上及其它原因(包括特征选取对神经网络大小和性能方面的影响),特征提取是模式识别中一个很重要的部分。我们将深入探讨应用于一个识别器网络的特征选择和提取。

6.2 几何特征(环、交叉点、端点)

一种特征提取的方法是使用从直观上人们容易理解的特征。环、交叉点和端点都是这样一些特征。在手写体字符识别方面,这么小的几何特征集合就能给出令人惊讶的良好识别效果[Darwiche 等,1992]。在这里的字符识别问题中,我们只使用了环、交叉点、端点这些特征。并且我们有意使字符的空间分辨率很低,而且没有使用交叉点和端点间的连结信息。

6.2.1 交叉点和端点

下面是检测交叉点和端点的方法。在这些方法中,都是假设对于骨架处理后的位图格式

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication information: luzhenbo@yahoo.com.cn

进行处理。3.5 节中已讨论过骨架处理的过程。下面的讨论中，我们将“on”的象素看作 1-象素，“off”象素看作 0-象素。

在骨架处理后的位图格式图像中，我们通过检查所有的单个 1-象素点来寻找端点。作为骨架处理后的结果，在端点的 8 邻域内有一个并且仅有一个象素点为 1-象素。因此，可以对一点的 8 邻域内的所有点求和，当求和结果为 1 时就认为它是端点。

我们可以以类似的方式来寻找交叉点。图像中的每个 1-象素都被检测，并且对中心象素邻接的 1-象素点进行计数（例如，对 8 邻域的象素点进行检测）。如果计数的结果 n 超过 2(n>=3)，那么，中心的象素点被认为是一个交叉点。

现在，我们将提供一程序代码例子来阐明交叉点和端点的检测。清单 6.1 提供了特征提取类，cThinner 的类定义如下：

清单 6.1

```
struct tgrid {
  int x;
  int y;
};

struct tLoop {
  int x;              // Center of mass X coord
  int y;              // Center of mass Y coord
  int m;              // Total Mass (Useful to disqualify ornamentation)
};
class cThinner {
private:
 int   xMax, yMax;                    // Size of the bitmap
 int   PelxMax, PelyMax;
 int   PelxMin, PelyMin;
 int   M[32][32];                     // Space for the whole bitmap
 int   Stage1[32][32];                // orig image archive
 int   W[32][32];                     // Work Space save area
  .
  .
  .
 tMap   Map[7];
 int   isEndPt(int, int);
 int   isIntersect(int, int);
 int   isIntersect2(int, int);
 int   EndPtCnt;                      // number of end points
 tgrid  EndPoint[324];                // endpoint coords
 int   IntPtCnt;                      // number of intersects
 tgrid  IntersectPoint[324];          // intersect coords
 int   LoopCnt;
 tLoop  LoopLocation[324];
 int ValidZeroChild(int, int, int, int);
 void Frame2Ones();
 void NQZeroes(coordq *, tgrid);
 void KillOneLoop(int&, int&, int&);  // deal w/ 1 loop

public:
 cThinner(); //Constructor
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
  void   SetGridSize(int Mx, int My){xMax=Mx; yMax=My;}
  void   Store(int x,int y,int Pel){Stage1[x][y]=M[x][y]=Pel;}
  int    QueryPel(int x,int y){return M[x][y];}
  void   Thin2();
    .
    .
    .
  void   LocateEndPts();
  void   LocateIntersects();
  int    QueryEndCount(){return  EndPtCnt;}
  int    QueryIntCount(){return  IntPtCnt;}
  tgrid  QueryEndPoint(int n);
  tgrid  QueryIntPoint(int n);
  void   Grid2Work();                    //save pixel map in work buffer
  void   Work2Grid();                    //save pixel map in work buffer
  int    isLoop();                       //1= there are loops 0=no loops
  void   LocateLoops();  //find all independent loops. (work buf is left w/ Endpoints excised)
  int    QueryLoopCnt(){return LoopCnt;}
  tLoop  QueryLoop(int k){return LoopLocation[k];} //returns info re Nth loop
  int    FindNeighbor(int&, int &);
  void   Kill1Path(int);
  void   KillEndPaths();
  void   IsolatedPelCleanup();
  int    isAnyPelOn();
  int    isAnyPelOff(int&, int&);
 };
```

与检测端点和交叉点最为相关的类 cTinner 方法，在清单 6.2 中可以找到。

清单 6.2

```
void cThinner::LocateEndPts() {
  int x,y,i,j;
EndPtCnt=0;
for (y=0; y<yMax; y++) {
  for (x=0; x<xMax; x++) {
    if ((M[x][y]==1) && isEndPt(x,y)) {
      EndPoint[EndPtCnt].x=x;            // we have an endpoint so record it.
      EndPoint[EndPtCnt].y=y;
       EndPtCnt++;
       } /* endif */
    } /* endfor */
  } /* endfor */
}

void cThinner::LocateIntersects() {
  int x,y,i,j;
IntPtCnt=0;
for (y=0; y<yMax; y++) {
  for (x=0; x<xMax; x++) {
    if ((M[x][y]==1) && isIntersect(x,y)) {
      // we have an intersection point so do something with it.
      IntersectPoint[IntPtCnt].x=x;
      IntersectPoint[IntPtCnt].y=y;
       IntPtCnt++;
       } /* endif */
    } /* endfor */
  } /* endfor */
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2

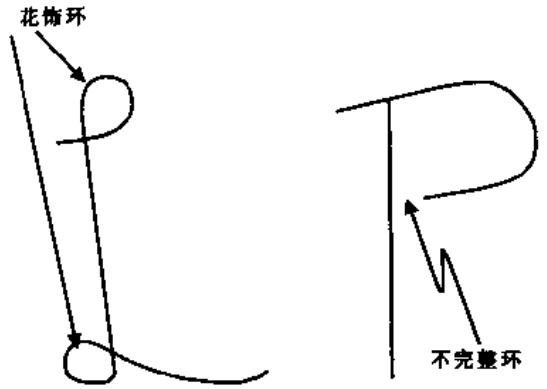
Communication & Question: luzhenbo@yahoo.com.cn

```
}

tgrid  cThinner::QueryEndPoint(int n){
  tgrid RetVal;
RetVal.x= EndPoint[n].x;
RetVal.y= EndPoint[n].y;
return RetVal;
}
tgrid  cThinner::QueryIntPoint(int n){
  tgrid RetVal;
RetVal.x= IntersectPoint[n].x;
RetVal.y= IntersectPoint[n].y;
return RetVal;
}
```

6.2.2 环

有许多检测环的方法。例如在 Darwiche[1992]中，环的检测是基于字符中圆环或椭圆形状的出现。基于这一点，我们描述两个检测环方法，每种方法都有其优点。第一个方法是建立一个字符树，因此也就建立了字符的连结性；第二个方法是在环中建立一个象素成员表，这样能允许我们计算环的“质心”——另一个相关的字符量度。

这里有同特征环相关联的两个问题：(1)断的或不完全的环和(2)花饰环。图 6.1(a)给出了这种情况的例子。随后我们讨论环时，再讨论这个问题。第二个例子中，环很小并且花饰部分对于识别并不突出。图 6.2(b)中给出了这一情况下的例子。



(a)花饰环和　　(b)不完整的环

图 6.1　(a)花饰环和(b)不完整的环

注意，一旦检测环方法已建立一个成员表，我们通过对所有成员象素求和，就能计算每个环的总质量。即每一个象素被认为有一个单位质量，然后建立一阈值，T。因此，如果环的总质量低于阈值的话，这个环被认为是花饰环。

对于已确定为花饰性的环的可行处理方法，是对这些环进行 1－填充(即把环内的所有的象素值设为 1)，然后再重新对字符细化。

环能从一个位图格式的图像中提取出来，并且它可以用骨架的轮廓线或连结端点和交叉

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xaut.edu.cn

点的路径来描述。在此过程中，建立了字符连接树，树的节点是端点和交叉点，边是连结节点的已标记象素路径。为了便于讨论，假设将字符合并，这样字符中所有象素都互相连结了。此例中，如果交叉连结的数目 I 和端点数目 E 都是 0，那么此字符含有单个环。否则，我们依次看一下每一个交叉点。所有与交叉点 I_k 相关的环，都可以通过构造以 I_k 为根的树来建立。此树按以下方式来建立，根的中心节点是 I_k，对于每一个节点建立一系列子节点，这些子节点含有从父中心节点的所有可能路径。这样的路径是沿着一条轨迹直到轨迹到达第一交叉点或端点为止，已经经过的边和节点(根节点除外)应该去除。一个子树的中心是象素轨迹的目的节点。随着每一个子树的建立，也就标记了边。含有根节点的终端节点，就构成了环。沿着确定为环的终端节点回到根节点，也就能确定环的路径。这个过程完成后，可归为其它两个或更多环的重复环须被去除。只要环 L_1、L_2 合在一起能覆盖第三个环 L_3 所有的象素路径，那么如果 L_1 内的任意一点和 L_2 内的任意一点同时也是 L_3 内部的点，L_3 应被去除。

图 6.2 和图 6.3 分别给出了用上面描述的方法建立字符“R”和“8”的树。

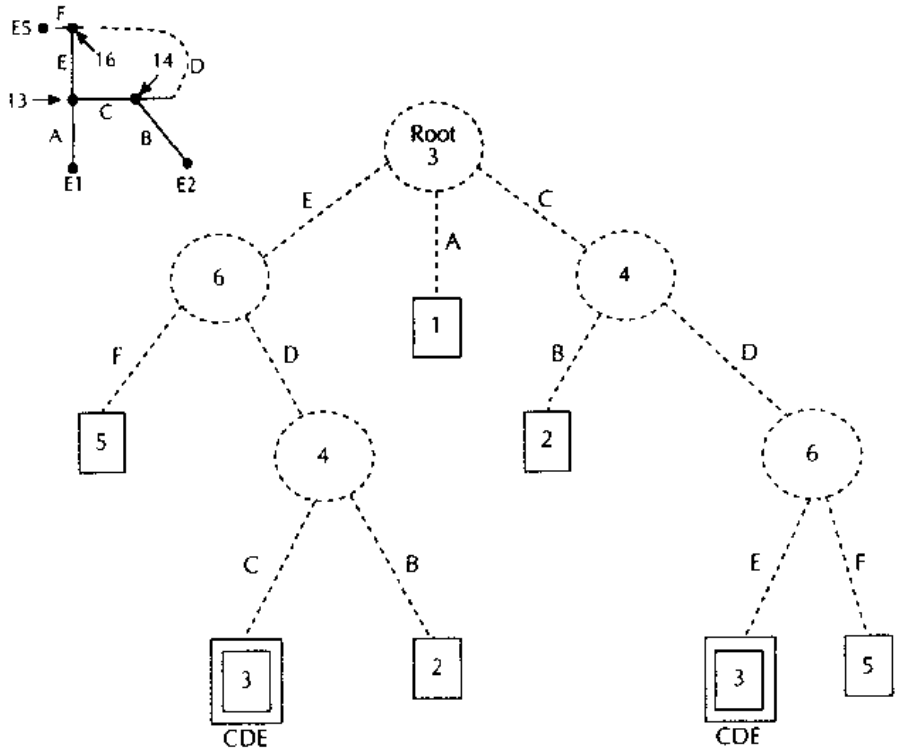


图 6.2 字母“R”的环树

应特别注意的是：在通过象素路径的过程中，可采用其它一些可能的、潜在的有用措施。例如，为了详细描述字符形状，可沿着边以一定的间隔标记一定数量的中间点。

我们现在离开主题，简略讨论不完全环。假设在系统中在每个端点的周围我们建立一个最小的阈半径。如果由阈半径定义的圆中包括一个象素或多个象素与中心点不相连，那么我们就构造一象素桥连到最近的不相连象素(如果已知歧点的位置，由此也应当这样做)。尽管很容易找出一合适的手写体字符例子，在此字符中不完全环存在于不是端点或歧点的点上。然而此过程使得前面讨论的环检测技术，对于许多通常的不完全环的检测也很有效。

环也可通过建立成员表来进行检测。首先过程从字符中删除所有终止在端点上的边开始。如果一个 1-象素也没剩下，则不存在环，此过程结束；否则，至少有一个环存在。我们继

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

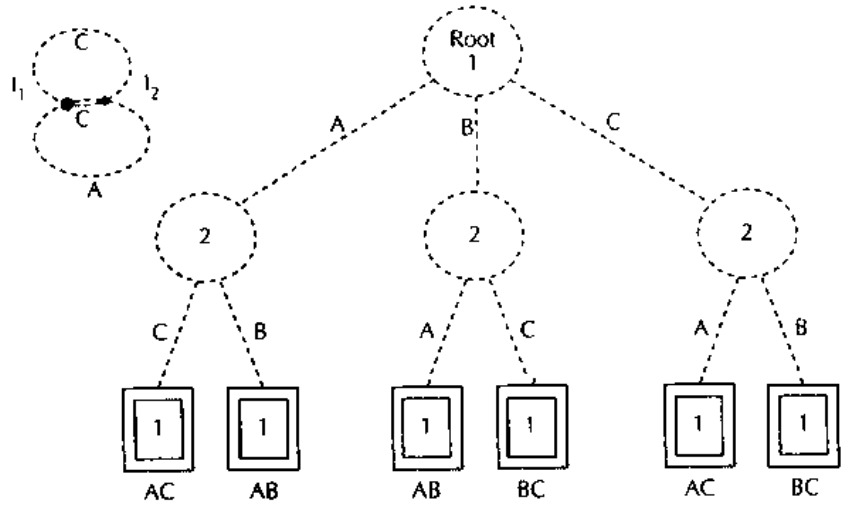


图 6.3 数字"8"的环树

续建立一个包含所有的0-象素点的表 Q_0(除了那些与位图边界相邻的点)。Q_0 中每一个象素都在某些环内。我们也可以从 Q_0 中移走任一象素,建立一个此点邻域的所有的0-象素点的集合,这个集合不能超出由1-象素点形成的边界。此集合是单个环 L 的成员。L 所有的成员象素点都从 Q_0 中移走,直到 Q_0 空了为止。因此字符内所有的环都已被检测出来。基于类 **cThinner**(见清单6.1)定义的使用,这种环检测技术的方法由清单6.3给出。类"coordp"和"coorqel"共同在算法中起到保存顺序的作用,也在清单6.3给出。

现在,我们将研究位图中一个环的几何位置。其中的一个方法是计算它的质心。N 个质量的质心(二维)计算方法如下:

$$x_{cm} = \frac{\sum_{i=1}^{N} m_i x_i}{\sum_{i=1}^{N} m_i}, y_{cm} = \frac{\sum_{i=1}^{N} m_i y_i}{\sum_{i=1}^{N} m_i} \tag{6-1}$$

N 是环中象素点的数量,x_i 和 y_i 是质量 m_i 的坐标。

我们的例子中,每个质量 m_i 都是具有单位质量的象素点(是一个环的成员),因此环的质心公式(6-1)变成如下:

$$x_{cm} = \frac{1}{N}\sum_{i=1}^{N} x_i, y_{cm} = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{6-2}$$

它只是简单的均值。

清单 6.3

```
//-------------------------------------------------------------------
class coordqel {
private:
 tgrid   coord;
 coordqel *next;
public:
 coordqel(){next=NULL;}
 void setnext(coordqel *p){next=p;}
 void setxy(int x,int y){coord.x=x; coord.y=y;}
 void query(int& x, int& y){x=coord.x; y=coord.y;}
 coordqel *querynext(){return next;}
};
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
//-----------------------------------------------------------------------
class coordq {
private:
 coordqel *head;
 int     count;
public:
 coordq(){head=NULL;}
 void enQTop(int x, int y);
 void enQTop(tgrid pel){enQTop(pel.x, pel.y);}
 int  deQTop(int& x, int& y);
 int  Qdelete(coordqel *p);
 int  find(int x, int y);
 int  DatAvail();
};

int coordq::DatAvail(){
if (head==NULL)
  return 0;
 else
  return 1;
}

int coordq::find(int x, int y){
  int x1,y1;
  coordqel *tmp;
tmp=head;
while (tmp!=NULL) {
 tmp->query(x1,y1);
 if ((x==x1) && (y==y1))
    return 1;
  else
    tmp=tmp->querynext();
  } /* endwhile */
 return 0;
}

int coordq::Qdelete(coordqel *p){
  coordqel *cur;
  coordqel *prev;
  cur=head; prev=NULL;
   while ((cur!=NULL) && (cur!=p)) {
     prev=cur;
    cur=cur->querynext();
    } /* endwhile */
   if (cur!=NULL) {
    if(prev!=NULL)
     prev->setnext(cur->querynext());
     else
      head=cur->querynext();
    delete cur;
    return 1;
    }
   else
    return 0;                    // bad pointer, Q empty or just not found. delete failed.
}
 void coordq::enQTop(int x, int y){
  coordqel *tmp;
  tmp=new coordqel;
  tmp->setxy(x,y);
  tmp->setnext(head);
  head=tmp;
}
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication E-mail: luzhenbo@yahoo.com.cn


```
int coordq::deQTop(int& x, int& y){
 coordqel *tmp;
 if(head!=NULL) {
  head->query(x,y);
   tmp=head;
  head=head->querynext();
   delete tmp;
   return 1;
   }
  else
   return 0;
}

//-------------------------------------------------------------

//*********************************************************
// Copy fns between pixel map & temporary work space.   *
//*********************************************************
void cThinner::Grid2Work(){
int x,y;
for (x=0; x<xMax; x++) {
  for (y=0; y<yMax; y++) {
    W[x][y]=M[x][y];
    } /* endfor */
  } /* endfor */
}

void cThinner::Work2Grid(){
int x,y;
for (x=0; x<xMax; x++) {
  for (y=0; y<yMax; y++) {
    M[x][y]=W[x][y];
    } /* endfor */
  } /* endfor */
}

//*********************************************************
//Deletes pel iff all surrounding pells are zero
//*********************************************************
void cThinner::IsolatedPeldeanup() {
int x,y;
for (x=0; x<xMax; x++) {
  for (y=0; y<yMax; y++) {
    if (M[x][y]==1){                          // faster than a for loop
      if ( (M[x-1][y-1]==0) && (M[x-1][y]==0) && (M[x-1][y+1]==0) && (M[x][y-1]==0)
        && (M[x][y+1]==0) && (M[x+1][y-1]==0) && (M[x+1][y]==0) && (M[x+1][y+1]==0) )
          M[x][y]=0;
       }
    } /* endfor */
  } /* endfor */
}

//*********************************************************
// Returns 1 if there is a nearest neighbor 1 pel to x,y *
//         0 otherwise                                   *
// also modifies x,y to be the neighbors coord           *
//*********************************************************
int cThinner::FindNeighbor(int& x, int& y){
 int i,j;
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
for (i=0; i<3; i++) {
  for (j=0; j<3; j++) {
    if (M[x-1+i][y-1+j]==1) {
      x=x-1+i; y=y-1+j;
       return 1;
       } /* endif */
    } /* endfor */
  } /* endfor */
return 0;
}

int cThinner::isAnyPelOn(){
int x,y;
for (x=0; x<xMax; x++) {
  for (y=0; y<yMax; y++) {
    if (M[x][y]==1) return 1;
     } /* endfor */
  } /* endfor */
return 0;
}

int cThinner::isAnyPelOff(int& x, int& y){
for (x=0; x<xMax; x++) {
  for (y=0; y<yMax; y++) {
    if (M[x][y]==0) return 1;
     } /* endfor */
  } /* endfor */
return 0;
}

//*********************************************************
// Starting at a given endpoint eliminate all pixels until *
// either:                                                 *
//      A) Another endpoint is encountered (inclusive)*
//   or                                                    *
//      B) An Intersect is encountered  (exclusive)   *
//*********************************************************
void cThinner::Kill1Path(int EPindx){
  int x,y;
  int Done;
x = EndPoint[EPindx].x;
y = EndPoint[EPindx].y;
Done=0;
while (!Done) {
  M[x][y] = 0;                          // kill the current pel
  if (!FindNeighbor(x,y)) Done =1;      //No neighbor found...we're done
  if (isEndPt(x,y)) {                   //check whether x,y is an endpoint
    M[x][y]=0;                          //endpoint<->endpoint delete is inclusive
    Done=1;
     }
  if (isIntersect2(x,y)) Done=1;
  } /* endwhile */
}

void cThinner::KillEndPaths(){
  int i;
for (i=0; i<EndPtCnt; i++) {
  if(M[EndPoint[i].x][EndPoint[i].y])
     Kill1Path(i);
  } /* endfor */
}
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication E-mail: luzhenbo@yahoo.com.cn


```
int cThinner::isEndPt(int x, int y) {
 int sum,i,j;
sum=-1;                                  //account for centrer pixel being on
for (j=y-1; j<=y+1; j++) {
  for (i=x-1; i<=x+1; i++) {
    if((i>=0) && (j>=0) &&(i<xMax) && (j<yMax))
      sum+=M[i][j];                      // count the neighbors
    } /* endfor */
  } /* endfor */
if (sum==1)                              // Exactly one neighbor ON
  return 1;                              // defines an endpoint.
return 0;
}

int cThinner::isIntersect(int x, int y) {
 int sum,i,j;
sum=-1;                                  //account for center pixel being on
for (j=y-1; j<=y+1; j++) {
  for (i=x-1; i<=x+1; i++) {
    if((i>=0) && (j>=0) && (i<xMax) && (j<yMax))
      sum+=M[i][j];                      // count the neighbors
    } /* endfor */
  } /* endfor */
if (sum>=3) {                            // three or more neighbors 'ON'
  return 1;                              // defines an intersection.
  }/* endif */
return 0;
}

void cThinner::LocateLoops() {
  int x,y,m;
LoopCnt=0;
Grid2Work();                             //Save copy of orig
while (QueryEndCount()>0){
LocateEndPte();                          //Calc Endpoints
LocateIntersects();                      //  and intersects
KillEndPaths();                          //Remove the pixel-path
Thin2();                                 //Re-Thin
 }
IsolatedPelcleanup();                    // Remove pels w/ all zero neighbor
if (!isAnyPelOn()) {
 LoopCnt=0;                              //No loops detected
 }
else {
 Frame2Ones();
 if (QueryIntCount()==0) {
    LoopCnt=1;                           //There is exactly 1 loop
    isAnyPelOff(x,y);
    // Now locate the loops
    KillOneLoop(x,y,m);                  // kill the pel and all its friends
    LoopLocation[0].x=x;
    LoopLocation[0].y=y;
    LoopLocation[0].m=m;
     }
  else {
  //Here there are multiple loops, so establish how many & locate them
  LoopCnt=0;
   while (isAnyPelOff(x,y)) {
     KillOneLoop(x,y,m);                 // kill the pel and all its friends
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@163.com


```
      LoopLocation[LoopCnt].x=x;
      LoopLocation[LoopCnt].y=y;
      LoopLocation[LoopCnt].m=m;
       LoopCnt++;
       } /* endwhile */
     } /* endif */
  } /* endif */
Work2Grid();                          //Restore orig grid
LocateEndPts();                       //Played with these so...
LocateIntersects();                   // ...they need restoration too
}

int  cThinner::isIntersect2(int x, int y){
int found,i;
found=0;
for (i=0; i<IntPtCnt; i++) {
  if ( (IntersectPoint[i].x==x) && (IntersectPoint[i].y==y))
     found=1;
  } /* endfor */
return found;
}

void cThinner::Frame2Ones(){
  coordq f21;
  tgrid  fpoint;
  int x,y;
fpoint.x =0;
fpoint.y =0;
NQZeroes(&f21,fpoint);
while (f21.DatAvail()) {
  f21.deQTop(x, y);
  M[x][y]=1;
  } /* endwhile */
}

int cThinner::ValidZeroChild(int cx, int cy, int px, int py){
if ((cx<0) || (cy<0) || (cx>17) || (cy>25)) // stay on grid
        return 0;
if (M[cx][cy]!=0)                     // Pel must be zero
        return 0;
if ((cx==px) || cy==py)               // no x-ings possible for
        return 1;                     // directly hor/vert neighbor
if ((M[cx][py]==1) && (M[px][cy]==1))  // Check diagonal neighors
        return 0;
return 1;
}

void cThinner::NQZeroes(coordq *targQ, tgrid seedpel){
  coordq open;
  int x,y,i,j;
if (M[seedpel.x][seedpel.y]==0) open.enQTop(seedpel);
      else return ;                   //exit if seed nonzero
while (open.DatAvail()) {
  open.deQTop(x, y);
  targQ->enQTop(x,y);
  // Now examine each NN of pel at (x,y) ...
  // ...if its valid & not already present we'll  place it on open.
  for (j=-1; j<=1; j++) {
    for (i=-1; i<=1; i++) {
      if (!((i==0) && (j==0)) ) {     //each NN visited here
        if(ValidZeroChild(x-i,y-j,x,y)){
          if ((open.find(x-i,y-j)==0) && (targQ->find(x-i,y-j)==0)){
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
            open.enQTop(x-i,y-j);
             }/* endif */
           }/* endif */
         } /* endif */
       } /* endfor */
     } /* endfor */
     } /* endwhile */
   }

   void cThinner::KillOneLoop(int& x, int& y, int& m) {
     coordq targQ;
     tgrid  seedpel;
     int sx,sy;
   m=0;
   seedpel.x=x;
   seedpel.y=y;
   sx=sy=0;
   NQZeroes(&targQ, seedpel);          //all interior loop-pels to targq
   while (targQ.DatAvail()) {          //while they last
     targQ.deQTop(x, y);               //examine them 1 by 1
     sx+=x;                            // to determine loop
     sy+=y;                            //  coords and
     m++;                              //    mass
     M[x][y]=1;                        // and delete
     } /* endwhile */
   if (m>0){
     x=sx/m;
     y=sy/m;}
   }
```

在调用程序中使用的几何特征检测方法,由清单 6.4 给出。注意,在应用特征检测方法之前,必须先调用字符细化程序。第三章中讨论了字符细化。确定特征之后,属于每个特征类型的点的位置和数量能被查询并且画出。

清单 6.4

```
cThinner      Thinner;

int main(int argc, char *argv[]) {
  int        mx,my,i,j,d,chIndx, Lcnt;
  tgrid fpt;
  int fcnt;
  tLoop Lpt;
    .
    .
    .
  Thinner.Thin2();                    // First Thin the character
    .
    .
  Thinner.LocateEndPts();             // Build a list of all End Points
  Thinner.LocateIntersects();         // Build a list of all intersections
  Thinner.LocateLoops();              // Build a list of all loops

  fcnt= Thinner.QueryEndCount();      //Display any endpoints
  printf("\n  End Points:");
  for (i=0; i<fcnt; i++) {
    fpt=  Thinner.QueryEndPoint(i);
    printf("[%d,%d]", fpt.x,fpt.y);
    } /* endfor */
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo3
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
fcnt= Thinner.QueryIntCount();          //Display any intersections
printf("\n  Intersects:");
for (i=0; i<fcnt; i++) {
 fpt = Thinner.QueryIntPoint(i);
 printf("[%d,%d]", fpt.x,fpt.y);
 } /* endfor */
Lcnt=Thinner.QueryLoopCnt();            //Display any loops
if (Lcnt==0) {
  printf("\nNo loops detected.");
} else {
  printf("\n  Loops:    ");
  for (i=0; i<Lcnt; i++) {
   Lpt = Thinner.QueryLoop(i);
   printf("[m=%d,(%d,%d)]", Lpt.m,Lpt.x,Lpt.y);
   } /* endfor */
  printf("\n");
} /* endif */
return 0;
}
```

我们在对端点、交叉点和环的检测详细论述的同时，值得一提的是，这些特征仅仅是那些已发现的和已经被使用的特征的一小部分。为了提供更完整的形状描述，区别线和弧也是很有用的；另外拐点和歧点在表示数字曲线形状方面也是有意义的，并且是有益的。对于检测拐点和歧点的更多的讨论，请看(Pikaz 和 Dinstein [1994])。

6.3 特 征 映 射

此外有另一种寻找几何特征的方法。特征映射采用的思想是，如果在图像中的某一个点上的局部特征是很有用的，那么同样在一些其它区域可能也是很有用的。如果，一个畸变字符含有突出的特征，它可以由一个典型字符的特征稍微平移就能得到，因此特征映射在其检测中将很有用。

特征映射由神经元平面组成，我们将神经元的权值限制为相等。这样，某一特征映射内，每一个神经元执行相同的操作，也就是说它们检测同一特征。然而，每一个神经元从图像的不同部分接收输入量。某一神经元从图像中接收信息的区域或部分，称为神经元的感受野。图 6.4 给出了特征映射中被限制的权值以及神经元的感受野。如图所示，多特征映射——每个要检测的特征——结构是相同的。

从图 6.5 我们也应注意，特征映射中的神经元感受野可以是叠代的或相邻的。特征映射输出被传递到网络较高阶段。通常，第一层含有叠代感受野，下一层含有相邻的感受野。其目的是限制特征平移后的影响。这种结构同十一章讨论的神经认知机网络的 S-层非常相似。在同一图像中，需要多个特征映射来描捉不同的特征。与神经认知机类似后续的隐含层，可以使用特征映射层来提取高阶特征。这样的高阶特征不需要像简单特征那样的精确位置信息。这样在特征映射层后可以使用类似神经认知机 C-层的那样的层(见十一章，图 11.1)。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and cooperation: luzhenbo@yahoo.com.cn

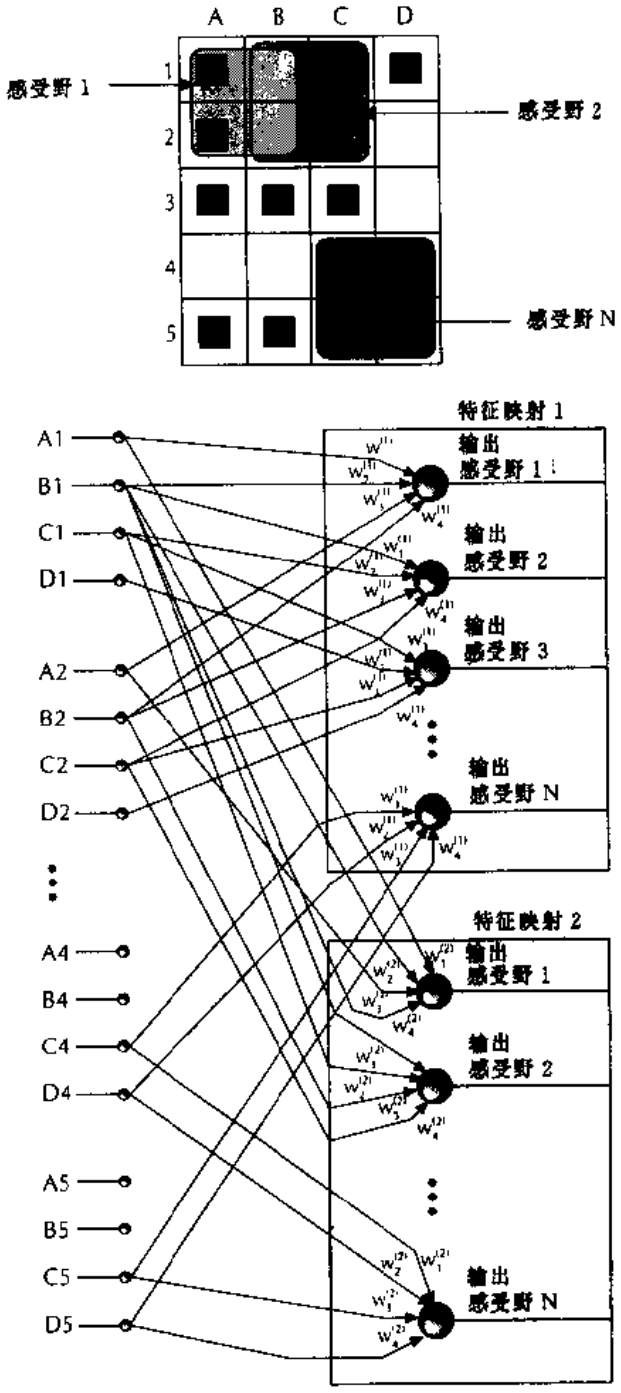


图 6.4 感受野和相应的特征映射

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

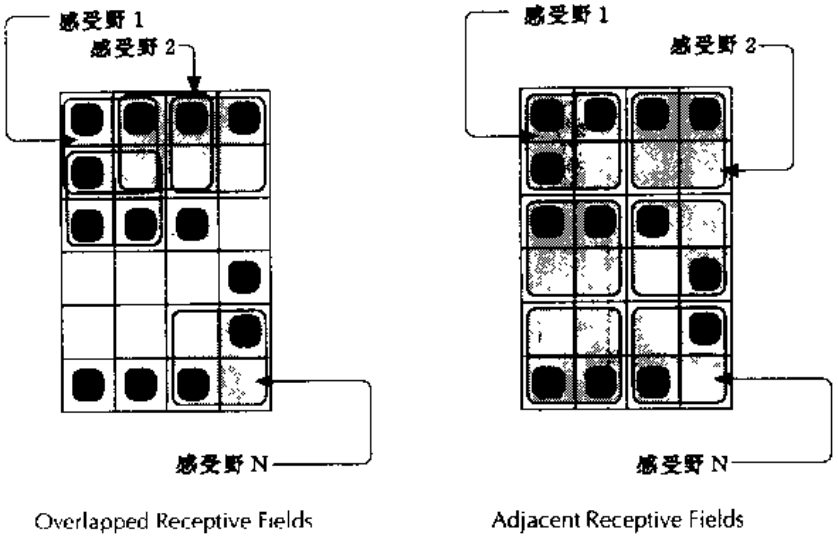


图 6.5 重叠的和相邻的感受野

6.4 基于几何特征的一个网络例子

我们已经详细讨论了环、交叉点和端点的提取，似乎也应该给出它们在神经网络识别器中的重要应用。

图 6.6 给出了一个可能的网络，它利用了到目前为止讨论的几个几何特征。目的是建造一个工具来研究几何特征的应用，而不是提出一个理想的解决方法。这个结构同 Darwiche 等[1992]提出的结构是相似的，所以有理由期望它具有类似的所需特性(见 6.2 节)。

图 6.6 画出的网络的特征点是由 n 个四元组的(x,y,I,E)量所组成。坐标 x 和 y 是象素坐标格中规一化的位置，也就是对于一个 M×M 象素格，在第 i 排和第 j 列一个象素点的坐标为 $x=j/M, y=i/M$，I 和 E 是二进制值并且分别表示特征点是交叉点或端点。如果二者都不是时，参考点应是沿着象素路径的中间点，大量的这样的中间点对于明确地定义字符是很必要的。包含有这样的中间点从某种意义上说可认为能提供字符的连结信息。很明显，很容易增加另外的特征项。例如，包含环特征(见 6.2.2 节)就变为 5-元组(x,y,I,E,L)。

并非所有字符都有同样的特征点数。网络分成 n 个四元组是依据不同的字符(实际上同一特征的不同例子)，能够并且有可变数目的特征点这一事实提出的。对子那些具有少于 n 个特征的字符，输入到多余的输入层神经元的值设为 0。特征点将以连续的、可重复的方式在网络中出现。一个简单并且较好的选择将是对这些点首先按 x，然后按 y 归类，由此决定表示的顺序。

这里需要特别提及两点，首先是用模拟值表示特征点的位置。这是希望有助于进行内插值的推广(至少能推广至原始的位图格式输入)，对于如旋转、平移及比例变化这样的变换，这些几何描述并不是内在的不变量；在某种扰动范围内，上面提到的推广方法，能使网络具有容错性，特别是在加上合适的训练矢量集合之后。

另外一点是尽管上面所示的网络(见图 6.6)称为 ACON 网络，这方面没有什么可特别提及的。此网络的重要性在于它的输入，对于这种结构我们可很容易地按需要建立一

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication or cooperation: luzhenbo@yahoo.com.cn

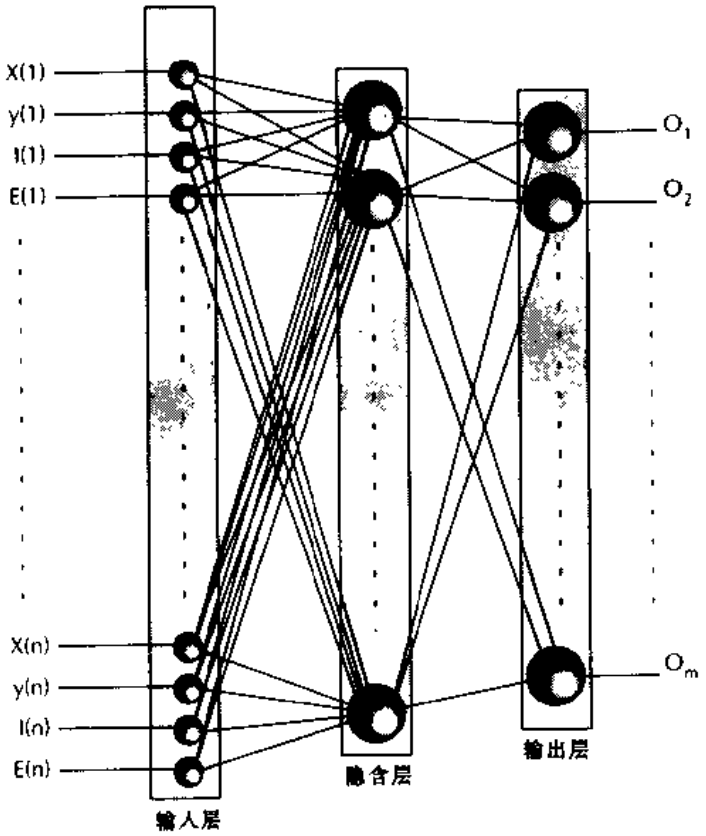


图 6.6 几何特征网络

OCON 网络。

6.5 利用变换进行特征提取

以上各节，我们讨论了不同的几何特征以及计算它们的方法。下面几节中，我们给出已用于各种模式识别应用的各种主要变换中的两个方法。为了便于讨论这些变换，我们也同时讨论能保存各种不变量形式的特征。特别将要讨论的变换是 Gabor 变换和傅立叶描述符(FD)。

6.6 傅立叶描述符(FD)

FD 提供了一种用于描述非重叠对象的边界的方法。这个方法具有一个额外的优点，能产生一种对于旋转、平移和比例变化都具有不变性的表示方法。对于 FD 特性的详细讨论，请看 Boas[1996]。根据 Venugopoal 等[1992]的讨论，我们讨论图 6.7 给出的闭合曲线 γ。

图中 $x(0)$，$y(0)$表示沿着曲线的任意起始点，$\theta(l)$是从始点沿着曲线在距离 l 处的正切方向，如果曲线总长为 L，那么弧长 l 为 $0 \leqslant l \leqslant L$。从函数 $\varphi(l)=\theta(l)-\theta(0)$可得到弧长 l 相对于始点角度。

对长度规一化得到 $\varphi^*(t)=\varphi(Lt/2\pi)+t$ 这里 $0 \leqslant t \leqslant 2\pi$。注意通过用规一化的形状函数来表示形状 φ^*，我们就得到了对于旋转、平移以及比例变化的所需不变性。

现在我们可以把 $\varphi^*(t)$展开为 Fourier 序列，得到：

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

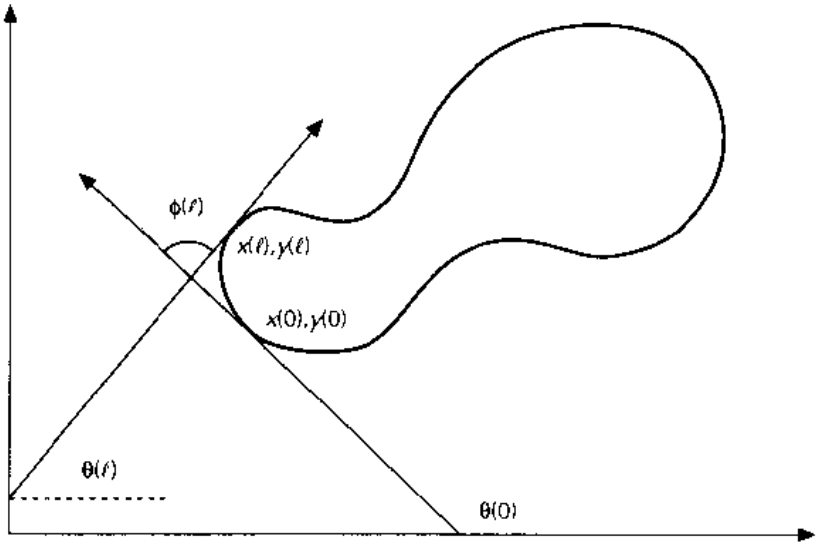


图 6.7 对于闭合曲线 γ 的形状函数的定义

$$\varphi^*(t) = \mu_0 + \sum_{k=1}^{\infty} a_k \cos k_t + b_k \sin k_t \tag{6-3}$$

在连续情况下：

$$\mu_0 = \frac{1}{2\pi}\int_0^{2\pi} \varphi^*(t)\,dt \tag{6-4}$$

$$a_n = \frac{1}{\pi}\int_0^{2\pi} \varphi^*(t)\cos(nt)\,dt \tag{6-5}$$

$$b_n = \frac{1}{\pi}\int_0^{2\pi} \varphi^*(t)\sin(nt)\,dt \tag{6-6}$$

转化到极化坐标为：

$$\varphi^*(t) = a_0 + \sum_{k=1}^{\infty} A_k \cos(kt - \lambda_k) \tag{6-7}$$

$$A_k = \sqrt{a_k^2 + b_k^2} \tag{6-8}$$

$$\lambda_k = \tan^{-1}\left(\frac{b_k}{a_k}\right) \tag{6-9}$$

集合$\{A_k, \lambda_k k = 1, \cdots, \infty)$是对曲线的傅立叶描述符。

最后举一个有 m 个顶点的多边曲线 $V_0 \cdots V_{m-1}$(见图 6.8)的特殊例子，傅立叶描述符可计算为：

$$a_k = -\frac{1}{k\pi}\sum_{n=1}^{m} \Delta\varphi_n \sin\frac{2\pi l_n}{L} \tag{6-10}$$

$$b_k = \frac{1}{k\pi}\sum_{n=1}^{m} \Delta\varphi_n \cos\frac{2\pi l_n}{L} \tag{6-11}$$

$$l_n = \sum_{i=1}^{n} \Delta l_i \tag{6-12}$$

这里 l_i 是边(V_{i-1}, V_i)的长度，并且 $\Delta\varphi_n$ 是顶点 V_n 的角方向。

此情况对字符识别有特殊意义，在这种逐段方式下，它很自然地代表特征映射。总之，我

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

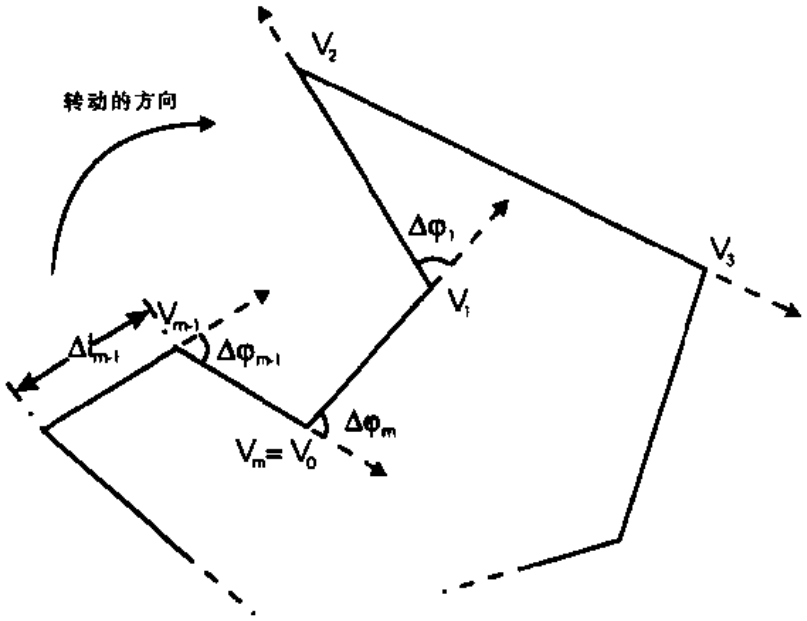


图 6.8 逐段多边形曲线的形状表示方法

们试图用能保存形状的最小线段集合来表示字符。

注意通过定义特征 φ^* 已引入了所需的不变性。那么计算 FD 就是为了达到降低输入矢量维数的目的。

在 Venugopal 等[1992]中,使用前 15 个 FD 作为特征,在一前馈网络中就产生了 30-分量的输入矢量。由 ALOPEX(见第四章讨论的)训练的这个网络,对一个手写数字问题的识别精度为 98%,使用的误差函数是:

$$e_k = \log\left(\frac{1}{out_k}\right), \text{当要求值为 1 时} \tag{6-13}$$

$$e_k = \log\left(\frac{1}{1-out_k}\right), \text{当要求值为 0 时} \tag{6-14}$$

$$E = \sum_k e_k \tag{6-15}$$

这里 e_k 是在第 k 个输出神经元的误差,E 是总误差。

6.7 Gabor 变换和子波

Gabor 变换是一种受到生物启发的变换,它被用来将原始的输入数据变换(为更为有用的图像格式),以便于对模式编码来进一步分类。生物视觉系统特别是视觉神经层的简单细胞只使用局部的频率信号,这与 Gabor 运算符[Porat 和 Zeevi,1988]的运算结果类似。也就是说,仿造此生物模型,我们寻找用一函数族来表示一幅图像的方法,这些函数既包含空间域信息同时也包含频域信息。

位图图像仅包含空间信息,Kosko[1992]指出位图格式也可看作是一种变换,这里基函数是 delta 函数。Fourier 变换用 sin 和 cos 项表示图像,它仅提供了频域信息。它给出了属于某一频率的图像空间周期性,但丢失了所有的局部度量。先不考虑生物视觉特性,包括两种信息的特征很直观,它比那些只包含一种信息的更优。下面就只是去看怎样把它完成得更好。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@163.com

首先,我们寻找的基函数族是空间项和频率项的乘积,表示如下:

$$f_k = g(x)\exp(ikx) \tag{6-16}$$

一函数(即时间宽度和频域的乘积)所能达到的精度受到测不准法则的限制。已经证明,当空间项 $g(x)$ 是一Gaussian 窗口时,能达到在测不准法则下的极小值,由此定义了一维的 Gabor 函数:

$$\gamma_k = a_{k,\sigma}\exp\left(-\frac{x^2}{2\sigma^2}\right)\exp(ikx) \tag{6-17}$$

$a_{k\sigma}$ 是一规一化初始常量;下面是基于 Gabor 函数的 Morlet 子波定义:

$$\Psi_k = b_{k,\sigma}\exp\left(-\frac{-k^2x^2}{2\sigma^2}\right)\exp(ikx) \tag{6-18}$$

图 6.9 给出一个 Morlet 子波例子。

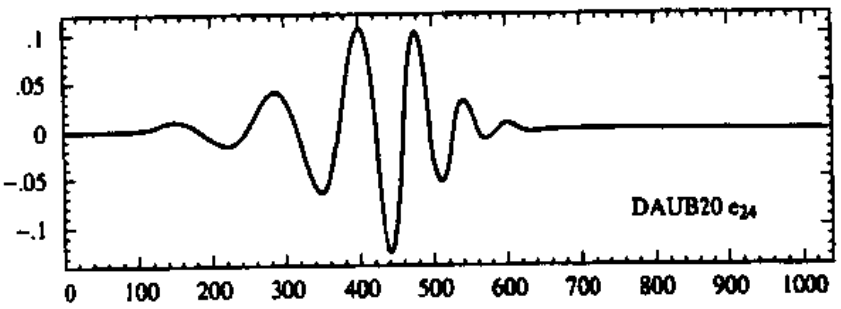


图 6.9 一个 Morlet 子波例子(摘自 Press, W. H. 等, Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press, 1992. 经过授权)

二维条件下,Morlet 滤波器变成:

$$\Psi_k = n_{k,\sigma}\exp\left(-\frac{-\mathbf{k}^2\mathbf{x}^2}{2\sigma^2}\right)\exp(i\mathbf{kx}) \tag{6-19}$$

它可以重新构造为:

$$\Psi(n,y,\sigma,k_x,k_y) = \exp\left(-\frac{-k_x^2k_y^2}{2\sigma^2}(x^2+y^2)\right)\exp|i(k_xx+k_yy)| \tag{6-20}$$

这里 (x,y) 是空间域的位置变量,(k_x,k_y) 是波数,σ 是高斯窗参口数(也就是 Gaussian 包络的标准偏差)。从这里很明显看出子波有一方向角 θ,频率 ω_0 如下:

$$\theta = \tan^{-1}\left(\frac{k_y}{k_x}\right) \tag{6-21}$$

$$\omega_0 = \sqrt{k_x^2+k_y^2} \tag{6-22}$$

可以将 Morlet 变换表达成图像 $I(\mathbf{x})$ 和 Morlet 基函数族的图像卷积:

$$G(\mathbf{k},\mathbf{x}_0) = \int \Psi(x_0-x)I(x)d^2x \tag{6-23}$$

$\mathbf{k}$ 和 $\mathbf{x}_0$ 的范围遍及图像和频率平面。

实际上,卷积表示一窗口中每一面像元素的乘积的和,这个窗口的尺寸是相应滤波值的滤波器的大小,x_0 确定了图像中的滤波窗口的位置。当然 Gabor 变换是类似的,只是简单地用 γ 代替 Ψ。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication: luzhenbo@yahoo.com.cn

为了完全确定基于滤波器集合的子波，必须选定方向角 θ、频率 ω 和 Gaussian 偏差 σ。通常只使用 σ 的某一值时，对于 θ 和 ω_0，必须选定一系列的值，目的是为了很好地对图像描述。因为 σ 决定了当我们从窗口中心偏移时，空间域分量值降低的尖锐程度，所以窗口的大小对 σ 的选择影响很大。

在 Lu 等[1991]的研究中使用了这样一组参数，对于每个滤波窗口中的点总共 24 个特征：σ＝π 坐标 4 等分 ω ＝π/8，π/4，π/2，π 和 6 个方向 θ＝0，30，60，90，120，150 度。在此研究中，基于它们的子波描述特征被聚类了。在此研究中，对于以每一象素为中心的窗口，得到了全部的 24 元素特征集；很显然，这变得相当复杂。对于 64×64 窗口将有 4096 矢量，每个矢量都是 24 维的。因此，必须重点选择窗口的大小和窗口的重叠程度。从生物模型得知，80% 的窗口都是重叠的，或许，这代表一个有优点的图从哪开始。在任何情况下，它当然比在每个象素点确定窗的中心效率要高的多。此研究[Lu 等 1991]中的聚类，使用自组织特征映射来完成。

Gabor 子波变换已成功地应用在怎样确定信封中块位置的问题[Jain 和 Bhattacharjee 1992]。这时，使用下面的偶对称 Gabor 滤波器：

$$G(x,y)=\exp\left\{-\frac{1}{2}\left[\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}\right]\right\}\cos(2\pi u_0 x+\phi) \tag{6-24}$$

注意，Gaussian 窗口参数现在取决于方向，且指数项已由 cos 项代替，使用了 4 个方向角度，同时径向频率的数目是 $\log_2$(图像中的列数)。

到目前为止，Gabor 变换已频繁地应用于图像的纹理分割，它们也被应用于字符识别问题上。Garris 等[1991]运用 Gabor 特征，作为由反向传播算法训练的网络的输入，训练和测试仅包含单种 OCR 字体，识别精度达到 100%。Wilson 等[1990]将 Gabor 滤波器用于自适应共振理论(ART)网络，对于印刷体的识别率是 99%，而对于手写体则是 80%。总之，将子波变换应用于图像识别及字符识别问题中，结果是特别吸引人的。但是它的主要缺点就是计算太复杂。

已证明，Gabor 变换能几乎与以先前讨论的 FD 表示方法相同的方式，来表示字符的形状。实际上，这正是 Eichman 等[1990]已经做过的。

回想先前讨论的傅立叶描述符所使用的形状函数：

$$\varphi(l)=\theta(l)-\theta(0) \tag{6-25}$$

$$\varphi(t)=\varphi\left(\frac{Lt}{2\pi}\right)+t \tag{6-26}$$

标记中微小的改变，就是我们现在用 φ' 代替最初 φ^*。这是为了避免混淆，因为在后面“*”表示复数的共轭。在 FD 的讨论中，函数 φ 展开为 Fourier 序列，其中的基函数由 sin 和 cos 组成。这种表示的主要缺点是 FD 并没有反映局部谱信息，形状函数的局部畸变将影响 FD 的全部集合。通过使用 Gabor 基本函数族来改正此缺陷。

$$f_{mn}=w(t-mD)e^{inWt} \tag{6-27}$$

现在函数 $\varphi'(t)$ 可用这种基函数族来表示：

$$\varphi'(t)=\sum_{n=-\infty}^{n=\infty}\sum_{m=-\infty}^{m=\infty}a_{mn}f_{mn}(t) \tag{6-28}$$

同先前一样，我们将 w(t) 选定为 Gaussian 函数：

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@hotmail.com

$$w(t)=g(t)=\left(\frac{\sqrt{2}}{D}\right)^{1/2}e^{\left[-\pi\left(\frac{t}{D}\right)^2\right]} \tag{6-29}$$

系数 a_{mn} 可由下式得到：

$$a_{mn}=\int\varphi'(t)\gamma^*(t-mD)e^{inWt}dt \tag{6-30}$$

有必要使用附加的双正交函数，γ(t)，因为 Gabor 基函数本身并不是正交的，γ(t)由下式给出：

$$\gamma(t)=\left(\frac{1}{\sqrt{2}D}\right)^{1/2}\left(\frac{K_0}{\pi}\right)^{-3/2}e^{\left[\pi\left(\frac{t}{D}\right)^2\right]}\sum_{n+1/2\geqslant t/D}(-1)^n e^{[-\pi(n+1/2)^2]}$$
$$t=0,1,\cdots,D-1 \tag{6-31}$$

$k_0=1.8540746$ 是规一化因子。

同早期 Fourier 描述符相似，复数的 Gabor 系数 a_{mn}，现表示图像的边界形状，可作为识别器的特征集合。

应指出，尽管 Gaussian 函数在上面讨论的某种意义上说是最优的，但其它的窗口函数也是可用的。现给出文献中这种函数的一种，如下：

$$w(t)=\mathrm{rect}(t/D)=\begin{cases}1, & -D/2<t<D/2\\ 0, & \text{其它情况}\end{cases} \tag{6-32}$$

图 6.10 原始字母“B”图像(摘自 Eichman, G. 等，SPIE, vol.1297, pp.86～94, 1990. 经过授权)

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

相应的双正交函数:

$$\gamma(t)=\frac{1}{D}\mathrm{rect}\left(\frac{t}{D}\right) \tag{6-33}$$

摘自 Eichman 等[1990]的图 6.10~图 6.13,给出了一个有用的处理阶段图形表述。在此处理阶段中,首先是将图像用 Gabor 系数进行表达,然后经两次变换恢复。此例中,使用的参数为 M=16,N=D=32。

原始字符,字母"B",由图 6.10 中给出。

与形状函数 $\varphi'(t)$相关的谱,在图 6.11 中给出。使用 Gabor 系数描述的字符,由图 6.12 给出。最后,图 6.13 给出了由 Gabor 系数重构的字符。

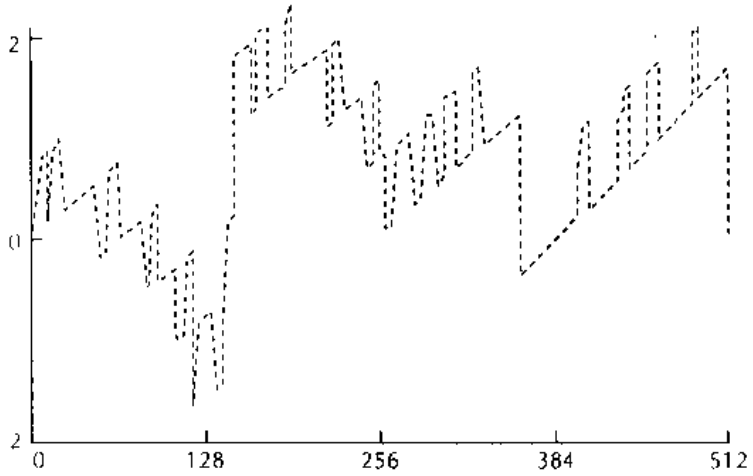


图 6.11 字母"B"的形状函数 $\varphi'(t)$的谱(摘自 Eichman, G. 等,SPIE,vol.1297, pp.86-94,1990. 经过授权)

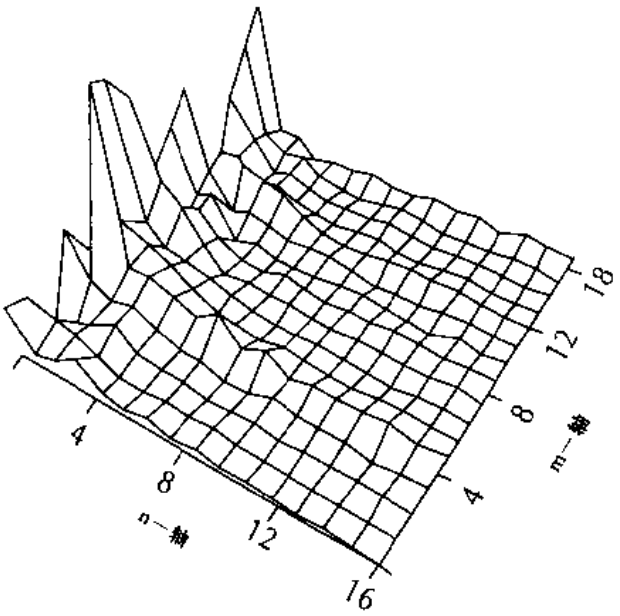


图 6.12 使用 Gabor 系数描述的字符(摘自 Eichman, G. 等,SPIE,vol.1297, pp.86-94,1990. 经过授权)

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xaut.edu.cn

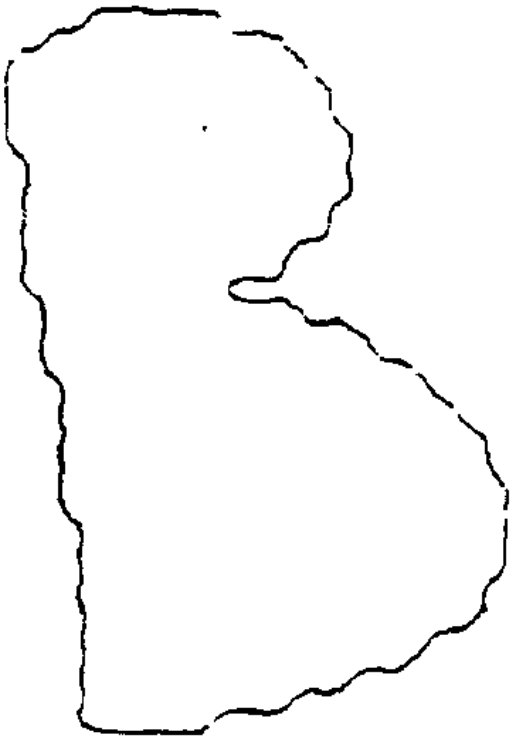


图 6.13 由 Gabor 系数重构的字母"B"(摘自 Eichman, G. 等,SPIE,vol.1297, pp.86-94,1990. 经过授权)

参考书与文献

Boas, J. L., *Mathematical Methods in the Physical Sciences,* John Wiley & Sons, New York, 1966.

Darwiche, E., Pandya, A. S., and Mandalia, A. D., "Automated optical recognition of degraded handwritten characters," *SPIE,* vol. 1661, pp. 203–214, 1992.

Eichman, G., Lu, C., Jankowski, M., and Tolimieri, R., "Shape representation by Gabor expansion," Hybrid Image and Signal Processing II, *SPIE,* vol. 1297, pp. 86–94, 1990.

Garris, M. D., Wilkinson, R. A., and Wilson, C. L., "Analysis of a biologically motivated neural network for character recognition," *ACM ANNA,* pp. 160–175, 1991.

Jain, A. K. and Bhattacharjee, S. K., "Address block location on envelopes using Gabor filters," *Pattern Recognition,* vol. 25, no. 12, pp. 1459–1477, 1992.

Kosko, B., *Neural Networks for Signal Processing,* Prentice-Hall, Englewood Cliffs, NJ, 1992.

Le Cun, Y. L., Boser, B., Denker, J. S., Henderson, D., Hubbard, W., and Jackel, L. D., "Handwritten Digit Recognition with a Backpropagation Network," *Advances in Neural Information Systems,* Vol. 2, Morgan Kaufman, San Mateo, CA, pp. 396–404, 1990.

Lu, S., Hernandez, J. E., and Clark, G. E., "Texture segmentation by clustering Gabor feature vectors," *IEEE Joint Conf. Neural Networks,* vol. 1, pp. 683–687, 1991.

Pikaz, A. and Dinstein, I., "Using simple decomposition for smoothing and feature point detection of noisy digital curves," *IEEE PAMI,* vol. 16, no. 8, pp. 808–813, 1994.

Porat, M. and Zeevi, Y. Y., "The Generalized Gabor scheme of image representation in biological and machine vision," *IEEE PAMI,* vol. 10, no. 4, pp. 452–468, 1988.

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & cooperation: luzhenbo@yahoo.com.cn

Press, W. H., Tenkoiseley, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, England, 1992.

Venugopal, K. P., Pandya, A. S., and Sudhakar, R., "Invariant recognition of 2-D objects using Alopex," *Proc. SPIE*, vol. 1709, pp. 182–190, 1992.

Wang, J. and Jean, J., "Resolving multifont character confusion with neural networks," *Pattern Recognition*, vol. 5, no. 1, pp. 175–187, 1993.

Wilson, C. L., Wilkinson, R. A., and Garris, M. D., "Self organizing neural network character recognition on a massively parallel computer," *IEEE IJCNN*, vol. 2, pp. 325–329, 1990.

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xidian.edu.cn

第七章 特征提取Ⅱ:主分量分析

7.1 降 维

对高维数据集合的分析,常常是一项很复杂的工作。模式识别系统经常受到高维问题的困扰。特征提取过程是在对象分类的过程之前,它利用变量变换使某些变量比其他变量更重要,因此可以将其看成是一种降维。我们把那些更重要的变量称为特征。如果次要的变量可忽略不计,那么就达到了所需的降维的目的。此外,从统计意义上讲,次要分量有可能来自于噪声(或者是来自于模式固有特性无关的外界干扰)。因此,除去这些分量可能会有益处。

特征提取过程是将数据空间(模式空间)变换为特征空间。虽然同最初的数据空间相比,特征空间维数降低得很多,但它仍然保留了数据内容的大多数本质信息。降低维数的方法很多,比如主分量分析、因素分析以及特征聚类等。因素分析通过特征值的线性组合来达到减少数据量的目的,从而用较低维数形式就可以说明特征之间的相互关系。特征聚类则是通过将一些密切相关的特征进行合并,来减少冗余信息[Duda 和 Hart ,1973]。

主分量分析(PCA)提供了进行此类变换的一种方法。在这种变换中,特征空间应尽可能地代表全体变量。在变量的重要性是由统计估计的多变量分析中,主分量分析是众所周知的技术。此外,通过引入一种可检测因忽略变换向量的某些分量所造成误差的量度,我们可知这些摒弃分量的数目以及特性,并能以最小丢失率来降低随机输入矢量的维数。

假设将一个 n 维模式矢量 $\mathbf{X}$ 映射成 m 维空间中的特征矢量 $\mathbf{Y}$,其中 $m<n$。E 代表均方差,它等于为获得 $\mathbf{Y}$ 而从 $\mathbf{X}$ 中删除元素的方差总和。主分量分析找到了线性可逆变换 T,因此从 $\mathbf{X}$ 中删除元素后得到

$$\mathbf{Y}=T\cdot\mathbf{X} \tag{7-1}$$

它在均方差上是最优的。

这种建立在矢量表示的统计特性基础上的变换,通常指的是特征向量法、主分量分析、Hotelling 变换或是 Karhunen-Loeve 变换。1901 年,Pearson 首先将变换引入生物学领域,重新对线性回归进行了分析,得出了一种新的形式。Hotellin[1993]则将其与心理测验学领域联系起来,把离数变量转变为无关联系数。几年后[1956],Kramer 和 Mathews 也发现了 Hotelling 变换。

1947 年,在概率论理论建立的同时,Hotelling 变换又单独出现,用来进行连续数据变换。随后,Loeve[1948, 1963]将其归纳总结。因此,这一变换被称为 Karhunan-Loeve 变换。Koschman[1954]证明了 K-L 变换均方差最小。为了更好地理解主分量分析,读者可查阅多变量分析教程,如 Preisendorfer[1988] 或 Jolliffe[1986]所著。

针对那些具有信号处理基础的读者,Kung[1933]给出了关于主分量神经网络的精辟见解。他将主分量分析问题应用于 Wiener 滤波器。后者是当时以最小方差恢复原始信号的流行技术。在这里主特征分量代表的是信号能量最大方向。

图 7.1 画出了在笛卡尔坐标系中一个二维随机矢量及其主分量。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & cooperation: luzhenbo@yahoo.com.cn

注意图 7.1 中 X_1 轴和 X_2 轴都不能有效地区分各分布点，而第一主分量轴却做得很好。我们也可观察到在次分量轴上点分布是较少的。

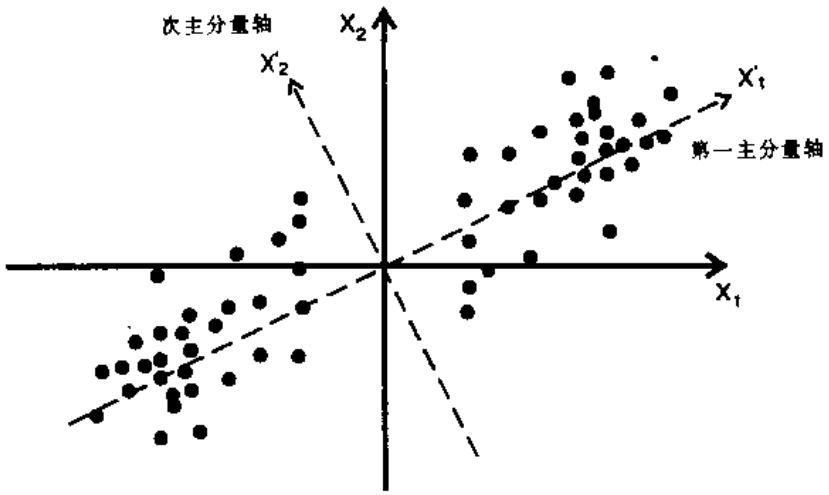


图 7.1 带有主分量轴的模式空间

7.2 主 分 量

这一节我们将首先对主分量进行讨论，而有关求得主分量的 Karhunen-Loeve 变换将在下节阐述。最后，我们将举一个例子，来讲述将主分量作为两个不同类中的样本特征进行提取的过程。

考虑一个 n 维模式矢量组 $\mathbf{X}$，表示为：

$$\mathbf{X}=\begin{bmatrix} x_1 \\ x_1 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} \tag{7-2}$$

一般来说，当给出概率分布函数$\{\mathbf{X}\}$，矢量组均值可表示为：

$$\mu_x = E[\mathbf{X}] \tag{7-3}$$

$E[\mathbf{X}]$用来表示模式矢量 $\mathbf{X}$ 的期望值。

当然，在没有概率密度函数的条件下，矢量均值可以通过离散抽样近似得出：

$$\mu_x = \frac{1}{M}\sum_{k=1}^{M} x_k \tag{7-4}$$

模式矢量组的协方差一般可定义为：

$$\sum\nolimits_x = E[(x_k-\mu_x)(x_k-\mu_x)^T] \tag{7-5}$$

这是一个 $n\times n$ 阶的实对称矩阵。$\sum_x$ 的元素 σ_{ij} 是矢量 $\mathbf{X}$ 的分量 x_i, x_j 的协方差。如果 x_i, x_j 无关，则协方差为零，即：

$$\sigma_{ij}=\sigma_{ji}=0 \tag{7-6}$$

因此，协方差矩阵为：

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@tom.com

$$\Sigma_x = \frac{1}{M-1}\sum_{k=1}^{M}(x_k - \mu_x)(x_k - \mu_x)^T \tag{7-7}$$

一般来说，如果概率分布函数$\{r\}$是先验的，则可给出自相关矩阵为：

$$R_x = E[\mathbf{X}\mathbf{X}^T] \tag{7-8}$$

对于自相关矩阵 R_x $\{\Phi_i\}$代表归一化特征矢量，例如：

$$\| \Phi_i \| = \sqrt{\Phi_i^T \Phi_i} = 1 \tag{7-9}$$

相应的特征值为$\{\lambda_i\}$。

模式矢量 $\mathbf{X}$ 可以以特征矢量 Φ_x 为间隔投影到一维子空间上，从而得到特征分量。如图 7.1 所示，模式矢量 $\mathbf{X}$ 的第一主分量(例如，最大分量)可表示为：

$$\mathbf{X}_i = \Phi_i^T \cdot \mathbf{X} \tag{7-10}$$

其余的 m 个主分量也可类似地通过投影获得。注意最后一个分量有最小值。

实际上，很少能得到模式矢量组自相关矩阵的先验知识。不过，对于大量的输入矢量样本$\{\mathbf{X}(t), t=1,\cdots,M$，自相关矩阵 R_x 的近似值可由样本矢量均值表示：

$$\tilde{R}_x = \frac{1}{M}\sum_{t=1}^{M}\mathbf{X}(t)\cdot\mathbf{X}^T(t) \tag{7-11}$$

同时也可假定：

$$R_x = \lim_{M\to\infty}\tilde{R}_x \tag{7-12}$$

因此，当 M 足够大时，在某种意义上这种近似能够令人满意。

7.2.1 PCA 示例

考虑四个模式矢量：

$$\mathbf{x}_1 = \begin{pmatrix}1\\0\\1\end{pmatrix}, \mathbf{x}_2 = \begin{pmatrix}2\\3\\1\end{pmatrix}, \mathbf{x}_3 = \begin{pmatrix}0\\1\\1\end{pmatrix}, \mathbf{x}_4 = \begin{pmatrix}1\\4\\1\end{pmatrix} \tag{7-13}$$

由式 7-4 可求得均值矢量：

$$\mu_x = \frac{1}{4}\begin{pmatrix}1+2+0+1\\0+3+1+4\\1+1+1+1\end{pmatrix} = \begin{pmatrix}1\\2\\1\end{pmatrix} \tag{7-14}$$

再由式 7-7 可求出协方差矩阵：

$$\begin{aligned}\sum_x &= \frac{1}{4}\left\{\begin{pmatrix}1-1\\0-2\\1-1\end{pmatrix}(0\ -2\ 0) + \begin{pmatrix}1\\1\\0\end{pmatrix}(1\ 1\ 0) + \begin{pmatrix}-1\\-1\\0\end{pmatrix}(-1\ -1\ 0) + \begin{pmatrix}0\\2\\0\end{pmatrix}(0\ 2\ 0)\right\}\\ &= \frac{1}{4}\begin{pmatrix}0+1+1+0 & 0+1+1+0 & 0+0+0+0\\ 0+1+1+0 & 4+1+1+4 & 0+0+0+0\\ 0+0+0+0 & 0+0+0+0 & 0+0+0+0\end{pmatrix}\\ &= \begin{pmatrix}0.5 & 0.5 & 0\\ 0.5 & 0.5 & 0\\ 0 & 0 & 0\end{pmatrix}\end{aligned} \tag{7-15}$$

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and cooperation: luzhenbo@yahoo.com.cn

7.3 KARHUNEN-LOEVE(K-L)变换

相对于方差来说，主分量是具有特殊性质的随机变量的线性组合。第一主分量则是最大方差的正规化线性组合，第二主分量是次最大方差的组合，依次类推。实际上，主分量是根据区分类的能力进行划分的。K-L 变换就是具有这一性质的标准正交变换，它将 n 维矢量 $\mathbf{X}$ 变换成 n 维矢量 $\mathbf{Y}$。现已证明主分量即协方差矩阵的特征矢量。

由主矢量的线性组合可以得出一个特征矢量，而相应的特征值反映了某一特定矢量的重要性。那么多少个主分量才足以表达全体变量呢？在某些情况下，可使用阈值进行判决，舍去小于阈值的主分量。有时分量数是预先确定的，对于特征空间有具体要求的情况，则根据二维或三维空间的要求加限制条件。

但要找到主分量，仍需求出协方差矩阵 $\sum_x$ 的特征矢量 Φ_i 和特征值 λ_i：

$$\sum\nolimits_x \Phi_i = \lambda_i \Phi_i \tag{7-16}$$

然后，n 维随机输入矢量 $\mathbf{X}$ 可用主分量表示为：

$$\mathbf{Y} = \Phi^T \mathbf{X} \tag{7-17}$$

分量形式为：

$$y_i = \Phi_i^T x_i$$

式(7-17)也称作 Hotelling 变换。

模式识别中，每个分量 y_i 可以被看作模式矢量 $\mathbf{X}$ 的一个特征。这些特征互不相关。因此，协方差矩阵是对角矩阵。即：

$$\sum\nolimits_y = \Phi^T \mathbf{X} \Phi = \begin{bmatrix} \lambda_1 & 0 & 0 \cdots & 0 \\ 0 & \lambda_2 & 0 \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 \cdots & \lambda_n \end{bmatrix} \tag{7-18}$$

进而，由最初的 $m < n$ 个主分量表示($\lambda_1 > \lambda_2 \cdots \lambda_n > 0$)的 $\mathbf{X}$ 的均方差为：

$$\varepsilon_{(m)}^2 \sum_{i=m+1}^{n} \lambda_i \tag{7-19}$$

舍弃一些分量后的模式矢量 $\mathbf{X}$ 的近似表达式为：

$$\mathbf{X}' = \sum_{i=1}^{m} y_i \Phi_i \tag{7-20}$$

误差矢量 e 表示为：

$$e = \mathbf{X} - \mathbf{X}' \tag{7-21}$$

均方差表示为：

$$\| e \| = \| X - X' \| = \left(\sum_i (x_i - x_i')^2 \right)^{\frac{1}{2}} \tag{7-22}$$

图 7.2 给出了误差矢量 e 和原来的模式 $\mathbf{X}$ 的相应关系。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xidian.edu.cn

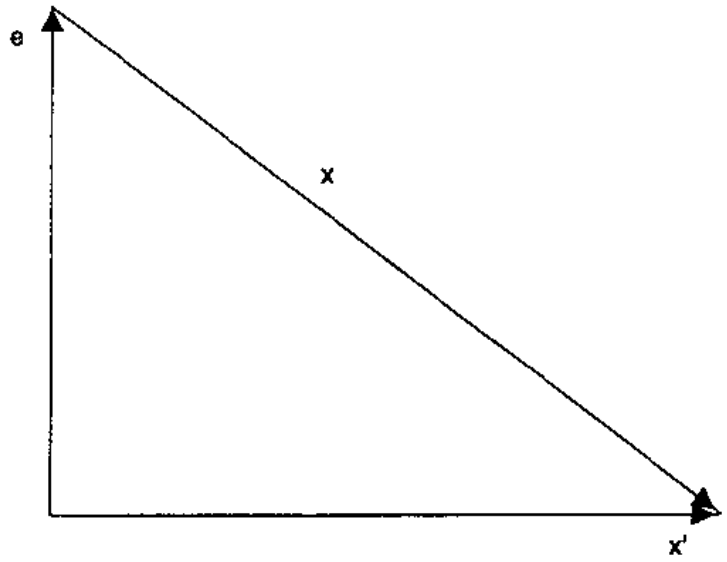


图 7.2 初始模式矢量与近似模式矢量关系

7.3.1 变换示例

考虑各具有四个模式的两类情况为：

$$c_1\left\{\begin{pmatrix}-2\\-2\end{pmatrix},\begin{pmatrix}-4\\-4\end{pmatrix},\begin{pmatrix}4\\4\end{pmatrix},\begin{pmatrix}2\\2\end{pmatrix}\right\}$$
$$c_2\left\{\begin{pmatrix}-2\\2\end{pmatrix},\begin{pmatrix}-4\\-4\end{pmatrix},\begin{pmatrix}4\\4\end{pmatrix},\begin{pmatrix}2\\-2\end{pmatrix}\right\} \tag{7-23}$$

各模式可以标绘在一个二维模式空间中，如图 7.3 所示。

由式(7-4)求出均值矢量：

$$\mu_1=0 \quad 和 \quad \mu_2=0 \tag{7-24}$$

由式(7-7)求出每类的协方差矩阵：

$$\sum\nolimits_a=\frac{1}{4}\sum_{i=1}^{4}x_i x_i^T \tag{7-25}$$
$$=\frac{1}{4}\left\{\begin{pmatrix}-4\\-4\end{pmatrix}^{(-4,-4)}+\begin{pmatrix}-2\\-2\end{pmatrix}^{(-2,-2)}+\begin{pmatrix}4\\4\end{pmatrix}^{(4,4)}+\begin{pmatrix}2\\2\end{pmatrix}^{(2,2)}\right\}$$
$$=\frac{1}{4}\begin{pmatrix}40&40\\40&40\end{pmatrix}=\begin{pmatrix}10&10\\10&10\end{pmatrix} \quad 对于\ C_1\ 类$$

$$\sum\nolimits_b=\frac{1}{4}\sum_{i=1}^{4}x_i x_i^T \tag{7-26}$$
$$=\frac{1}{4}\left\{\begin{pmatrix}-4\\-4\end{pmatrix}^{(-4,-4)}+\begin{pmatrix}-2\\2\end{pmatrix}^{(-2,2)}+\begin{pmatrix}4\\4\end{pmatrix}^{(4,4)}+\begin{pmatrix}2\\-2\end{pmatrix}^{(2,-2)}\right\}$$
$$=\frac{1}{4}\begin{pmatrix}40&24\\24&40\end{pmatrix}=\begin{pmatrix}10&6\\6&10\end{pmatrix} \quad 对于\ C_2\ 类$$

用 K-L 变换，我们可以估计 C_1 类的特征值 λ_i 和相应的特征矢量。

对于 C_1 类：

$$\lambda_{1a}=20,\lambda_{2a}=0$$

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

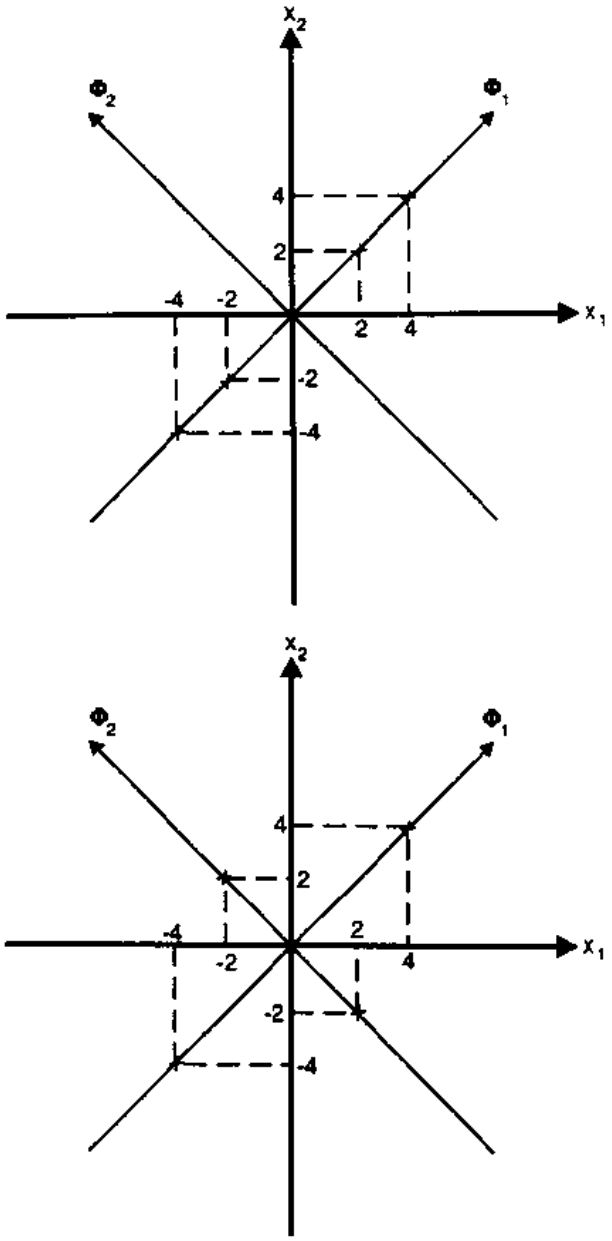


图 7.3 沿特征矢量标绘的模式

$$\varphi_{1a}=\begin{bmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{bmatrix},\varphi_{2a}=\begin{bmatrix}\frac{-1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{bmatrix} \tag{7-27}$$

对于 C_2 类：

$$\lambda_{1b}=16,\lambda_{2b}=0$$

$$\varphi_{1b}=\begin{bmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{bmatrix},\varphi_{2b}=\begin{bmatrix}\frac{-1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{bmatrix} \tag{7-28}$$

把这些特征向量以 45°方向标绘在模式空间上(见图 7.3)。

为了降低维数，我们考虑 C_1 类取消次主分量造成的影响。由于 $\lambda_{2a}=0$，由式 7-22 可知，

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xidian.edu.cn

均方差也为零。由图 7.3 我们清楚地看到,对于第一类四个模式可只用 Φ_1 表示。对于第二类,由于 $\lambda_{2b}=4$,舍去 Φ_2,误差值为 4。如图 7.3 所示,点 $(-4,4)^T$ 和 $(4,4)^T$ 可只由 Φ_1 表示,但对于点 $(-2,2)^T$ 和 $(2,-2)^T$ 将会造成误差。

此种情况下均方差为:

$$\| e \| = \frac{(0^2+0^2+(16^2)^{\frac{1}{2}})}{4} = 4 \tag{7-29}$$

图 7.4 说明了怎样将主分量应用于标绘在二维模式空间中的模式矢量(数据点)。在图表上,用水平轴和垂直轴分别表示自然坐标 X, Y。一旦求得了主分量(如 7.3.1 节的例子),就能画出旋转轴 1 和 2。把模式云图分别投影到 1,2 两个轴就可得到密度标绘图,如图 7.4 所示。在这个简单例子中,重要的是将数据集合投影到轴 1 上能得到数据的显著特征,即具有双峰性(即,此结构是由两类组成);而如果投影到原来的 x 和 y 轴则根本看不出这些。

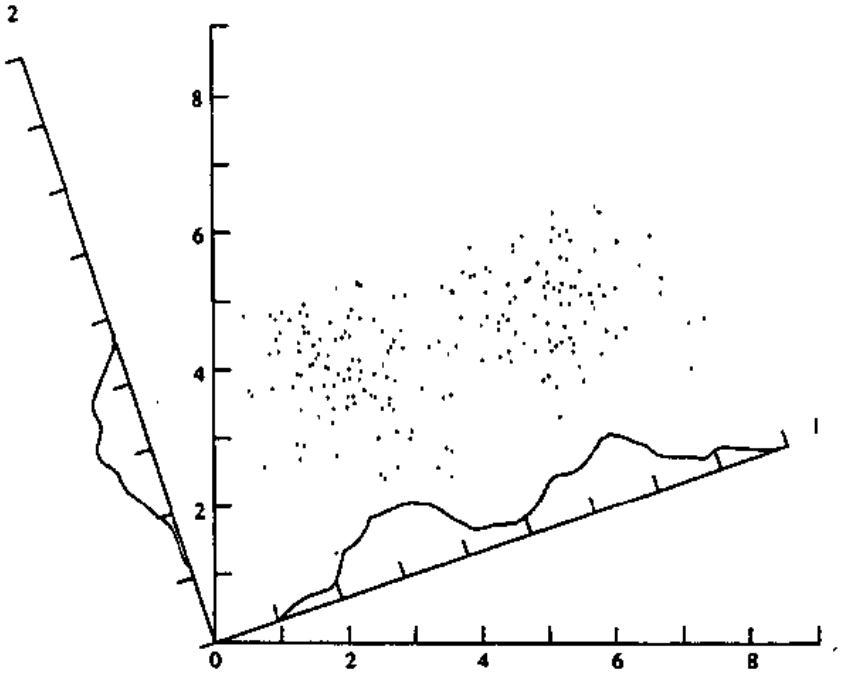


图 7.4 主分量分析揭示出数据点的双峰特性(选自 Linsker, R., Computer, vol.21, pp.105-117, 1998. IEEE 授权)

7.4 主分量神经网络

主分量法和 K-L 变换,早已被成功地应用在特征提取、图像/数据压缩和模式分类领域。特征矢量及特征值的计算需要数值方法(比如,见 Press 等,[1992]),计算量相当大就成了这种方法的缺点。在这一节中,我们将简单地讨论能完成这项艰巨工作的神经网络的应用。

在这些方法中,神经网络用来从高维模式矢量空间中,提取最有代表性的低维子空间。图 7.5 给出了一个多层感知网络,它可以用来提取数据中最关键和最有代表性的特性。输入层和输出层的单元数,由模式矢量 **X** 和 **Y** 各自的维数决定。这种方法的核心是用一些相对较少的隐含神经元来完成数据压缩。

矢量 **Y** 作为目标样本,产生所需的权值误差校正信号,所以这种结构是有监督训练。如

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and collaboration: luzhenbo@yahoo.com.cn

反向传播或 ALOPEX 等学习算法均可用来训练权值。

图 7.5 所示的网络，也可用于自联想网络。此时，网络通过训练输出 y，使得 y 尽量模拟 x。从这种意义上说，网络是无监督训练。在训练过程中，因为所需输出应与输入相同，所以在比较阶段，引入适当的反馈来完成 y 模拟 x 的过程。可以证明，如果网络只包含线性结点，那么经过适当的训练，网络权值将收敛到上面所讨论的主分量上[Kung，1993]。

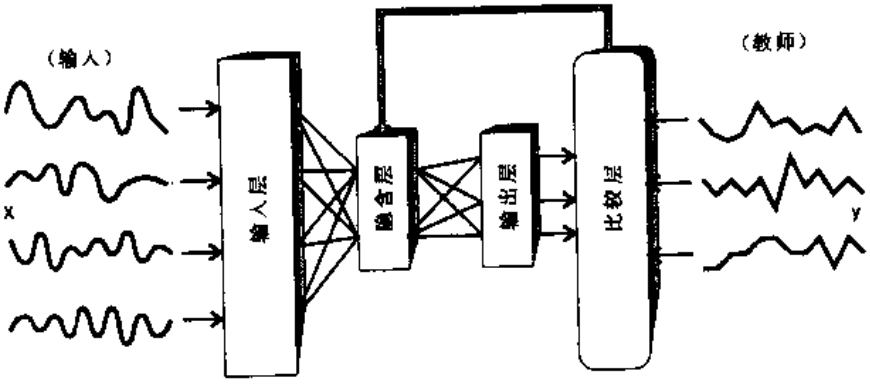


图 7.5 提取相关特征的一自联想神经网络(摘自 Kung[1993])

抛开主分量分析的严格概念不管，但这种自联想网络仍然是令人非常感兴趣的。设想使用这一网络进行特征提取，网络中使用非线性压缩函数(如 S 形函数)和远小于输入/输出层的隐含层。在这些条件下，网络能找到有效方法对输入数据集合中的信息进行编码。我们自然希望，非线性神经元能具有更强大的识别能力(我们可将讨论中的这些变化，看作径向基函数神经元)。剩下的就是前端自监督特征提取阶段了。这一阶段的特征由隐含神经元的输出端组成。也就是说，经过训练，去掉输出层，隐含层直接输出到识别网络。

Oja[1982]阐述了如何将单一线性神经元模型用于最大特征滤波器的设想。运用 Hebbian 类型的学习规则调整权值，使单一线性神经元可以作为模式分布中第一主分量的滤波器。分析在修正 Hebbian 规则下工作的一自组织线性神经元，Haykin[1994]证明了，用改进的 Hebbian 学习规则，调整自组织线性神经元的权值矢量，此权值矢量以 1 的概率收敛到欧几里德单位长度的矢量。此外，这个矢量与模式矢量相关矩阵的最大特征矢量方向一致。

Kung 和 Diamantaras[1990]提出了一种自适应主分量提取(APEX)的神经网络学习算法，可以用来对各个主分量作反复计算。给出最初 j 个主分量算法，以迭代方式工作，来计算出第(j+1)个主分量。有关此算法详细的讨论，读者可参阅 Kung 和 Haykin[1994]。

7.5 应　　用

Oja 的改进 Hebbian 规则，可以用来训练单一线性网络，用作最大特征滤波器。这一规则可推广到训练由线性神经元组成的单层前馈网络。Haykin[1994]详细的讨论了这一学习算法，它被称作广义 Hebbian 算法。Sanger[1989]论述了怎样训练网络，使之对任意大小的模式矢量进行主分量分析。

本节中将 GHA 应用于图像压缩问题上。如图 7.6 用三个小孩的图像来训练网络权值。

为了将图像信息输入神经网络，图像数字化为 256 灰度级，形成 256×256 象素位图。线性前馈网络只有一层，含 8 个神经元，每个接收 64 个输入。这些输入由 8×8 非重迭的图像块

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

组成，通过从左到右，从上到下的扫描获得。用 GHA 训练网络权值，与每个神经元相联系的 64 个权值构成 8×8 模板，如图 7.7。图中白色象素代表正权值(兴奋连接)；黑色象素代表负权值(抑制连接)；灰色象素表示零权值(即未连接)。

通过以下步骤，完成对图 7.6 中的图像的压缩：

(a)　　(b)

图 7.6　(a)经神经网络压缩得到的图像　(b)神经网络压缩率决定下的重构图像
(选自 Sanger, T.D., Neural Network, vol.12, pp.459-473, 1989. Elsevier Science 授权)

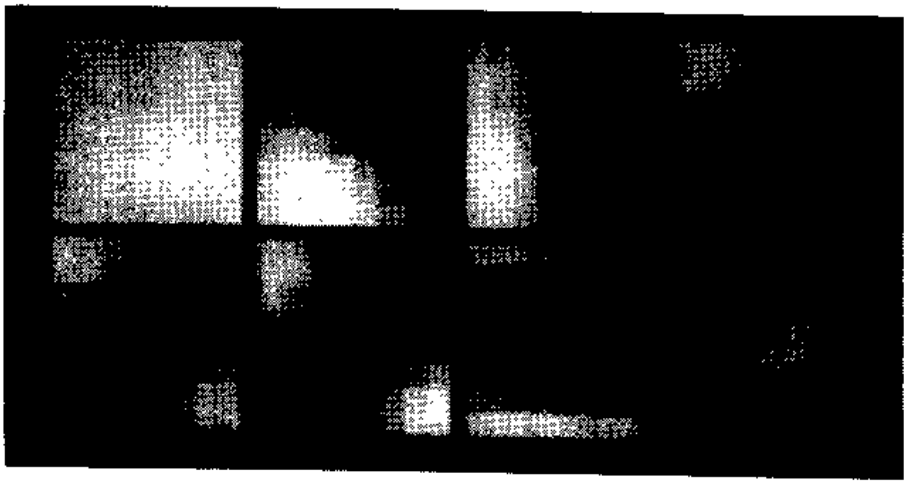


图 7.7　通过对单层网络的权值进行训练得到的 8 个 8×8 象素块

1. 将每个 8×8 图像块送入网络，对 8 个神经元中的每个神经元结果输出进行计算，得到图像压缩的 8 个系数。

2. 采用计算整个图像各系数方差的算法，大体确定用来代替系数所需的比特位数。

3. 然后，统一量化系数，头 2 个模板用 5 比特表示；第 3 个模板用 3 比特；余下 5 个模板各用 2 比特。

4. 最终每个 8×8 象素块被压缩成 23 比特数据，压缩结果为每个象素用 0.36 比特表示。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication: luzhenbo@yahoo.com.cn

图 7.6(b)给出了用量化系数方法压缩后经重建后的小孩图像。首先,所有模板通过量化系数进行加权,然后,将这些结果合在一起重建每块图像。

更为有趣的是,我们可以将相同的模板应用到与小孩图像在统计意义上相似的另一图像上。图 7.8 给出了这样的图像(画面上是一只狗),此时系数被量化如下:

1. 头两个模板各用 7 比特。
2. 第三个模板用 5 比特。
3. 第四个模板用 4 比特。
4. 余下四个模板各用 3 比特。

其结果是,每个 8×8 的象素块被压缩成 35 比特数据,压缩结构为每象素占 0.55 比特。使用从图 7.7 获得的量化系数,得到狗的重构图像如图 7.8(b)。

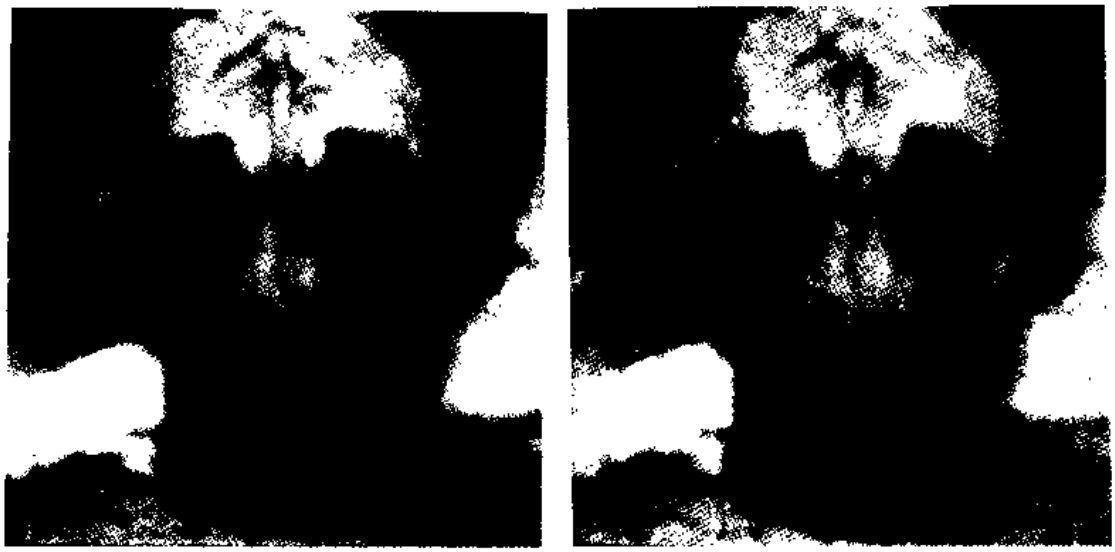


图 7.8 (a)使用由三个小孩图像训练后所得到的模板进行压缩的狗的图像 (b)重构后的图像
(选自 Sanger,T.D.,Neural Network,vol.12,pp.459-473,1989. Elsevier Science 授权)

Karhunen-Loeve 变换现在也大量用于手写字符识别问题中[Grother,1992]。学习过程包含 944 个不同书写者的 76,753 个训练字符。此系统在一次 15,000 字符的测试中,识别率高达 96.1%。应用系统包括一个多级神经网络,其第一级进行主分量特征提取。由于特征矢量可用神经方法计算得到(见 7.4 节中的讨论),它们也可被认为是一训练好的权值层。这里采用的方法是,常常首先用 Karhunen-Loeve 变换计算主分量,然后,直接用特征矢量作为主分量特征提取阶段的权植。这样寻求特征分量的繁重的计算任务就只局限在训练阶段了,并且,不影响运转的性能。此阶段后就是使用共轭梯度算法来训练前馈感知网络。注意到第一特征提取阶段是线性的,面紧接的分类阶段却不是。分级网络使用的是单一隐含层。

使用两种拓扑结构:

1.A 32 输入×32 隐层结点×10 输出——通过使用头 32 个主分量达到 93.7% 的精确度。

2.A 48 输入×48 隐层结点×10 输出——通过使用头 48 个主分量达到 94.7% 的精确度。

我们认为,K-L 变换一般用来处理原始输入数据(如位图)。然而实际并不是必须这样。事实上,当 K-L 变换用于处理曾经计算过的特征模式集合时,可以达到相似的目的。问题是,

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo
Communication & Cooperation: luzhenbo@yahoo.com.cn

某些输入数据中的相关信息未出现于所选的特征集合中，在统计意义上存在丢失的可能。针对这种情况，我们该怎么办？在很容易找到某些不变特征的情况下，可以找到一种答案，通过使设计的特征集合对于所需特性不变，来达到保存特性目的。另外，主分量分析还可以用来在不丢失所需不变性的情况下，在统计意义上最大限度地降低输入矢量的维数。

参考书与文献

Anderson, T. W., *An Introduction to Multivariate Statistical Analysis*, John Wiley & Sons, New York, 1984.

Duda, R. O. and Hart, P. E., *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, 1973.

Fu, L., *Neural Networks in Computer Intelligence*, McGraw-Hill, New York, 1994.

Fukunaga, K., *Introduction to Statistical Pattern Recognition*, Academic Press, New York and London, 1972; 2nd ed., 1985.

Gonzalez, R. C. and Woods, R. E., *Digital Image Processing*, Addison-Wesley, Reading, MA, 1992.

Grother, P. J., "Karhunen-Loeve Feature Extraction for Neural Handwritten Character Recognition," NISTR 4824, U.S. Dept. of Commerce, pp. 1–12, 1992.

Haykin, S., *Neural Networks: A Comprehensive Foundation*, Macmillan College Publishing, New York, 1994.

Hotelling, H., "Analysis of a complex of statistical variables into principal components," *J. Educ. Psychol.*, vol. 24, pp. 417–441, 498–520, 1933.

Jolliffe, I. T., *Principal Component Analysis*, Springer-Verlag, New York, 1986.

Karhunen, K., "Uber lineare methoden in der Wahrscheinlichkeitsrechung," Annales Academiae Scientiarum Fennicae, Series A1: Mathematica-Physica, vol. 37, pp. 3–79, 1947 (Translation: Rep. T-131, RAND Corp., Santa Monica, CA, 1960).

Koschman, A., "On the filtering of nonstationary time series," *Proc. 1954 Natl. Electron. Conf.*, p. 126, 1954.

Kramer, H. P. and Mathews, M. V., "A linear coding for transmitting a set of correlated signals," *IRE Trans. Inf. Theory*, vol. IT-2, pp. 41–46, 1956.

Kung, S. Y., *Digital Neural Networks*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

Kung, S. Y. and Diamantaras, C. I., "A neural network learning algorithm for adaptive principal component extraction (APEX)," *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, vol. 2, pp. 861–864, 1990.

Linsker, R., "Self-organization in a perceptual network," *Computer*, vol. 21, pp. 105–117, 1988.

Loeve, M., "Fonctions Aleatoires de Second Ordre," in P. Levy (Ed.), *Processus Stochastiques et Mouvement Brownien*, Hermann, Paris, 1948.

Loeve, M., *Probability Theory*, 3rd ed., Van Nostrand, New York, 1963.

Oja, E., "A simplified neuron model as a principal component analyzer," *J. Math. Biol.*, vol. 15, pp. 267–273, 1982.

Preisendorfer, R. W., *Principal Component Analysis in Meteorology and Oceanography*, Elsevier, New York, 1988.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, Cambridge, 1992.

Sanger, T. D., "Optimal unsupervised learning in a single-layer feedforward neural network," *Neural Networks*, vol. 12, pp. 459–473, 1989

Schalkoff, P. J., *Pattern Recognition Statistical Structural and Neural Approaches*, John Wiley & Sons, New York, 1992.

Watanabe, S., *Pattern Recognition: Human and Mechanical*, John Wiley & Sons, New York, 1985.

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

第八章 Kohonen 网络和学习矢量量化

8.1 概 述

生物系统的共同特征是具有在无导师的情况下,从其自然环境中得出规律,并且按照规律去做的能力。同样,这也是自组织神经网络的目标:即在无监督的情况下,从输入数据中找出有意义的规律来。在第三、六和七章中,我们重点讨论了如何为便于识别器的处理,在某种意义上可以简化外部环境数据的有关机制。尤其是第六、七章,讨论了特征提取问题,这些特征用于有监督识别器网络的输入。相应地,在自组织网络中考虑的特征检测的任务,是由网络自身完成的。第四、五章探讨了几种可用的有监督学习网络。

本章集中讨论自组织系统,该系统能像生物系统一样直接从周围环境中发现其结构、模式或特征。这就是 Kohonen 自组织特征映射(SOFM)。特别应该指出,它是受视网膜皮层的生物功能的启发而提出的。另外,SOFM 也部分受到生物特性的影响,而具有在网络输出层内按几何中心或特征进行聚类的独特性质。也就是说,在特征空间中具有较小的欧氏距离的相近的特征,将由 SOFM 的几何上相互接近的输出神经元产生输出响应。

然而,我们注意到这样的自组织行为并不局限于神经网络的范围。在这一章中,从统计模式识别中的聚类算法(K-均值算法)来开始我们的研究。这种安排看起来有点特别,但是我们观察到 K-均值算法很明显具有自组织特征,并且聪明的读者也将能注意到这一算法和随后要讨论的神经技术之间的相似性。

本章中提出的网络与使用竞争学习的网络是有区别的。竞争学习可描述成这样一个过程:输出层神经元互相之间竞争,以便能够激活,对给定输入模式作出响应。在我们所描述的网络中,激活的神经元是权矢量与输入空间中当前的模式矢量最接近的神经元。这又使人回想起 K-均值方法。在 K-均值方法中,相对输入模式矢量有最短欧氏范数距离的聚类中心获得该模式(并且赢得对该模式响应的权力)。

这里提出的网络的另一种共同的特点是它们都采用某种形式的侧反馈。神经元通过侧反馈对输出层神经元的有限区域或邻域的作用,来改变网络的性能。侧反馈可以通过一组连接以及输出层神经元的连接权值来建立。在典型情况下,当从源神经元到目标神经元的半径增加时,权值将减小。如果权值是负的,其结果将趋向于抑制其它输出层神经元的激活。这种侧抑制可看作是前面描述的胜者取全部的策略的发展。

Kohonen 的 SOFM 采用侧反馈的原因稍微不同,它的目的是为了减少输出层神经元中的拓扑组织。输出层神经元的典型结构,是由一个一维或二维的网格或阵列结构构成。更高维数的结构也是可能的,但是使用的很少。侧反馈的作用是使得相近的特征能够在输出层阵列几何相近的位置上表现出来。

本章在最后介绍学习矢量量化网络。这一 Kohonen 模型的变形之所以令人感兴趣,在于它溶合了自组织和有导师的技术。尽管仍然采用竞争学习,但是其产生方式是有教师监督的。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Copyright: luzhenbo@yahoo.com.cn

用这种方法意味着,竞争学习是在由训练输入指定的各类中局部地发生的。

8.2 K-均值算法

n 维模式矢量,可以被认为是代表一个 n 维欧氏空间中的点。建立模式矢量之间相似性测度的最明了的方法之一,是衡量它们互相之间的接近程度。图 8.1 说明了这一点。K-均值算法是按照最小距离进行聚类的许多聚类技术之一。简单地说,如果代表各点的矢量几何上互相接近,那么在某种意义上可被看作属于一类:通过联系合而为一。在介绍 K-均值算法的详细操作之前,有必要介绍一下更准确的距离测度的概念。矢量的欧氏范数,$\mathbf{x}=[x_1, x_2, \cdots\cdots, x_n]^T$,定义如下:

$$\|\mathbf{x}\| = \left[\sum_{i=1}^{n} x_i^2\right]^{1/2} \tag{8-1}$$

公式(8-1)表明了矢量 $\mathbf{x}$ 的长度。既然我们想知道在模式空间中两个矢量间的距离或长度,我们就仅需要把公式(8-1),应用到如下矢量差中去:

$$\|\mathbf{x}-\mathbf{z}\| = \left[\sum_{i=1}^{n} (x_i - z_i)\right]^{1/2} \tag{8-2}$$

其中,$\mathbf{x}$ 和 $\mathbf{z}$ 是 n 维的模式矢量。

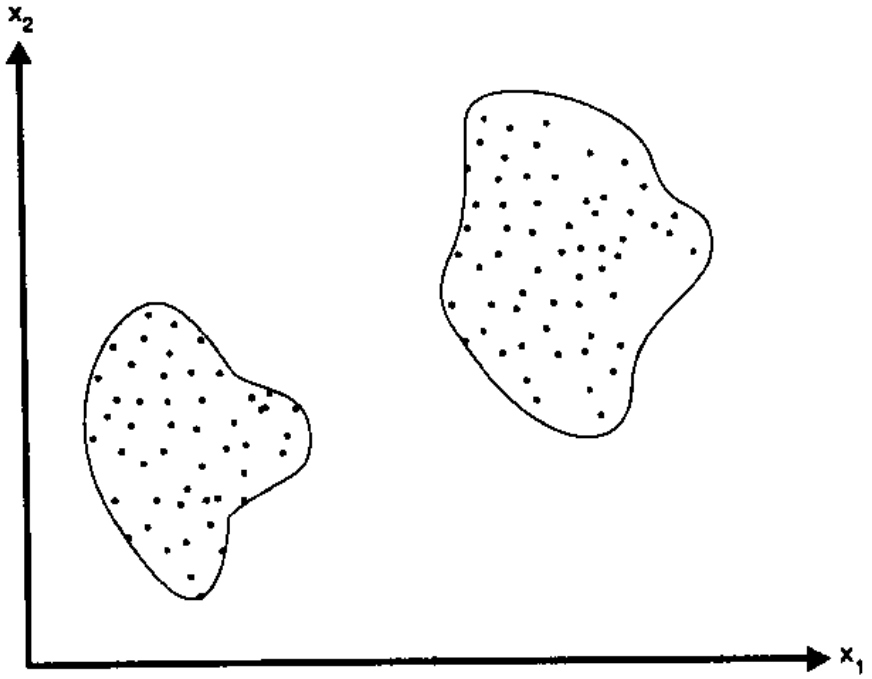


图 8.1 按照接近程度很容易区分的模式类别

既然已经建立了一种模式相似度的测量方法,那么必须着手建立一个过程。通过该过程把各模式分成类。也就是说,需要一个建立一系列类别(与聚类中心相联系)的过程,以便于用输入矢量和最接近于聚类中心的矢量之间的距离来标志该矢量。K-均值算法就代表这样的一个方法。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

K-均值方法假定将被用来表示样本空间的聚类中心的个数是预先知道的。这种假定本身在某种程度上限制了这一方法的利用。其它的有关最小距离统计学聚类技术的方法，减少了这种困难，但是又有其它的问题，比如对输入数据顺序很敏感的问题。详见 10.3 中的矢量量化。

讨论 K-均值过程以前，首先明确一些术语。令 $\mathbf{x}^{(p)}$ 代表第 p 个输入空间矢量。所有的输入矢量的集合表示成 $\{\mathbf{x}^{(1)},\mathbf{x}^{(2)},\cdots\cdots,\mathbf{x}^{(p)}\}$。矢量 $\mathbf{z}$ 代表 K 个类别的聚类中心，也就是说，它指出了聚类中心在欧氏空间中所处的位置。因为有 K 个类别，所以也就有 K 个聚类中心：$Z_1,Z_2,\cdots,Z_K$。最后用符号 $S_j=\{\mathbf{x}|\mathbf{x}$ 最接近于类别 $j\}$ 来代表所有属于第 j 个聚类中心的样本的集合。K-均值算法按如下步骤实现：

步骤 1　初始化：

设置类别数 K。每个类别的聚类中心赋初值：

$$\{\mathbf{z}_1(l),\mathbf{z}_2(l),\cdots\cdots,\mathbf{z}_k(l)\}$$

其中，$\mathbf{z}_j(l)$ 代表第 l 次迭代的聚类中心值。初始值可以是任意的，但通常都设置成样本矢量的前 K 个值。

步骤 2　样本划分：

划分所有的样本矢量。通过这一步使每个样本矢量 $\mathbf{x}^{(p)}$ 与 K 类中之一相联系，其划分条件为：

$$\mathbf{x}^{(p)}\in S_j(l),\text{如果}\|\mathbf{x}^{(p)}-\mathbf{z}_j(l)\|<\|\mathbf{x}^{(p)}-\mathbf{z}_i(l)\| \tag{8-3}$$

对于所有的 $i=1,2,\cdots\cdots,K,i\neq j$

其中，$S_j(l)$ 代表第 l 次迭代时类别 j 的全体。公式(8-3)中的关系可以用任意方法判定。

步骤 3　计算新的聚类中心：

用在步骤 2 中建立的新类的所有成员集合，来重新计算每类的中心位置，以便使从类别中的每个矢量到新的聚类中心的距离之和最小。特别地，我们希望最小化 J_j：

$$J_j=\sum_{\mathbf{x}^{(p)}\in S_j(l)}\|\mathbf{x}^{(p)}-\mathbf{z}_j(l+1)\|^2,j=1,2,\cdots,K \tag{8-4}$$

$\mathbf{z}_j(l+1)$ 是使公式(8-4)最小化的所有样本，$S_j(l)$ 的平均值。因此，新的聚类中心用如下的公式(8-5)计算：

$$\mathbf{z}_j(l+1)=\frac{1}{N_j}\sum_{\mathbf{x}^{(p)}\in S_j(l)}\mathbf{x}^{(p)} \tag{8-5}$$

其中，N_j 是步骤 2 中属于 S_j 的样本矢量的数量。

步骤 4　检查收敛：

收敛情形是在步骤 3 中没有任何聚类中心再变化其位置。这种情况在数学上可表示成：

$$\mathbf{z}_j(l+1)=\mathbf{z}_j(l)\qquad j=1,2,\cdots,K \tag{8-6}$$

如果满足公式(8-6)，那么就已经收敛了。否则再回到步骤 2 继续迭代。

许多因素都可能会影响 K-均值算法的性能。其中包括聚类中心的数量 K、初始聚类中心的选择和输入数据的几何分布特性。这就需要大量实验来确定 K 值的选择及初始化参数的选择。虽然没有收敛性的标准证明，但是，当数据的特性与采用最小距离作为相似性测度的假设所对应的情况一致时，K-均值算法可能会获得较好的性能。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
E-mail:luzhenbo@yahoo.com.cn

清单 8.1 列出了实现 K-均值算法的全部程序。随后,在 8.2.1 节中,将用 K-均值算法举一个例子。

清单 8.1

```
/*************************************************************************
*                                                                        *
* KMEANS                                                                 *
*                                                                        *
*************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <math.h>

// FUNCTION PROTOTYPES


// DEFINES
#define     SUCCESS      1
#define     FAILURE      0
#define     TRUE         1
#define     FALSE        0
#define     MAXVECTDIM   20
#define     MAXPATTERN   20
#define     MAXCLUSTER   10
// ***** Defined structures & classes *****
struct aCluster {
  double    Center[MAXVECTDIM];
  int       Member[MAXPATTERN];          //Index of Vectors belonging to this cluster
  int       NumMembers;
};

struct aVector {
  double    Center[MAXVECTDIM];
  int       Size;
};

class System {
private:
  double    Pattern[MAXPATTERN][MAXVECTDIM+1];
  aCluster  Cluster[MAXCLUSTER];
  int       NumPatterns;                 // Number of patterns
  int       SizeVector;                  // Number of dimensions in vector
  int       NumClusters;                 // Number of clusters
  void      DistributeSamples();         // Step 2 of K-means algorithm
  int       CalcNewClustCenters();       // Step 3 of K-means algorithm
  double    EucNorm(int, int);           // Calc Euclidean norm vector
  int       FindClosestCluster(int);     //ret indx of clust closest to pattern
                                         //whose index is arg
public:
  system();
  int LoadPatterns(char *fname);         // Get pattern data to be clustered
  void InitClusters();                   // Step 1 of K-means algorithm
  void RunKMeans();                      // Overall control K-means process
  void ShowClusters();                   // Show results on screen
  void SaveClusters(char *fname);        // Save results to file
};

int System::LoadPatterns(char *fname){
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
  FILE *InFilePtr;
  int   i,j;
  double x;
if((InFilePtr = fopen(fname, "r")) == NULL)
   return FAILURE;
fscanf(InFilePtr, "%d", &NumPatterns);          // Read # of patterns
fscanf(InFilePtr, "%d", &SizeVector);           // Read dimension of vector
fscanf(InFilePtr, "%d", &NumClusters);          // Read # of clusters for K-Means
for (i=0; i<NumPatterns; i++) {                 // For each vector
  for (j=0; j<SizeVector; j++) {                // create a pattern
    fscanf(InFilePtr,"%lg",&x);                 // consisting of all elements
    Pattern[i][j]=x;
   printf("Pattern[%d][%d]=%f\n",i,j,Pattern[i][j]);
    } /* endfor */
  } /* endfor */
printf("\n");
return SUCCESS;
}
//*****************************************************************************
// InitClusters                                                               *
//  Arbitrarily assign a vector to each of the K clusters                     *
//  We choose the first K vectors to do this                                  *
//*****************************************************************************
void System::InitClusters(){
int i,j;
printf("Initial cluster centers:\n");
for (i=0; i<NumClusters; i++) {
  Cluster[i].Member[0]=i;
  for (j=0; j<SizeVector; j++) {
    Cluster[i].Center[j]=Pattern[i][j];
    printf("Cluster[%d].Center[%d]=%f\n",i,j,Cluster[i].Center[j]);
    } /* endfor */
  } /* endfor */
printf("\n");
}

void System::RunKMeans(){
 int converged;
converged=FALSE;
while (converged==FALSE) {
  DistributeSamples();
  converged=CalcNewClustCenters();
// converged=TRUE;
  } /* endwhile */
}

double System::EucNorm(int p, int c){           // Calc Euclidean norm of vector difference
double dist;                                    // between pattern vector, p, and cluster
int i;                                          // center, c.
dist=0;
for (i=0; i<SizeVector ;i++){
  dist += (Cluster[c].Center[i]-Pattern[p][i])*(Cluster[c].Center[i]-Pattern[p][i]);
  } /* endfor */
//dist = sqrt(dist);
return dist;
}

int System::FindClosestCluster(int pat){
  int i, ClustID;
  double MinDist, d;
MinDist =9.9e+99;
ClustID=-1;
for (i=0; i<NumClusters; i++) {
  d=EucNorm(pat,i);
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication E-mail: luzhenbo@yahoo.com.cn


```
//  printf("Distance from pattern %d to cluster %d = %f \n",pat,i,d);
  if (d<MinDist) {
    MinDist=d;
    ClustID=i;
    } /* endif */
  } /* endfor */
if (ClustID<0) {
  printf("Aaargh");
  exit(0);
  } /* endif */
return ClustID;
}

void System::DistributeSamples(){
int i,pat,Clustid,MemberIndex;
//Clear membership list for all current clusters
for (i=0; i<NumClusters;i++){
  Cluster[i].NumMembers=0;
  }
for (pat=0; pat<NumPatterns; pat++) {
  //Find cluster center to which the pattern is closest
  Clustid= FindClosestCluster(pat);
  printf("patern %d assigned to cluster %d\n",pat,Clustid);
  //post this pattern to the cluster
  MemberIndex=Cluster[Clustid].NumMembers;
  Cluster[Clustid].Member[MemberIndex]=pat;
  Cluster[Clustid].NumMembers++;
  } /* endfor */
}

int System::CalcNewClustCenters(){
  int ConvFlag,VectID,i,j,k;
  double tmp[MAXVECTDIM];
ConvFlag=TRUE;
for (i=0; i<NumClusters; i++) {                    //for each cluster
  for (j=0; j<SizeVector; j++) {                   // clear workspace
    tmp[j]=0.0;
    } /* endfor */
  for (j=0; j<Cluster[i].NumMembers; j++) {        //traverse member vectors
    VectID=Cluster[i].Member[j];
    for (k=0; k<SizeVector; k++) {                 //traverse elements of vector
      printf("Cluster[%d]  Pattern[%d][%d]=%f,
Member_ID=%d\n",i,VectID,k,Pattern[VectID][k],VectID);
//      x=Pattern[VectID][k];
//      tmp[k] = tmp[k]+x;                         // add (member) pattern elmnt into temp
      tmp[k] += Pattern[VectID][k];                // add (member) pattern elmnt into temp
      } /* endfor */
    } /* endfor */
  for (k=0; k<SizeVector; k++) {                   //traverse elements of vector
    tmp[k]=tmp[k]/Cluster[i].NumMembers;
    if (tmp[k] != Cluster[i].Center[k])
      ConvFlag=FALSE;
    Cluster[i].Center[k]=tmp[k];
    } /* endfor */
  } /* endfor */
return ConvFlag;
}

void System::ShowClusters(){
  int cl;
for (cl=0; cl<NumClusters; cl++) {
  printf("\nCLUSTER %d ==>[%f,%f]\n", cl,Cluster[cl].Center[0],Cluster[cl].Center[1]);
  } /* endfor */
}
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
    void System::SaveClusters(char *fname){
    }

main(int argc, char *argv[]) {
  System kmeans;
if (argc<3) {
  printf("USAGE: KMEANS PATTERN_FILE(input) CLUSTER_FILE(output)\n");
  exit(0);
  }
if (kmeans.LoadPatterns(argv[1])==FAILURE ){
  printf("UNABLE TO READ PATTERN_FILE:%s\n",argv[1]);
  exit(0);
  }
kmeans.InitClusters();
kmeans.RunKMeans();
kmeans.ShowClusters();
}
```

8.2.1 K-均值算法举例

下面说明 K-均值算法的应用。在这一示例中,将利用 K-均值算法来对图 8.2 中所示的点聚类。

与图 8.2 中相应的点的二维模式矢量如下:

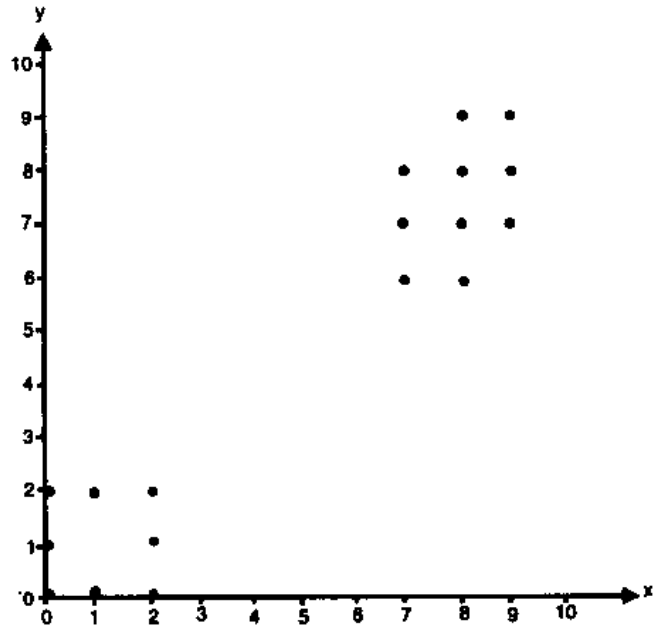


图 8.2 用 K-均值算法聚类的样本数据

模式[0]=(0.000,0.000)
模式[1]=(1.000,0.000)
模式[2]=(0.000,1.000)
模式[3]=(2.000,1.000)

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

模式[4]=(1.000,2.000)
模式[5]=(2.000,2.000)
模式[6]=(2.000,0.000)
模式[7]=(0.000,2.000)
模式[8]=(7.000,6.000)
模式[9]=(7.000,7.000)
模式[10]=(7.000,8.000)
模式[11]=(8.000,6.000)
模式[12]=(8.000,7.000)
模式[13]=(8.000,8.000)
模式[14]=(8.000,9.000)
模式[15]=(9.000,7.000)
模式[16]=(9.000,8.000)
模式[17]=(9.000,9.000)

初始聚类中心取为前两个模式矢量,因此初始聚类中心是:

聚类中心[0]=(0.000,0.000)
聚类中心[1]=(1.000,0.000)

下面准备应用算法计算。对于第一次迭代,计算从最接近聚类中心的每个输入矢量开始。

第一次迭代

模式 0 到类别 0 的距离可计算成:
d=sqrt(.0000+.0000)=0.000000
模式 0 到类 1 的距离可计算成:
d=sqrt(1.0000+.0000)=1.000000
} 模式 0 可划分到类别 0 中

模式 1 到类别 0 的距离可计算成:
d=sqrt(1.0000+.0000)=1.000000
模式 1 到类别 1 的距离可计算成:
d=sqrt(.0000+.0000)=0.000000
} 模式 1 可划分到类别 1 中

模式 2 到类别 0 的距离可计算成:
d=sqrt(.0000+1.0000)=1.000000
模式 2 到类别 1 的距离可计算成:
d=sqrt(1.0000+1.0000)=1.414214
} 模式 2 可划分到类别 0 中

模式 3 到类别 0 的距离可计算成:
d=sqrt(4.0000+1.0000)=2.236068
模式 3 到类别 1 的距离可计算成:
d=sqrt(1.0000+1.0000)=1.414214
} 模式 3 可划分到类别 1 中

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

模式 4 到类别 0 的距离可计算成：
d＝sqrt(1.0000＋4.0000)＝2.236068
模式 4 到类别 1 的距离可计算成：
d＝sqrt(.0000＋4.0000)＝2.000000
模式 4 可划分到类别 1 中

模式 5 到类别 0 的距离可计算成：
d＝sqrt(4.0000＋4.0000)＝2.828427
模式 5 到类别 1 的距离可计算成：
d＝sqrt(1.0000＋4.0000)＝2.236068
模式 5 可划分到类别 1 中

模式 6 到类别 0 的距离可计算成：
d＝sqrt(4.0000＋.0000)＝2.000000
模式 6 到类别 1 的距离可计算成：
d＝sqrt(1.0000＋.0000)＝1.000000
模式 6 可划分到类别 1 中

模式 7 到类别 0 的距离可计算成：
d＝sqrt(.0000＋4.0000)＝2.000000
模式 7 到类别 1 的距离可计算成：
d＝sqrt(1.0000＋4.0000)＝2.236068
模式 7 可划分到类别 0 中

模式 8 到类别 0 的距离可计算成：
d＝sqrt(49.0000＋36.0000)＝9.219544
模式 8 到类别 1 的距离可计算成：
d＝sqrt(36.0000＋36.0000)＝8.485281
模式 8 可划分到类别 1 中

模式 9 到类别 0 的距离可计算成：
d＝sqrt(49.0000＋49.0000)＝9.899495
模式 9 到类别 1 的距离可计算成：
d＝sqrt(36.0000＋49.0000)＝9.219544
模式 9 可划分到类别 1 中

模式 10 到类别 0 的距离可计算成：
d＝sqrt(49.0000＋64.0000)＝10.630146
模式 10 到类别 1 的距离可计算成：
d＝sqrt(36.0000＋64.0000)＝10.000000
模式 10 可划分到类别 1 中

模式 11 到类别 0 的距离可计算成：
d＝sqrt(64.0000＋36.0000)＝10.000000
模式 11 到类别 1 的距离可计算成：
d＝sqrt(49.0000＋36.0000)＝9.219544
模式 11 可划分到类别 1 中

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and cooperation: luzhenbo@yahoo.com.cn

模式 12 到类别 0 的距离可计算成：
d = sqrt(64.0000 + 49.0000) = 10.630146
模式 12 到类别 1 的距离可计算成：
d = sqrt(49.0000 + 49.0000) = 9.899495
模式 12 可划分到类别 1 中

模式 13 到类别 0 的距离可计算成：
d = sqrt(64.0000 + 64.0000) = 11.313708
模式 13 到类别 1 的距离可计算成：
d = sqrt(49.0000 + 64.0000) = 10.630146
模式 13 可划分到类别 1 中

模式 14 到类别 0 的距离可计算成：
d = sqrt(64.0000 + 81.0000) = 12.041595
模式 14 到类别 1 的距离可计算成：
d = sqrt(49.0000 + 81.0000) = 11.401754
模式 14 可划分到类别 1 中

模式 15 到类别 0 的距离可计算成：
d = sqrt(81.0000 + 49.0000) = 11.401754
模式 15 到类别 1 的距离可计算成：
d = sqrt(64.0000 + 49.0000) = 10.630146
模式 15 可划分到类别 1 中

模式 16 到类别 0 的距离可计算成：
d = sqrt(81.0000 + 64.0000) = 12.041595
模式 16 到类别 1 的距离可计算成：
d = sqrt(64.0000 + 64.0000) = 11.313708
模式 16 可划分到类别 1 中

模式 17 到类别 0 的距离可计算成：
d = sqrt(81.0000 + 81.0000) = 12.727922
模式 17 到类别 1 的距离可计算成：
d = sqrt(64.0000 + 81.0000) = 12.041595
模式 17 可划分到类别 1 中

现在，所有的模式矢量都与一个聚类中心相联系在一起。下面的目的是要重新计算与新的成员集合相一致的聚类中心。新的聚类中心计算如下：

聚类中心 0 = (1/3)(0.000 + 0.000 + 0.000), (1/3)(0.000 + 1.000 + 2.000))

聚类中心 1 = (1/15)(1.000 + 2.000 + 1.000 + 2.000 + 2.000 + 7.000 + 7.000 + 7.000 + 8.000 + 8.000 + 8.000 + 8.000 + 9.000 + 9.000 + 9.000),
(1/15)(0.000 + 1.000 + 2.000 + 2.000 + 0.000 + 6.000 + 7.000 + 8.000 + 6.000 + 7.000 + 8.000 + 9.000 + 7.000 + 8.000 + 9.000)

所以，新的聚类中心变为：

聚类中心[0] = (0.000000, 1.000000)

聚类中心[1] = (5.866667, 5.333333)

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

第一次迭代已完成,因为聚类中心已改变,所以需第二次迭代。第二次迭代计算如下:

第二次迭代

模式 0 到类别 0 的距离可计算成:
d=sqrt(.0000+1.0000)=1.000000
模式 0 到类别 1 的距离可计算成:
d=sqrt(34.4178+28.4444)=7.928570
→ 模式 0 可划分到类别 0 中

模式 1 到类别 0 的距离可计算成:
d=sqrt(1.0000+1.0000)=1.414214
模式 1 到类别 1 的距离可计算成:
d=sqrt(23.6844+28.4444)=7.220034
→ 模式 1 可划分到类别 0 中

模式 2 到类别 0 的距离可计算成:
d=sqrt(.0000+.0000)=.000000
模式 2 到类别 1 的距离可计算成:
d=sqrt(34.4178+18.7778)=7.293528
→ 模式 2 可划分到类别 0 中

模式 3 到类别 0 的距离可计算成:
d=sqrt(4.0000+.0000)=2.000000
模式 3 到类别 1 的距离可计算成:
d=sqrt(14.9511+18.7778)=5.807658
→ 模式 3 可划分到类别 0 中

模式 4 到类别 0 的距离可计算成:
d=sqrt(1.0000+1.0000)=1.414214
模式 4 到类别 1 的距离可计算成:
d=sqrt(23.6844+11.1111)=5.898776
→ 模式 4 可划分到类别 0 中

模式 5 到类别 0 的距离可计算成:
d=sqrt(4.0000+1.0000)=2.236068
模式 5 到类别 1 的距离可计算成:
d=sqrt(14.9511+11.1111)=5.10511
→ 模式 5 可划分到类别 0 中

模式 6 到类别 0 的距离可计算成:
d=sqrt(4.0000+1.0000)=2.236068
模式 6 到类别 1 的距离可计算成:
d=sqrt(14.9511+28.4444)=6.587530
→ 模式 6 可划分到类别 0 中

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

模式 7 到类别 0 的距离可计算成：
d = sqrt(.0000 + 1.0000) = 1.000000
模式 7 到类别 1 的距离可计算成：
d = sqrt(34.4178 + 11.1111) = 6.747510
模式 7 可划分到类别 0 中

模式 8 到类别 0 的距离可计算成：
d = sqrt(49.0000 + 25.0000) = 8.602325
模式 8 到类别 1 的距离可计算成：
d = sqrt(1.2844 + .4444) = 1.314872
模式 8 可划分到类别 1 中

模式 9 到类别 0 的距离可计算成：
d = sqrt(49.0000 + 36.0000) = 1.000000
模式 9 到类别 1 的距离可计算成：
d = sqrt(1.2844 + 2.7778) = 2.015496
模式 9 可划分到类别 1 中

模式 10 到类别 0 的距离可计算成：
d = sqrt(49.0000 + 49.0000) = 9.899495
模式 10 到类别 1 的距离可计算成：
d = sqrt(1.2844 + 7.1111) = 2.8975
模式 10 可划分到类别 1 中

模式 11 到类别 0 的距离可计算成：
d = sqrt(64.0000 + 25.0000) = 9.433981
模式 11 到类别 1 的距离可计算成：
d = sqrt(4.5511 + .4444) = 2.2354074
模式 11 可划分到类别 1 中

模式 12 到类别 0 的距离可计算成：
d = sqrt(64.0000 + 36.0000) = 10.000000
模式 12 到类别 1 的距离可计算成：
d = sqrt(4.5511 + 2.7778) = 2.707192
模式 12 可划分到类别 0 中

模式 13 到类别 0 的距离可计算成：
d = sqrt(64.0000 + 49.0000) = 10.630146
模式 13 到类别 1 的距离可计算成：
d = sqrt(4.5511 + 7.1111) = 3.415000
模式 13 可划分到类别 1 中

模式 14 到类别 0 的距离可计算成：
d = sqrt(64.0000 + 64.0000) = 11.313708
模式 14 到类别 1 的距离可计算成：
d = sqrt(4.5511 + 13.4444) = 4.242117
模式 14 可划分到类别 1 中

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

模式 15 到类别 0 的距离可计算成：
d＝sqrt(81.0000＋36.0000)＝10.816654
模式 15 到类别 1 的距离可计算成：
d＝sqrt(9.8178＋2.7778)＝3.549022
} 模式 15 可划分到类别 1 中

模式 16 到类别 0 的距离可计算成：
d＝sqrt(81.0000＋49.0000)＝11.401754
模式 16 到类别 1 的距离可计算成：
d＝sqrt(9.8178＋7.1111)＝4.114473
} 模式 16 可划分到类别 1 中

模式 17 到类别 0 的距离可计算成：
d＝sqrt(81.0000＋64.0000)＝12.041595
模式 17 到类别 1 的距离可计算成：
d＝sqrt(9.8178＋13.4444)＝4.823093
} 模式 17 可划分到类别 1 中

现在新的聚类中心可计算成：

聚类中心 0＝(1/8)(0.000＋1.000＋0.000＋2.000＋1.000＋2.000＋2.000＋0.000),
(1/8)(0.000＋0.000＋1.000＋1.000＋2.000＋2.000＋0.000＋2.000)

聚类中心 1＝(1/10)(7.000＋7.000＋7.000＋8.000＋8.000＋8.000＋8.000＋9.000＋9.000＋9.000),(1/10)(6.000＋7.000＋8.000＋6.000＋7.000＋8.000＋9.000＋7.000＋8.000＋9.000)

因而新的聚类中心为：

聚类中心[0]＝(1.000000,1.000000)
聚类中心[1]＝(8.000000,7.500000)

第二次迭代已完成，聚类中心已改变，因而我们需进行第三次迭代。

第三次迭代

模式 0 到类别 0 的距离可计算成：
d＝sqrt(1.0000＋1.0000)＝1.414214
模式 0 到类别 1 的距离可计算成：
d＝sqrt(64.0000＋56.2500)＝10.965856
} 模式 0 可划分到类别 0 中

模式 1 到类别 0 的距离可计算成：
d＝sqrt(.0000＋1.0000)＝1.000000
模式 1 到类别 1 的距离可计算成：
d＝sqrt(49.0000＋56.2500)＝10.259142
} 模式 1 可划分到类别 0 中

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and Cooperation: luzhenbo@yahoo.com.cn

模式 2 到类别 0 的距离可计算成：
d = sqrt(1.0000 + .0000) = 1.000000
模式 2 到类别 1 的距离可计算成：
d = sqrt(64.0000 + 42.2500) = 10.307764
模式 2 可划分到类别 0 中

模式 3 到类别 0 的距离可计算成：
d = sqrt(1.0000 + .0000) = 1.000000
模式 3 到类别 1 的距离可计算成：
d = sqrt(36.0000 + 42.2500) = 8.845903
模式 3 可划分到类别 0 中

模式 4 到类别 0 的距离可计算成：
d = sqrt(.0000 + 1.0000) = 1.000000
模式 4 到类别 1 的距离可计算成：
d = sqrt(49.0000 + 30.2500) = 8.902247
模式 4 可划分到类别 0 中

模式 5 到类别 0 的距离可计算成：
d = sqrt(1.0000 + 1.0000) = 1.414214
模式 5 到类别 1 的距离可计算成：
d = sqrt(36.0000 + 30.25000) = 8.139410
模式 5 可划分到类别 0 中

模式 6 到类别 0 的距离可计算成：
d = sqrt(1.0000 + 1.0000) = 1.414214
模式 6 到类别 1 的距离可计算成：
d = sqrt(36.0000 + 56.2500) = 9.604686
模式 6 可划分到类别 0 中

模式 7 到类别 0 的距离可计算成：
d = sqrt(1.0000 + 1.0000) = 1.414214
模式 7 到类别 1 的距离可计算成：
d = sqrt(64.0000 + 30.2500) = 9.708244
模式 7 可划分到类别 0 中

模式 8 到类别 0 的距离可计算成：
d = sqrt(36.0000 + 25.0000) = 7.810250
模式 8 到类别 1 的距离可计算成：
d = sqrt(1.0000 + 2.2500) = 1.802776
模式 8 可划分到类别 1 中

模式 9 到类别 0 的距离可计算成：
d = sqrt(36.0000 + 36.0000) = 8.485281
模式 9 到类别 1 的距离可计算成：
d = sqrt(1.0000 + .2500) = 1.118034
模式 9 可划分到类别 1 中

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

模式 10 到类别 0 的距离可计算成：
d＝sqrt(36.0000＋49.0000)＝9.219544
模式 10 到类别 1 的距离可计算成：
d＝sqrt(1.0000＋.2500)＝1.118034
模式 10 可划分到类别 1 中

模式 11 到类别 0 的距离可计算成：
d＝sqrt(49.0000＋25.0000)＝8.602325
模式 11 到类别 1 的距离可计算成：
d＝sqrt(.0000＋2.2500)＝1.500000
模式 11 可划分到类别 1 中

模式 12 到类别 0 的距离可计算成：
d＝sqrt(49.0000＋36.0000)＝9.219544
模式 12 到类别 1 的距离可计算成：
d＝sqrt(.0000＋.2500)＝.500000
模式 12 可划分到类别 1 中

模式 13 到类别 0 的距离可计算成：
d＝sqrt(49.0000＋49.0000)＝9.899495
模式 13 到类别 1 的距离可计算成：
d＝sqrt(.0000＋.2500)＝.500000
模式 13 可划分到类别 1 中

模式 14 到类别 0 的距离可计算成：
d＝sqrt(49.0000＋64.0000)＝10.630146
模式 14 到类别 1 的距离可计算成：
d＝sqrt(.0000＋2.2500)＝1.500000
模式 14 可划分到类别 1 中

模式 15 到类别 0 的距离可计算成：
d＝sqrt(64.0000＋36.0000)＝10.0000000
模式 15 到类别 1 的距离可计算成：
d＝sqrt(1.0000＋.2500)＝1.118034
模式 15 可划分到类别 1 中

模式 16 到类别 0 的距离可计算成：
d＝sqrt(64.0000＋49.0000)＝10.630146
模式 16 到类别 1 的距离可计算成：
d＝sqrt(1.0000＋.2500)＝1.118034
模式 16 可划分到类别 1 中

模式 17 到类别 0 的距离可计算成：
d＝sqrt(64.0000＋64.0000)＝11.313708
模式 17 到类别 1 的距离可计算成：
d＝sqrt(1.0000＋2.2500)＝1.802776
模式 17 可划分到类别 1 中

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication: luzhenbo@yahoo.com.cn

新的聚类中心计算如下：

聚类中心 0=(1/8/)(0.000+1.000+0.000+2.000+1.000+2.000+2.000+0.000),
(1/8)(0.000+0.000+1.000+1.000+2.000+2.000+0.000+2.000)

聚类中心 1=(1/10)(7.000+7.000+7.000+8.000+8.000+8.000+8.000+
9.000+9.000+9.000),(1/10)(6.000+7.000+8.000+6.000+7.000+
8.000+9.000+7.000+8.000+9.000)

所以，新的聚类中心为：

聚类中心[0]=(1.000000,1.000000)

聚类中心[1]=(8.000000,7.500000)

因为聚类中心没有变化，所以算法已经收敛了。因此，最后的聚类中心如下：

聚类中心 0=(1.000000,1.000000)

聚类中心 1=(8.000000,7.500000)

图 8.3 表明了在这个示例中类别成员与相应的聚类中心的联系情况。很明显，对于给定的数据，类别与它们所对应的聚类中心在直观上是合情合理的。

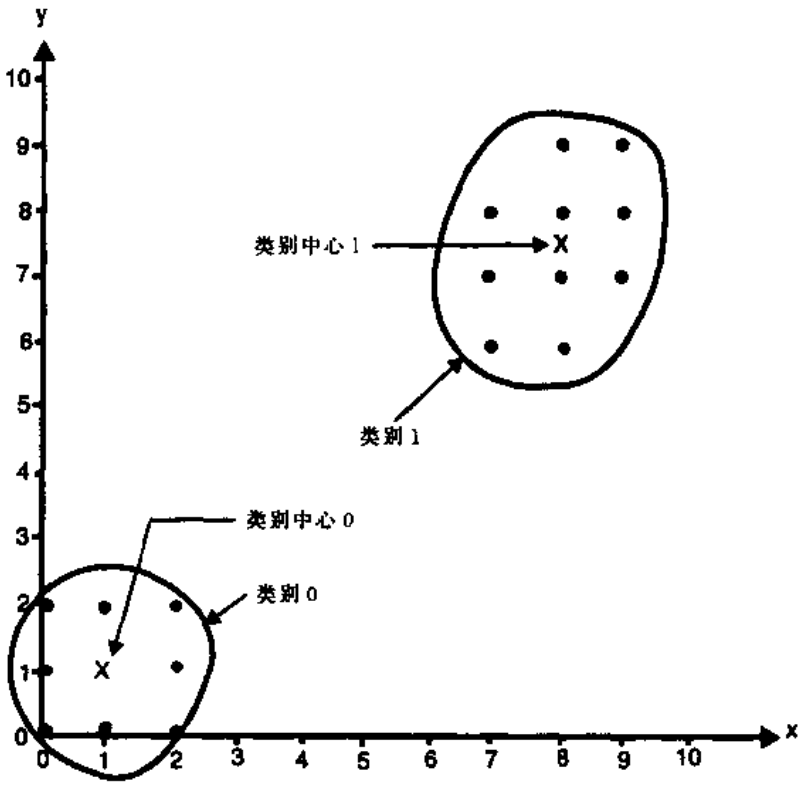


图 8.3 应用 K-均值算法获得的类别和聚类中心

8.3 Kohonen 模型介绍

在 8.2 节中，我们研究了聚类(也就是 K-均值算法中固有的自组织能力)。现在我们开始研究神经网络的自组织功能。下面将试着说明，Kohonen 模型结构像 K-均值算法一样具有识别聚类中心的能力。这种简单的介绍，省略了 Kohonen 模型在 SOFM 中的重要特性(尤其是

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

其局部侧反馈)。在这一简化的描述中,获胜神经元的选择是通过 MAXNET 方法来完成的。也就是说,有最大活性的神经元 net_j 成为获胜者。

一旦明确了这一简化模型的聚类性能,我们将继续讨论侧反馈,最后将讨论整个 SOFM。

Kohonen 网络结构由两层构成:输入层和 Kohonen 层。这两层是全连接的。每个输入层

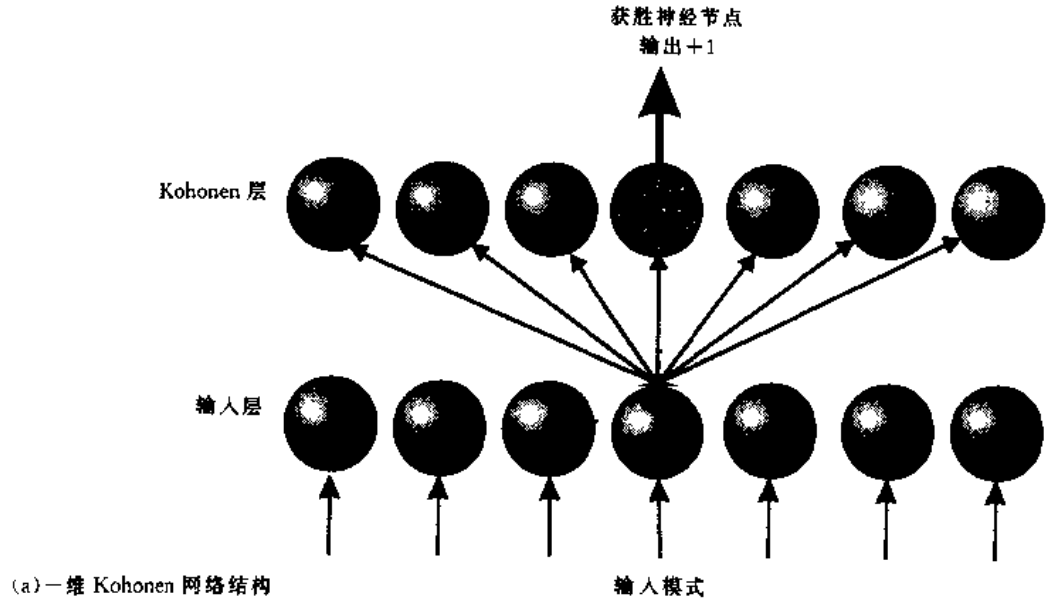


图 8.4(a) 一维 Konhonen 网络结构

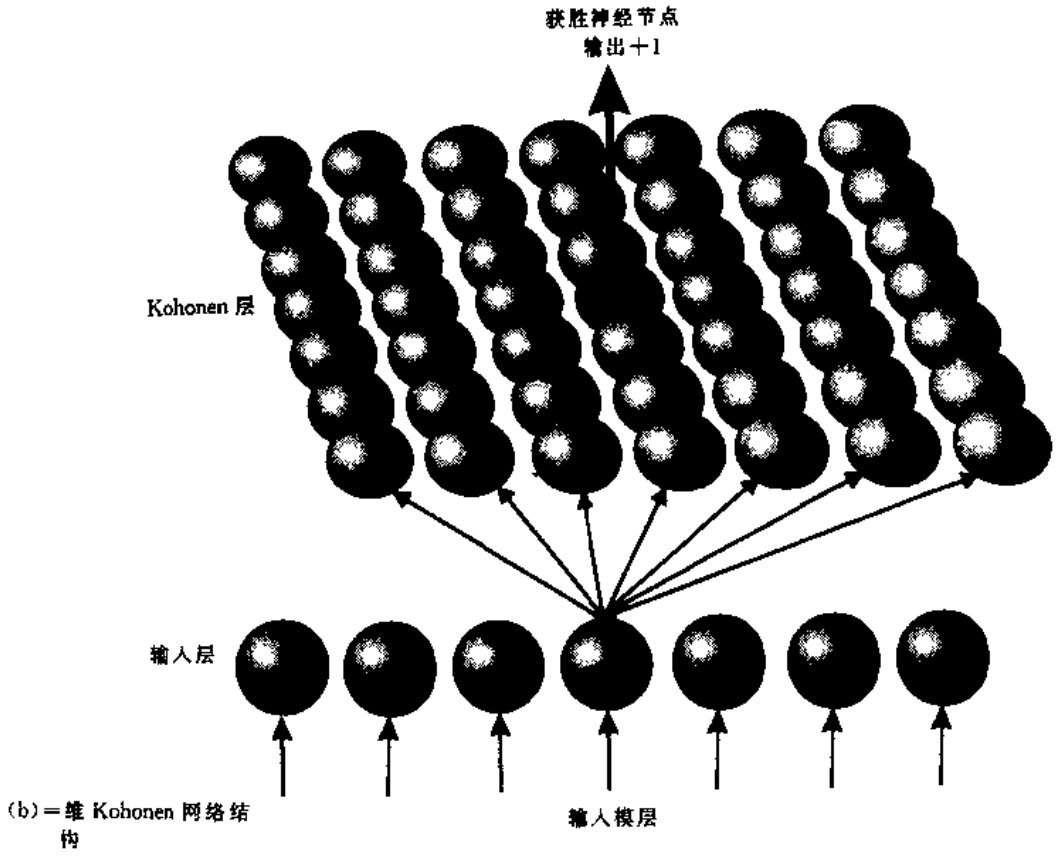


图 8.4(b) 二维 Konhonen 网络结构

神经元与每个输出层神经元有一前馈连接。图 8.4 示出了一维情形(图 8.4(a))和二维情形

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

(图 8.4(b))的 Kohonen 网络结构。尽管更高维的情形很少,但也是可能的,而且极难画出。假设输入是标准化的(即 $\|\mathbf{x}\| = 1$)。通常,到 Kohonen 层(即输出层)的输入,可用公式(8-7)计算:

$$I_j = \sum_{i=1}^{n} (\mathbf{W}_{ij}\mathbf{x}_i) \tag{8-7}$$

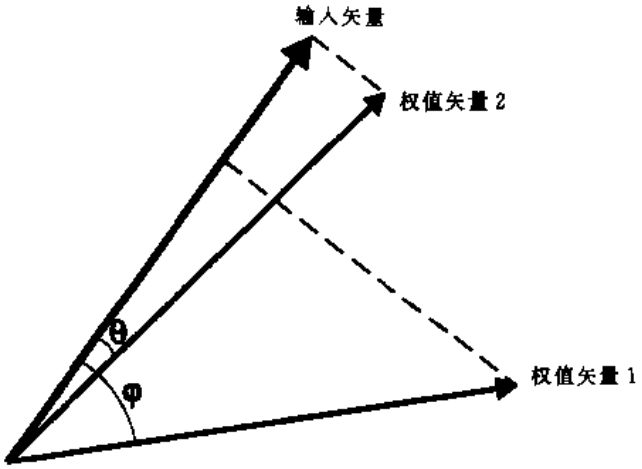


图 8.5 模式矢量与权值矢量点积,注意模式矢量
与最相似的权值矢量之间的夹角最小

应用胜者取全部的原则,获胜的输出层神经元将是有最大 I_j 的神经元。获胜神经元的输出是 $a+1$。Kohonen 层的所有其它神经元将无输出。事实上,公式(8-7)是神经元权值矢量和输入矢量之间的点积。因而这一方法选择一个获胜神经元,该神经元权值矢量与输入矢量的夹角要小于其它所有神经元所对应的点积(见图 8.5)。

另一种选择获胜神经元的方法,是简单地将权值矢量与输入矢量具有最小欧氏范数距离(即 $d_j = \|\mathbf{W}_j - \mathbf{x}\|$)的神经元作为获胜神经元。对于单位长度的矢量,这一方法与刚刚描述的方法,在选择同一神经元作为获胜者上是等价的。然而,利用欧氏距离来选择获胜神经元也许更有利,它不需要对权值和输入矢量进行标准化。

Kohonen 网络通过竞争学习(无监督)进行训练。当一矢量输入网络后,Kohonen 层中的神经元开始竞争,然后用上面介绍的方法之一选择获胜者(后面我们将看到竞争能通过内部神经元的侧反馈连接方便地传递)。获胜神经元根据下面公式训练:

$$\mathbf{W}_{ij}^{新} = \mathbf{W}_{ij}^{旧} + \eta(\mathbf{x}_i - \mathbf{W}_{ij}^{旧}) \tag{8-8}$$

其中,η 是学习参数或增益,典型地取小于 0.25 的初始值。公式(8-8)描述的学习被称为 Kohonen学习。

让我们选择标准化的矢量输入到网络中(每个矢量分量被矢量长度去除)。在这种情况下,权值矢量被随机地初始化,但它们也是标准化的。那么,获胜神经元的训练是按图 8.6 所说明的那样进行的。正像图中所示的那样,训练的结果是将权值矢量逐渐靠近输入矢量。要注意到由训练得出的权值不一定必须是单位长度的。权值矢量的重新标准化可认为是任选的,其影响预计会由所有输入矢量来抵消。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

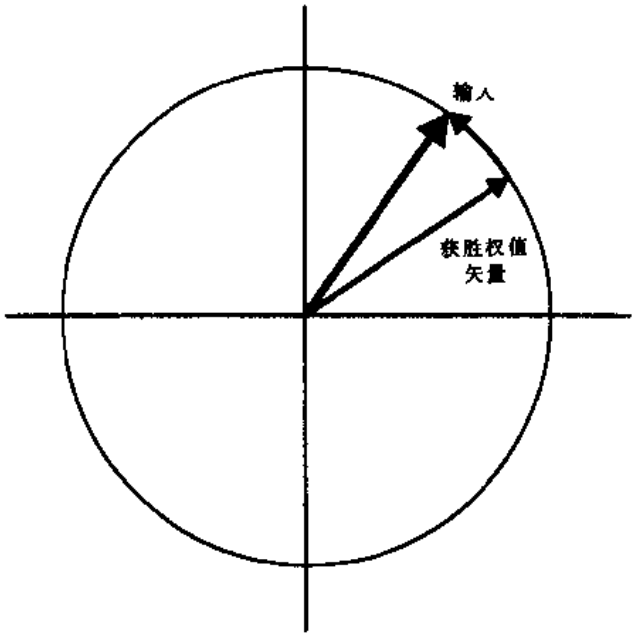


图 8.6 训练中获胜神经元权值矢量向模式矢量移动的示意图

清单 8.2

```
/*******************************
*KOHONEN NET                   *
*******************************
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

#define  MAXPATS     100
#define  MAXNEURONSIN 10
#define  MAXNEURONS   15

#define  MAXEPOCHS    1000
#define  ETAMIN       .001

unsigned int Random(int N) {
 unsigned int j;
j= (N*rand())/RAND_MAX;
if (j>=N) j=N;
return j;
}
class PATTERN {
 friend class KNET;
private:
 double       P[MAXPATS][MAXNEURONSIN];
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2

Communication & cooperation: luzhenbo@yahoo.com.cn

```
 int         NumPatterns;
 int         Shuffle[MAXPATS];
 int         SizeVector;
public:
 PATTERN();
 int GetPatterns(char *);          //load pattern from file
 int GetRandPats(int,int);         //random patterns arg1=# of patterns, arg2=dimension
 double Query(int,int);            //returns P[arg1][arg2]
 double QueryR(int,int);           //returns P[Shuffle[arg1]][arg2]
 void ReShuffle(int N);
};

PATTERN::PATTERN(){
int i;
for (i=0; i<MAXPATS; i++)
  Shuffle[i]=i;
}

int PATTERN::GetPatterns(char *fname) {
  FILE *fp;
  int i,j;
  double x;
fp=fopen(fname,"r");
if (fp==NULL) return 0;                    // Test for failure.
fscanf(fp,"%d",&NumPatterns);
fscanf(fp,"%d",&SizeVector);
for (i=0; i<NumPatterns; i++) {            // For each vector
  for (j=0; j<SizeVector; j++) {           // create a pattern
    fscanf(fp,"%lg",&x);                   // consisting of all elements
    P[i][j]=x;
    } /* endfor */
  } /* endfor */
fclose(fp);
return 1;
}

int PATTERN::GetRandPats(int n1,int n2) {
  int i,j;
  double x;
NumPatterns=n1;
SizeVector=n2;
for (i=0; i<NumPatterns; i++) {            // For each vector
  for (j=0; j<SizeVector; j++) {           // creats a pattern
    x=(double)rand()/RAND_MAX;             // consisting of random elements
    P[i][j]=x;                             // between 0 and 1
    } /* endfor */
  } /* endfor */
return 1;
}
void PATTERN::ReShuffle(int N) {
int i,a1,a2,tmp;
for (i=0; i<N ;i++) {
  a1=Random(NumPatterns);
  a2=Random(NumPatterns);
  tmp=Shuffle[a1];
  Shuffle[a1]=Shuffle[a2];
  Shuffle[a2]=tmp;
  }
}
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
double PATTERN::Query(int pat,int j) {
return P[pat][j];
}

double PATTERN::QueryR(int pat,int j) {
return P[Shuffle[pat]][j];
}

class KNET {
private:
 double W[MAXNEURONSIN][MAXNEURONS];   // The weight matrix
 double Yout[MAXNEURONS];              // The output layer neurons
 double Yin[MAXNEURONSIN];             //The input layer neurons
 int    YinSize;                       //input layer dimensions
 int    YoutSize;                      //outlayer dimensions

 int    epoch;
 double eta;                           //The learning rate
 double delta_eta;                     //Amount to change l.r. each epoch
 int    StochFlg;                      //Present vectors in rand order if 1
 PATTERN *Pattern;

 int   LoadInLayer(int);               //pattern->input layer
 double EucNorm(int);                  //Calc Euclidean distance
 int    FindWinner();                  //get coords of winning neuron
 void   Train(int);
 void   AdaptParms();
public:
 KNET();
 void SetPattern(PATTERN *);
 void SetParms(int, double);
 void PrintWeights();
 void PrintWinner();
 void RunTrn();
 void Run();
};

KNET::KNET(){
StochFlg=0;
}

void KNET::SetPattern(PATTERN *p) {
  Pattern=p;
  YinSize=p->SizeVector;
}
void KNET::SetParms(int X, double LR){
 int i,k;
YoutSize=X;
eta=LR;
delta_eta=0.005;
for (i=0; i<YoutSize; i++) {
  for (k=0; k<YinSize; k++) {
    W[k][i]= (double)rand()/(10.0 * (double)RAND_MAX);
     } /* endfor */
  } /* endfor */
}

int KNET::LoadInLayer(int P){
 int i;
for (i=0; i<YinSize; i++){
  if (StochFlg){
    Yin[i]=Pattern->QueryR(P,i);
     }
   else {
    Yin[i]=Pattern->Query(P,i);
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
    }
  }
return 1;
}


void KNET::AdaptParms(){
eta=eta-delta_eta;
if (eta<ETAMIN)
  eta=ETAMIN;
printf(" New eta=%f\n",eta);
}

void KNET::PrintWeights() {
  int i,k;
for (i=0; i<YoutSize; i++) {
    for (k=0; k<YinSize; k++) {
      printf("W[%d][%d]=%f ",k,i,W[k][i]);
      } /* endfor */
    printf("\n");
  } /* endfor */
}

void KNET::RunTrn(){
int i,np;
int Winner;
epoch=0;
np=Pattern->NumPatterns;
while (epoch<=MAXEPOCHS){
  for (i=0; i<np; i++){
   LoadInLayer(i);
    Winner=FindWinner();
    Train(Winner);
    }
  if(5*(epoch/5)==epoch) {
   printf("Epoch=%d\n",epoch);
    PrintWeights();
    }
  epoch++;
  if (StochFlg)
   Pattern->ReShuffle(np);
  AdaptParms();
  }
}

void KNET::Train(int Winner){
  int k;
for (k=0; k<YinSize; k++){
  W[k][Winner]=W[k][Winner]+eta*(Yin[k]-W[k][Winner]);
  } /*endfor*/
}

int KNET::FindWinner(){
  int i;
  double d,best;
  int Winner;
best=1.0e99;
Winner=-1;
for (i=0; i<YoutSize; i++){
  d=EucNorm(i);
  if (d<best) {
    best=d;
    Winner=i;
    } // endif
  } // endfor
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
return Winner;
}

double KNET::EucNorm(int x){                    // Calc Euclidean norm of vector dif
int i;
double dist;
dist=0;
for (i=0; i< YinSize;i++){
  dist += (W[i][x]-Yin[i]) * (W[i][x]-Yin[i]);
  } /* endfor */
dist=sqrt(dist);
return dist;
}

//=================================================================
// GLOBAL OBJECTS
//=================================================================

PATTERN InPat;
KNET   net;

//=================================================================
// Main()
//=================================================================

main(int argc, char *argv[]) {
//srand(17);
if (argc>1) {
  InPat.GetPatterns(argv[1]);                   //Establish pattern
  net.SetPattern(&InPat);                       //Inform the feature map about the pattern
  net.SetParms(3, 0.500);                       //Init fm parms
  net.RunTrn();                                 //Run the FM w/ training enabled
  }
 else {
  printf("USAGE: KNET PATTERN_FILE");
  }
}
```

8.3.1 Kohonen 网络示例

下面的例子，说明了以上所述的基本 Kohonen 网的工作过程。该网络中，将应用图 8.7 中所示的模式。选用一个具有三个输出层神经元的一维网络。在第一次迭代之后($\eta=0.5$)，网络已获得了对聚类中心的相当好的逼近。第一次迭代之后网络的状态如下：

W[0][0]=5.470356　　W[1][0]=0.563049
W[0][1]=0.563464　　W[1][1]=5.470420
W[0][2]=5.409211　　W[1][2]=5.281787

经过 250 次迭代，η 已减小到 0.001。网络状态如下：

W[0][0]=5.507603　　W[1][0]=0.503805
W[0][1]=0.503805　　W[1][1]=5.507603
W[0][2]=5.503805　　W[1][2]=5.499810

经过 500 次迭代，η 保留为最小值 0.001。网络权值(代表由网络确定的聚类中心)如下：

W[0][0]=5.502463　　W[1][0]=0.501232
W[0][1]=0.501232　　W[1][1]=5.502463

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Copyright: luzhenbo@yahoo.com.cn

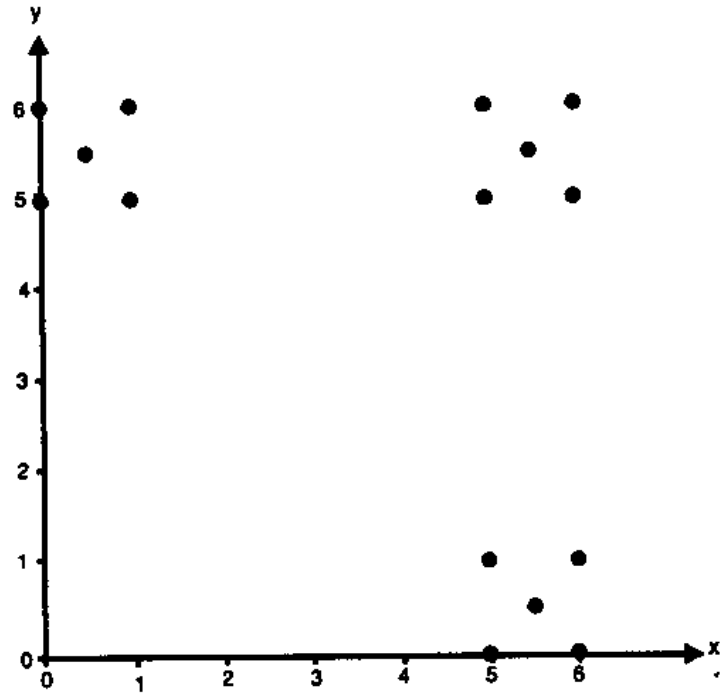


图 8.7 用 Kohonen 网络进行聚类的二维输入模式

W[0][2]=5.501232　　W[1][2]=5.499945

如下面的图 8.8 所示,可以看到网络已经很好地完成了输入模式聚类的任务。

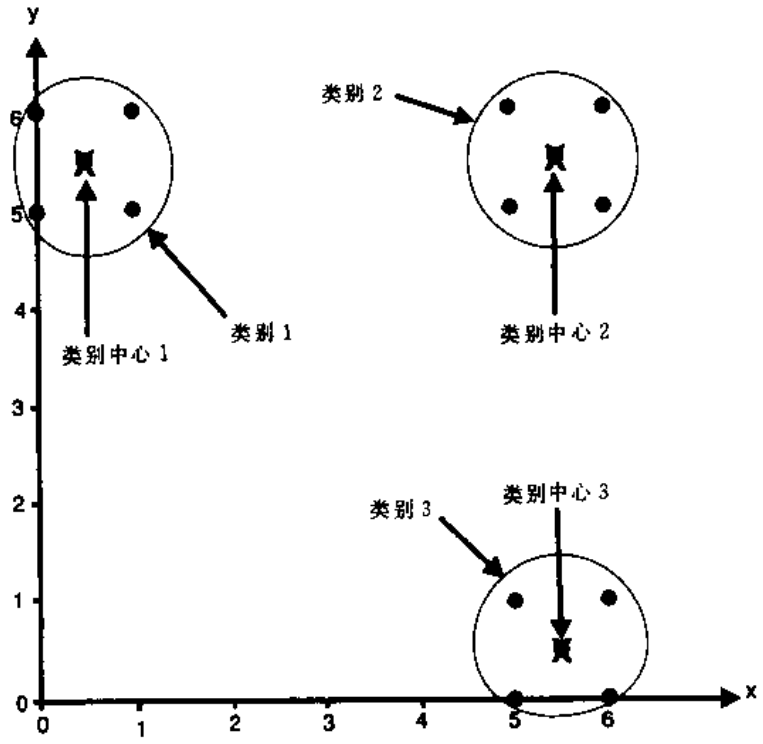


图 8.8 Kohonen 网络检测到的类别和聚类中心

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xidian.edu.cn

8.4 侧反馈规则

Kohonen SOFM 的工作过程中的侧反馈，比 8.3 节中介绍的要复杂一些。特别地，在 8.3 节中，并没有提供使聚类中心按特定方式有序化的实现方法。所以，虽然网络是自组织的，但是在输出层没有用任何相应的几何方法来组织特征。如果希望在输出层能够将特征进行聚集，以便于在阵列中相互靠近的位置可以找到类似的特征，那么必须改变侧反馈的方法来获得这一性能。

为完成这一任务，需在输出层建立反馈连接。图 8.9 和图 8.10 分别示出了一维阵列和二维阵列的情形。反馈的大小和类型（兴奋或抑制）用侧向权值表示，它是阵列内神经元之间几何距离的一个函数，它决定了哪种侧向连接将产生预期的结果。在这种情况下，庆幸的是我们可以从生物系统中得到启发。

大脑中视觉皮层中的几何特征映射与我们要寻找的相类似。因此，在视觉皮层内发现的小范围侧反馈，提供了可合并入我们的网络中的预期模型。公式(8-9)，经常被称为是墨西哥草帽函数，因而成为我们的侧反馈的模型。

$$\nabla^2 h = \left(\frac{r^2-\sigma^2}{\sigma^4}\right) e^{\frac{r^2}{2\sigma^2}} \tag{8-9}$$

公式(8-9)的图形，参见图 8.11。

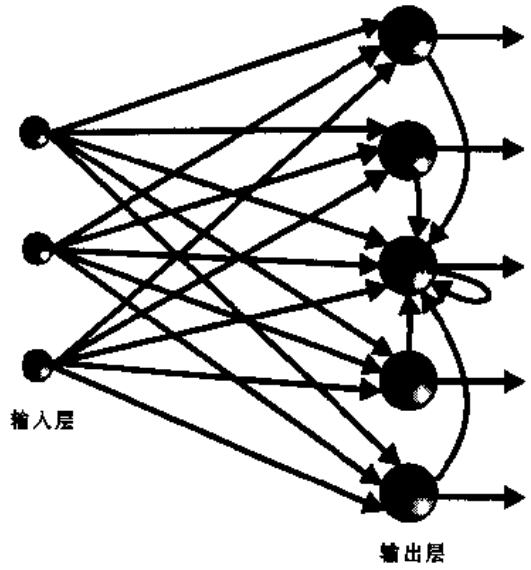


图 8.9 Kohonen 层为一维阵列的侧反馈情况

从图 8.11 可以看出，墨西哥草帽函数作为神经元之间的一个距离函数，包含三个明显的侧作用区域。当距离小于 R_0 时，侧反馈是兴奋的。在 R_0 和 R_1 之间的区域，有一抑制反馈的半阴影。超过 R_1 有一渐弱的兴奋。关于该模型与视觉皮层的关系的其它细节，详见 Kohonen〔1982〕的著作。

下面来看一下加上上述侧反馈后 Kohonen 层的操作。到输出层第 j 个神经元的网络输入，可表示成：

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

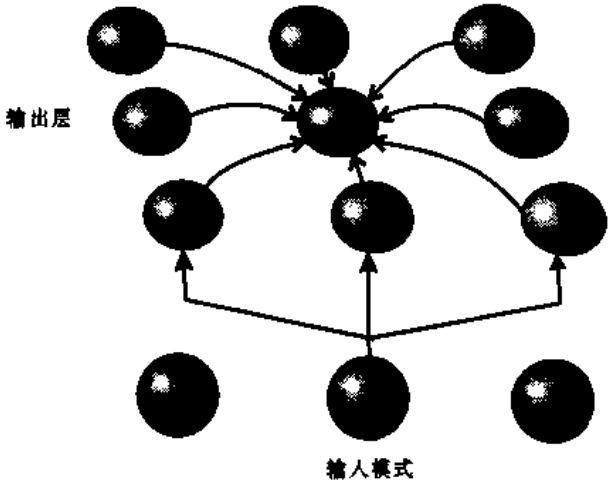


图 8.10 Kohonen 层为二维阵列的侧反馈情况

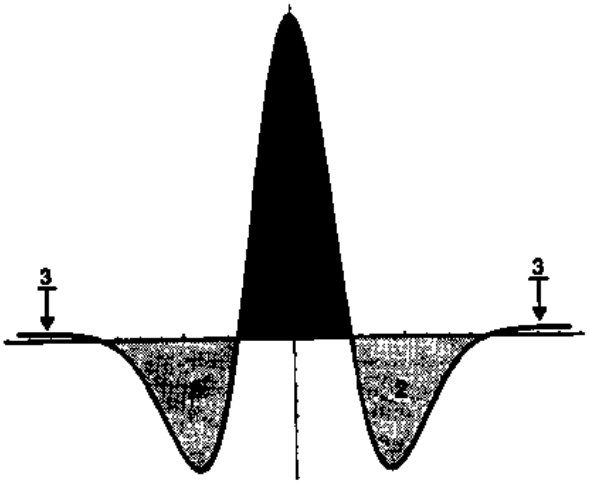


图 8.11 墨西哥草帽函数

$$net_j = \left(I_j + \sum_{K=-K}^{K} c_{j,j+k}\, y_{j+k}\right), j = 1,2,\cdots,n \tag{8-10}$$

这里，K 定义成包括所有激活的侧反馈的一个最大区域，I_j 定义如下：

$$I_j = \sum_{i=1}^{p} w_{ij} x_j \tag{8-11}$$

第 j 个神经元的输出可通过应用一个 net_j 的非线性函数 $\phi()$ 来获得，如下表示：$y_j = \phi(net_j)$。选择 ϕ 时要强制保证 $a > y_j \geqslant 0$，这里 a 是一任意常数。公式中的侧反馈包含在权值 c_{jk} 中。这些层内的权值是固定的，它意味着它们不经过训练，而是从公式(8-9)得到的或者至少它是一种合适的近似值。这样的侧向连接能够支持或刺激物理位置接近的神经元及抑制离得较远的神经元。通常，用墨西哥草帽函数的近似值来建立这些权值，就可以满足要求。图 8.12 给出了墨西哥草帽函数的近似情况。

需要用迭代的方法求解上面的公式(8-9)和(8-10)，在这种迭代中，输出层神经元实际上进入一种时间上的平衡状态。迭代公式如下：

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

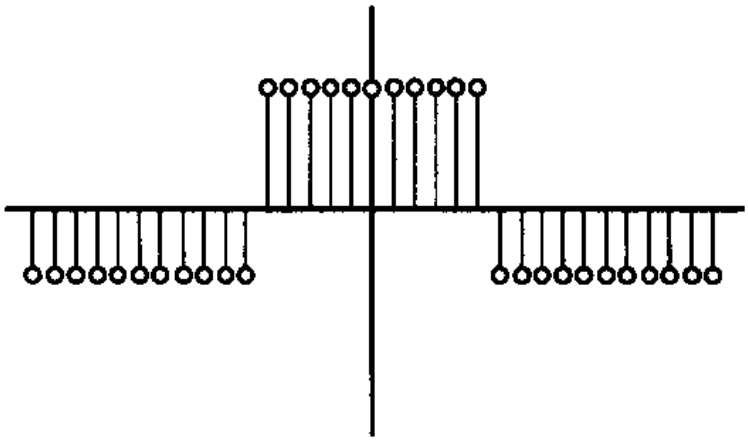


图 8.12 墨西哥草帽函数的近似

$$y_j(n+1)=\phi\Big(I_j+\beta\sum_{k=-K}^{K}c_{j,j+k}y_{j+k}(n)\Big),j=1,2,\cdots,N \tag{8-12}$$

其中，n 代表一离散的时间步长，β 是一控制收敛速率的常数。如果 $\phi()$ 和 β 选择适当的话，公式(8-11)将收敛，产生一个有界的空间区域或活性泡。在泡内，其响应是最大的，$y_j\to a$。在泡外，响应下降到零。泡的宽度由反馈连接控制。正向的反馈引起泡变宽，反向的反馈引起泡变小。泡是以具有最大 $y_j(0)$ 的神经元(由于 I_j 的刺激作用)为中心而形成的。这些结果表明，墨西哥草帽函数侧反馈的效果可用有效的计算方法来模拟。侧向权值被消去并且用一个区域代替，在这一区域中响应 y_j 被最大化，并且这一区域与活性泡相一致。侧反馈的调整可以通过简单地调整邻域的大小来体现：正向反馈越大，负向反馈就越小。这恰恰是在 8.5 节中学习 SOFM 时将要遇到的方法。

8.5 Kohonen 自组织特征映射

Kohonen SOFM 利用 Kohonen 结构和 8.3 节中所描述的 Kohonen 学习方法。通过讨论 8.4 节中所描述的侧反馈的实现方法，可以达到获得自组织特征映射算法的目的。这个目的就是把 n 维输入空间映射成(输出层神经元的)一维或二维阵列，构成一个存在有意义的拓扑序列的输出空间。

如前所述，每个输出层神经元相联系的权值矢量，被看作是神经元对输入矢量的响应的结果。将输入矢量 $\mathbf{x}$ 标记成：

$$\mathbf{x}=[x_1,x_2,\cdots,x_p]^T \tag{8-13}$$

与输出层神经元 j 相应的权值矢量 $\mathbf{w}_j$ 可写为：

$$\mathbf{W}_j=[w_{j1},w_{j2},\cdots,w_{jp}]^T \quad j=1,2,\cdots,N \tag{8-14}$$

获胜输出层神经元的确定，相当于选择权值矢量 $\mathbf{W}_j$ 与输入矢量 $\mathbf{x}$ 最为匹配的输出层神经元。回想一下我们先前的讨论，它可用两种方法来实现。我们可选刺激量 $I_j=\mathbf{W}_j^T\mathbf{x}$ 为最大的输出层神经元为获胜神经元。另外，我们可以选出权值矢量距离输入矢量有最小欧氏范数值的输出层神经元作为获胜神经元。如果用 $i(\mathbf{x})$ 来指定为获胜神经元的标号，则后一种方法

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2

Communication & cooperation: luzhenbo@yahoo.com.cn

可表示成：

$$i(\mathbf{x})=k \quad \text{当} \|\mathbf{W}_k-\mathbf{x}\|<\|\mathbf{W}_j-\mathbf{x}\| \quad j=1,2,\cdots,n \tag{8-15}$$

下面说明适度保持侧反馈的方法。首先介绍一个函数，该函数定义了围绕获胜神经元邻近区域(与活性泡相应)的大小。这个函数 $\Lambda_{i(x)}(n)$是一个离散时间(也就是迭代)的函数。使用这一函数意味着侧反馈的大小(也就是活性泡的宽度)可在整个训练网络的过程中变化。邻域越大意味着正向反馈越多，活性泡越大，训练区域越大。正是通过网络的早期训练期间邻域函数的较大值，使得网络达到拓扑结构有序化。随后邻域的缩小使类别更小，以致于类别分得更细。典型地，可以将邻域函数是很方便地表示成如图 8.13 所示的矩形阵列。注意，半径为零时，仅仅包含获胜神经元；半径为 1 时，包括 8 个近邻神经元。其它的形式，比如图 8.14 所示的六边形阵列，也是可能的。

可以用邻域函数来修改公式(8-15)所表示的学习过程：

$$\mathbf{W}_j(n+1)=\begin{cases}\mathbf{W}_j(n)+\eta(n)[\mathbf{x}-\mathbf{W}_j(n)] & j\in\Lambda_{i(\mathbf{x})}(n)\\ \mathbf{W}_j(n) & \text{其它}\end{cases} \tag{8-16}$$

注意到由公式(8-15)修改后的训练方法，可导致或者促进邻域范围内的神经元包含相似的权值矢量。

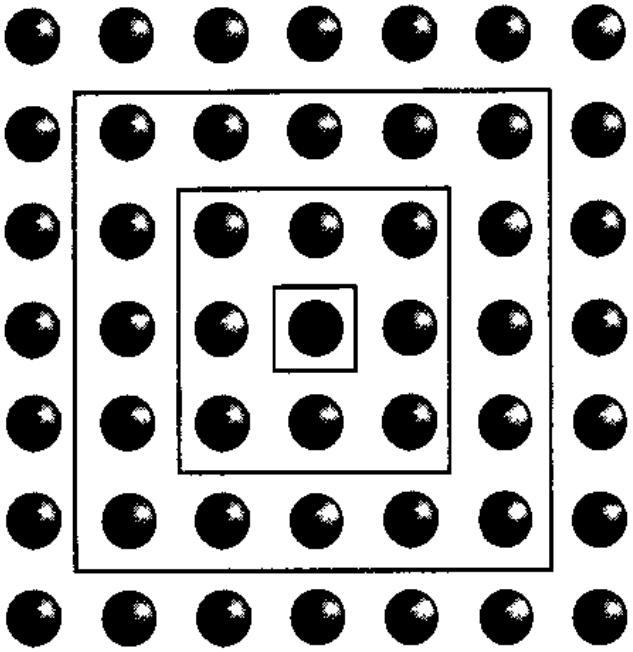


图 8.13 矩形阵列邻域

Kohonen 自组织特征映射算法，可用如下步骤表示：

步骤 1. 初始化：

初始化权值矢量 $\mathbf{W}_j(0)$时可选随机值。初始值通常选择小一点。初始化学习率 $\eta(0)$和邻域函数 $\Lambda_{i(\mathbf{x})}(0)$。它们初始化时应尽量取大一些(后面将详细论述如何选择这些初始值)。

步骤 2. 对于样本中每个矢量 x 执行步骤 2a、2b 和 2c。

步骤 2a. 将感觉刺激矢量 x 送入到网络的输入层上去。

步骤 2b. 相似匹配：

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

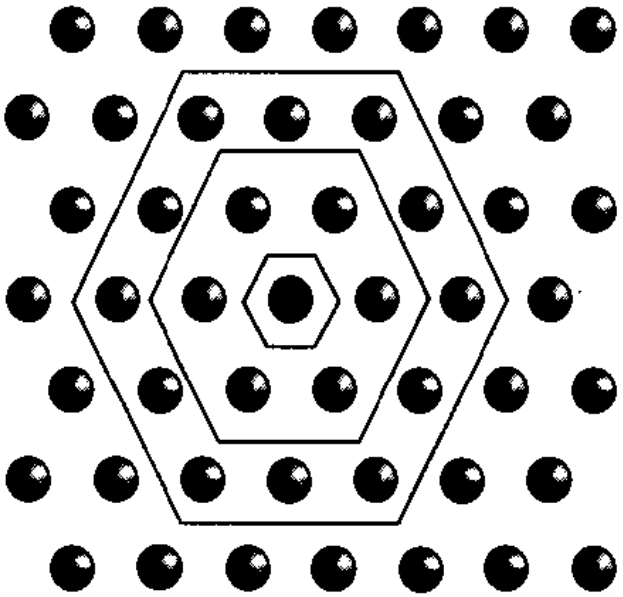


图 8.14 六边形阵列邻域

选择权值矢量最匹配 x 的神经元作为获胜神经元。运用欧氏法则，获胜神经元的标号为：

$$i(\mathbf{x})=k \quad \text{当} \ \|\mathbf{W}_k-\mathbf{x}\|<\|\mathbf{W}_j-\mathbf{x}\| \quad j=1,2,\cdots,n \tag{8-17}$$

步骤 2c. 训练：

训练权值矢量，使得在活性泡范围内的神经元朝着输入矢量方向移动：

$$\mathbf{W}_j(n+1)=\begin{cases}\mathbf{W}_j(n)+\eta(n)[\mathbf{x}-\mathbf{W}_j(n)] & j\in\Lambda_{i(\mathbf{x})}(n)\\ \mathbf{w}_j(n) & \text{其它}\end{cases} \tag{8-18}$$

步骤 3. 更新学习率 $\eta(n)$。学习率的线性减小将产生令人满意的结果。

步骤 4. 减小邻域函数 $\Lambda_{i(\mathbf{x})}(n)$。

步骤 5. 检查结束条件：

当特征映射不再发生明显的变化时退出，否则就转入步骤 2。

自组织特征映射网络中的学习，包括两个主要的阶段：有序化阶段和收敛阶段。有序化阶段在前。在这一阶段期间产生权值矢量 $\mathbf{W}_j$ 的拓扑有序化(总体)，η 值相对保留大一些。初始时，η 可选得接近 1.0；在有序化阶段，η 不要降到 0.1 以下。另外，在此阶段，邻域函数开始也相对地大一些，经常包括网中所有神经元。在整个算法的过程中，邻域函数将减小，被限制为几个神经元(或者甚至可能仅仅是获胜神经元一个)。有序化阶段可能持续大约一千左右次选代，比收敛阶段短得多。在收敛阶段，输出神经元寻找精确的样本权值矢量值。直到结束为止，在算法的收敛阶段中，总是希望 η 和邻域函数的值小一些。这时，η 的典型值将维持在 0.01 或更小。

下面的清单 8.3，提供了自组织特征映射的完整的源程序代码。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication E-mail: luzhenbo@yahoo.com.cn

清单 8.3

```
/*****************************************************
* Kohonen's self organizing feature map              *
*****************************************************/
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

#define  MAXPATS     100
#define  MAXNEURONSIN 10
#define  MAXNEURONSX  15
#define  MAXNEURONSY  15

#define  MAXEPOCHS   2000
#define  ETAMIN      .005

unsigned int Random(int N) {
 unsigned int j;
j= (N*rand())/RAND_MAX;
if (j>=N) j=N;
return j;
}
/*****************************************************
* Pattern class definition                           *
*****************************************************/
class PATTERN {
 friend class SOFM;
private:
 double       P[MAXPATS][MAXNEURONSIN];
 int          NumPatterns;
 int          Shuffle[MAXPATS];
 int          SizeVector;
public:
 PATTERN();
 int GetPatterns(char *);          //load pattern form file
 int GetRandPats(int,int);         //random patterns arg1=# of patterns, arg2=dimension
 double Query(int,int);            //returns P[arg1][arg2]
 double QueryR(int,int);           //returns P[Shuffle[arg1]][arg2]
 void ReShuffle(int N);
};

/*****************************************************
* Pattern class method implementation                *
*****************************************************/

//*****************************************************
// Pattern class constructor
//*****************************************************

PATTERN::PATTERN(){
int i;
for (i=0; i<MAXPATS; i++)
  Shuffle[i]=i;
}

//*****************************************************
// GetPatterns
//    Read pattern data from file
//*****************************************************

int PATTERN::GetPatterns(char *fname) {
 FILE *fp;
 int i,j;
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xaut.edu.cn


```
 double x;
fp=fopen(fname,"r");
if (fp==NULL) return 0;                    // Test for failure.
fscanf(fp,"%d",&NumPatterns);
fscanf(fp,"%d",&SizeVector);
for (i=0; i<NumPatterns; i++) {            // For each vector
  for (j=0; j<SizeVector; j++) {           // create a pattern
    fscanf(fp,"%lg",&x);                   // consisting of all elements
    P[i][j]=x;
    } /* endfor */
  } /* endfor */
fclose(fp);
return 1;
}
//*********************************************************
// GetRandPats
//    Creates n1 random pattern vectors
//    with n2 elements
//*********************************************************

int PATTERN::GetRandPats(int n1,int n2) {
  int i,j;
  double x;
NumPatterns=n1;
SizeVector=n2;
for (i=0; i<NumPatterns; i++) {            // For each vector
  for (j=0; j<SizeVector; j++) {           // create a pattern
    x=(double)rand()/RAND_MAX;             // consisting of random elements
    P[i][j]=x;                             // between 0 and 1
    } /* endfor */
  } /* endfor */
return 1;
}


//*********************************************************
// Reshuffle
//    Randomize an array consisting of all patterns
//*********************************************************

void PATTERN::ReShuffle(int N) {
int i,a1,a2,tmp;
for (i=0; i<N ;i++) {
  a1=Random(NumPatterns);
  a2=Random(NumPatterns);
  tmp=Shuffle[a1];
  Shuffle[a1]=Shuffle[a2];
  Shuffle[a2]=tmp;
  }
}

//*********************************************************
// Query
//    Returns the jth element of the
//    specified pattern vector
//*********************************************************

double PATTERN::Query(int pat,int j) {
return P[pat][j];
}


//*********************************************************
// QueryR
//    Returns the jth element of the pattern whose
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication Station: luzhenbo@yahoo.com.cn


```
//    index is obtained from the shuffle array.
//    (Used in conjunction with reshuffle to train
//      patterns in random order
//*******************************************************
double PATTERN::QueryR(int pat,int j) {
return P[Shuffle[pat]][j];
}

struct iPair {
 int x,y;
};

/*******************************************************
* SOFM class definition                                *
*******************************************************/

class SOFM {
private:
 double W[MAXNEURONSIN][MAXNEURONSX][MAXNEURONSY]; // The weight
matrix
 double Yout[MAXNEURONSX][MAXNEURONSY];          // The output layer neurons
 double Yin[MAXNEURONSIN];                       //The input layer neurons
 int    Lattice;                                 //Square Vs triangular lattice
 int    YinSize;                                 //input layer dimensions
 iPair  YoutSize;                                //outlayer dimensions

 int    R;                                       //update neighborhood radius
 int    MaxEpoch;
 int    epoch;
 double eta;                                     //The learning rate
 double delta_eta;                       //Amount to change l.r. each epoch
 double Erosion;                         //Urban decay metric..Neighborhoods shrink
 int    StochFlg;                        //Present vectors in rand order if 1
 PATTERN *Pattern;

 int   LoadInLayer(int);                 //pattern->input layer
 double EucNorm(int, int);               //Calc Euclidean distance
 iPair  FindWinner();                    //get coords of winning neuron
 void   Train(iPair);
 void   AdaptParms();
public:
 SOFM();
 void SetPattern(PATTERN *);
 void SetParms(int, int, double);
 void PrintWeights();
 void PrintWinner();
 void RunTrn();
 void Run();
};

/*******************************************************
* Pattern method implementations                       *
*******************************************************/

//*******************************************************
// SOFM class constructor
//*******************************************************

SOFM::SOFM(){
StochFlg=1;
Erosion=0;
}

//*******************************************************
// SetPattern
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
//  Establish linkage w/ the pattern
//*********************************************************

void SOFM::SetPattern(PATTERN *p) {
  Pattern=p;
  YinSize=p->SizeVector;
}

//*********************************************************
// SetParms
//   Establish needed parameters
//*********************************************************

void SOFM::SetParms(int X, int Y, double LR){
 int ix,iy,k;
YoutSize.x=X;
YoutSize.y=Y;
R=(X+Y)/4;
eta=LR;
delta_eta=0.005;
for (ix=0; ix<X; ix++) {
  for (iy=0; iy<Y; iy++) {
    for (k=0; k<YinSize; k++) {
      W[k][ix][iy]= (double)rand()/(10.0 * (double)RAND_MAX);
       } /* endfor */
    } /* endfor */
  } /* endfor */
}

//*********************************************************
// LoadInLayer
//   Initialize input layer w/ the designated pattern
//*********************************************************

int SOFM::LoadInLayer(int P){
  int i;
for (i=0; i<YinSize; i++){
  if (StochFlg){
    Yin[i]=Pattern->QueryR(P,i);
     }
   else {
    Yin[i]=Pattern->Query(P,i);
     }
  }
 return 1;
 }

 //*********************************************************
 // AdaptParms
 //   Adjust learning rate and neighborhood size //
 *********************************************************

 void SOFM::AdaptParms(){
 Erosion += .01;
 if (Erosion>=1.0) {
   Erosion=0.0;
   if (R>0)
     R--;
  printf("New neighborhood. Radius=%d", R);
   } /* endif */
 if (epoch<500) {                    //Reduce learning rate more slowly over 1st 1k epochs
  eta=eta-delta_eta/10.0;
   }
  else {
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication or cooperation: luzhenbo@yahoo.com.cn


```
  eta=eta-delta_eta;
  } /* endif */
if (eta<ETAMIN)
  eta=ETAMIN;
printf(" New eta=%f\n",eta);
}

//*************************************************************
// PrintWeights
//   Display the weight matrix
//*************************************************************

void SOFM::PrintWeights() {
  int ix,iy,k;
for (ix=0; ix<YoutSize.x; ix++) {
  for (iy=0; iy<YoutSize.y; iy++) {
     for (k=0; k<YinSize; k++) {
       printf("W[%d][%d][%d]=%f  ",k,ix,iy,W[k][ix][iy]);
        } /* endfor */
        printf("\n");
     } /* endfor */
  } /* endfor */
}


//*************************************************************
// RunTrn
//   Run the network w/ training enabled.
//*************************************************************

void SOFM::RunTrn(){
int i,np;
iPair Winner;
epoch=0;
np=Pattern->NumPatterns;
while (epoch<=MAXEPOCHS){
  for (i=0; i<np; i++){
   LoadInLayer(i);
   Winner=FindWinner();
   Train(Winner);
   }
  if(20*(epoch/20)==epoch) {
   printf("Epoch=%d\n",epoch);
    PrintWeights();
    }
  epoch++;
  if (StochFlg)
   Pattern->ReShuffle(np);
  AdaptParms();
  }
}
//*************************************************************
// Train
//   Update the weights of the winning neuron and
//   update the weights of neurons within the winning
//   neurons neighborhood
//*************************************************************
void SOFM::Train(struct iPair Winner){
 int ix,iy,k;
for (ix=Winner.x-R; ix<=Winner.x+R; ix++){
 if ((ix>=0) && (ix<YoutSize.x)){
   for (iy=Winner.y-R; iy<=Winner.y+R; iy++){
     if ((iy>=0) && iy<YoutSize.y) {
        for (k=0; k<YInSize; k++){
         W[k][ix][iy]=W[k][ix][iy]+eta*(Yin[k]-W[k][ix][iy]);
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
          } /*endfor*/
        } /*endif*/
      } /*endfor*/
    } /*endif*/
  } /*endfor*/
}
//*********************************************************
// FindWinner
//   Discover winning neuron based on min Euclidean
//   distance
//*********************************************************
iPair SOFM::FindWinner(){
 int ix,iy;
 double d,best;
 iPair Winner;
best=1.0e99;
Winner.x=-1;
Winner.y=-1;
for (ix=0; ix<YoutSize.x; ix++){
 for (iy=0; iy<YoutSize.y; iy++){
  d=EucNorm(ix,iy);
  if (d<best) {
    best=d;
    Winner.x=ix;
    Winner.y=iy;
    } // endif
  } // endfor
 } // endfor
return Winner;
}


//*********************************************************
// EucNorm
//   Calculate Euclidean distance between a pattern and
//   the neuron at x,y in the output layer lattice
//*********************************************************

double SOFM::EucNorm(int x, int y){              // Calc Euclidean norm of vector dif
int i;
double dist;
dist=0;
for (i=0; i< YinSize;i++){
  dist += (W[i][x][y]-Yin[i]) * (W[i][x][y]-Yin[i]);
  } /* endfor */
dist=sqrt(dist);
return dist;
}
//=================================================================
// GLOBAL OBJECTS
//=================================================================

PATTERN InPat;
SOFM   FMap;


//*********************************************************
// main
//*********************************************************

main(int argc, char *argv[]) {
//srand(17);
if (argc>1) {
  InPat.GetPatterns(argv[1]);                    //Establish pattern
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication E-mail address: luzhenbo@yahoo.com.cn


```
FMap.SetPattern(&InPat);                    //Inform the feature map about the pattern
FMap.SetParms(5,5,0.900);                   //Init fm parms
FMap.RunTrn();                              //Run the FM w/ training enabled
 }
else {
 printf("USAGE: SOFM PATTERN_FILE");
 }
}
```

8.5.1 SOFM 举例

下面说明 SOFM 的操作。图 8.15 表示了应用于 SOFM 的输入模式。与图 8.15 相应的输入矢量如下：

$$\{(5.0,5.0),(6.0,6.0),(5.0,6.0),(6.0,5.0),(5.5,5.5)$$
$$(5.0,0.0),(5.0,1.0),(6.0,0.0),(6.0,1.0)(5.5,0.5)$$
$$(0.0,5.0)(1.0,5.0),(0.0,6.0),(1.0,6.0),(0.5,5.5)\}$$

对于这一示例所选的 SOFM 网络拓扑是一个二维的 5×5 神经元方阵。初始时，邻域由以获胜神经元为圆心，以 3 为半径的区域内所有神经元组成。半径随着训练的进行面减小。初始学习率是 $\eta=0.9$。学习率也随着学习的进行而减小。

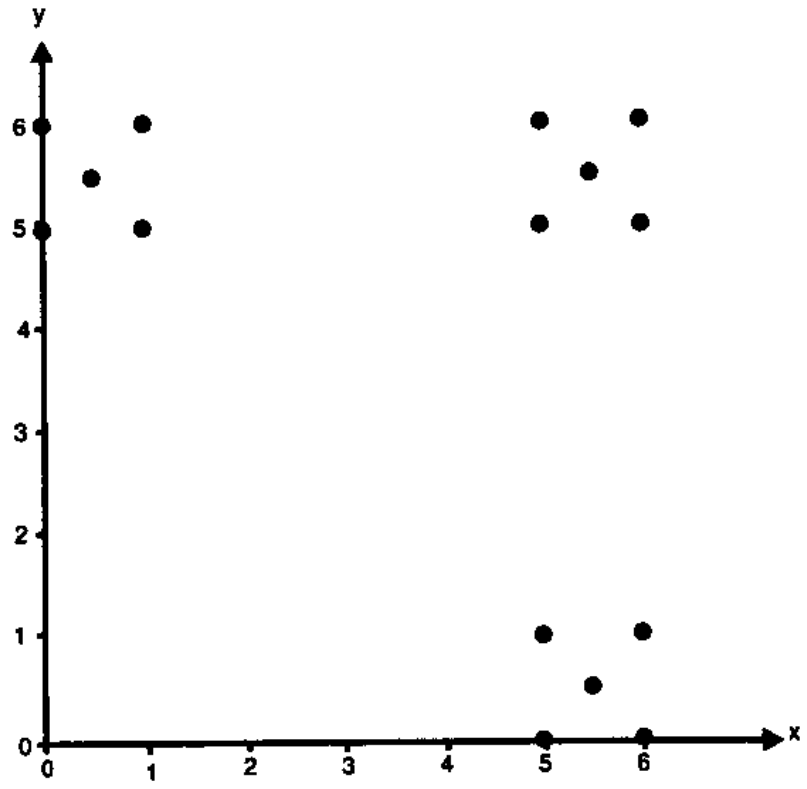


图 8.15 SOFM 举例中的输入模式

经过一次迭代之后，权值矩阵如下：

W[0][0][0]=0.995085 W[I][0][0]=5.009985
W[0][0][1]=1.004001 W[1][0][1]=4.995960
W[0][0][2]=1.004001 W[1][0][2]=4.995960
W[0][0][3]=1.004001 W[1][0][3]=4.995960

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@163.com

W[0][0][4]=1.003950 W[1][0][4]=4.995900
W[0][1][0]=0.540955 W[1][1][0]=5.548956
W[0][1][1]=0.540955 W[1][1][1]=5.548955
W[0][1][2]=0.540955 W[1][1][2]=5.548955
W[0][1][3]=0.540955 W[1][1][3]=5.548955
W[0][1][4]=1.004900 W[1][1][4]=5.004095
W[0][2][0]=0.540955 W[1][2][0]=5.548956
W[0][2][1]=0.540955 W[1][2][1]=5.548955
W[0][2][2]=0.540955 W[1][2][2]=5.548955
W[0][2][3]=0.540955 W[1][2][3]=5.548955
W[0][2][4]=1.004900 W[1][2][4]=5.004095
W[0][3][0]=0.540955 W[1][3][0]=5.548956
W[0][3][1]=0.540955 W[1][3][1]=5.548955
W[0][3][2]=0.540955 W[1][3][2]=5.548955
W[0][3][3]=0.540955 W[1][3][3]=5.548955
W[0][3][4]=1.004900 W[1][3][4]=5.004095
W[0][4][0]=0.540960 W[1][4][0]=5.548964
W[0][4][1]=0.540960 W[1][4][1]=5.548960
W[0][4][2]=0.540960 W[1][4][2]=5.548960
W[0][4][3]=0.540960 W[1][4][3]=5.548960
W[0][4][4]=1.049491 W[1][4][4]=5.045401

其中 W[x][y][i]代表输出层阵列中位置为(x,y)的输出层神经元上的权值的第 i 个分量。

经 100 次迭代之后,学习率已减小至 0.85,并且邻域半径也减小至 2。权值矩阵如下:

W[0][0][0]=5.083817 W[1][0][0]=5.066834
W[0][0][1]=5.862818 W[1][0][1]=1.609674
W[0][0][2]=5.552790 W[1][0][2]=0.650543
W[0][0][3]=5.555808 W[1][0][3]=0.554776
W[0][0][4]=5.555326 W[1][0][4]=0.552985
W[0][1][0]=5.083817 W[1][1][0]=5.066836
W[0][1][1]=5.775423 W[1][1][1]=1.601451
W[0][1][2]=5.541011 W[1][1][2]=0.662873
W[0][1][3]=5.473923 W[1][1][3]=0.650328
W[0][1][4]=5.473912 W[1][1][4]=0.650287
W[0][2][0]=5.083004 W[1][2][0]=5.067118
W[0][2][1]=5.775368 W[1][2][1]=1.601509
W[0][2][2]=5.537867 W[1][2][2]=0.666115
W[0][2][3]=5.454642 W[1][2][3]=0.672364
W[0][2][4]=5.454642 W[1][2][4]=0.672363
W[0][3][0]=5.064609 w[1][3][0]=5.086207

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication E-mail: luzhenbo@yahoo.com.cn

W[0][3][1]=4.499728 W[1][3][1]=5.012878
W[0][3][2]=4.348711 W[1][3][2]=5.065260
W[0][3][3]=0.555089 W[1][3][3]=5.428316
W[0][3][4]=0.554743 W[1][3][4]=5.425703
W[0][4][0]=5.064617 W[1][4][0]=5.086217
W[0][4][1]=4.499690 W[1][4][1]=5.012925
W[0][4][2]=4.348711 W[1][4][2]=5.065260
W[0][4][3]=0.555051 w[1][4][3]=5.428363
W[0][4][4]=0.553039 W[1][4][4]=5.427828

经 200 次迭代之后,学习率已减小至 0.80,并且邻域半径也减小至 1。权值矩阵如下:

W[0][0][0]=0.176284 W[1][0][0]=5.022428
W[0][0][1]=0.209436 W[1][0][1]=4.984109
W[0][0][2]=4.957143 W[1][0][2]=0.241577
W[0][0][3]=5.118544 W[1][0][3]=0.080189
W[0][0][4]=5.118544 W[1][0][4]=0.080189
W[0][1][0]=0.323467 W[1][1][0]=4.876737
W[0][1][1]=0.329895 W[1][1][1]=4.868832
W[0][1][2]=4.953037 W[1][1][2]=0.246759
W[0][1][3]=5.823735 W[1][1][3]=0.816265
W[0][1][4]=5.823737 W[1][1][4]=0.816262
W[0][2][0]=4.033593 W[1][2][0]=1.992169
W[0][2][1]=5.767417 W[1][2][1]=4.200818
W[0][2][2]=5.887439 W[1][2][2]=4.150783
W[0][2][3]=5.983738 W[1][2][3]=1.654697
W[0][2][4]=5.918693 W[1][2][4]=1.073483
W[0][3][0]=0.321344 W[1][3][0]=5.839687
W[0][3][1]=1.132862 W[1][3][1]=5.801589
W[0][3][2]=5.560012 W[1][3][2]=5.427188
W[0][3][3]=5.592318 W[1][3][3]=4.766708
W[0][3][4]=5.567962 W[1][3][4]=4.799698
W[0][4][0]=0.204840 W[1][4][0]=5.992078
W[0][4][1]=1.129061 W[1][4][1]=5.807937
W[0][4][2]=5.561553 W[1][4][2]=5.433309
W[0][4][3]=5.561591 W[1][4][3]=5.433542
W[0][4][4]=5.439809 W[1][4][4]=5.598542

经 300 次迭代之后,学习率已减小至 0.75,并且邻域半径减小至 0。因而从这一点开始,只有获胜的神经元被训练。经过 300 次迭代后的权值矩阵如下:

W[0][0][0]=0.152532 W[1][0][0]=5.141537
W[0][0][1]=0.919860 W[1][0][1]=5.140855
W[0][0][2]=5.210362 W[1][0][2]=5.038421

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

W[0][0][3]=5.097558 W[1][0][3]=5.856369
W[0][0][4]=5.847307 W[1][0][4]=5.858987
W[0][1][0]=0.131883 W[1][1][0]=5.879134
W[0][1][1]=0.919861 W[1][1][1]=5.890854
W[0][1][2]=5.052640 W[1][1][2]=5.008963
W[0][1][3]=5.847556 W[1][1][3]=5.106356
W[0][1][4]=5.472323 W[1][1][4]=5.483821
W[0][2][0]=1.713121 W[1][2][0]=4.239573
W[0][2][1]=2.114362 W[1][2][1]=4.810238
W[0][2][2]=5.193171 W[1][2][2]=4.807382
W[0][2][3]=5.243454 W[1][2][3]=4.807361
W[0][2][4]=5.225576 W[1][2][4]=4.228176
W[0][3][0]=4.745697 W[1][3][0]=1.301179
W[0][3][1]=5.002715 W[1][3][1]=0.953236
W[0][3][2]=5.387721 W[1][3][2]=0.565731
W[0][3][3]=5.577927 W[1][3][3]=0.563596
W[0][3][4]=5.905813 W[1][3][4]=0.894870
W[0][4][0]=4.751555 W[1][4][0]=1.295121
W[0][4][1]=5.008562 W[1][4][1]=0.802011
W[0][4][2]=5.096930 W[1][4][2]=0.141378
W[0][4][3]=5.519473 W[1][4][3]=0.515830
W[0][4][4]=5.905812 W[1][4][4]=0.143951

因为邻域半径已减小至零,所以已进入了收敛阶段。在 2000 次迭代之后,学习率已到了其最小值 0.005,网络状态如下:

W[0][0][0]=0.000000 W[1][0][0]=5.000000
W[0][0][1]=0.750000 W[1][0][1]=5.250000
W[0][0][2]=5.210362 W[1][0][2]=5.038421
W[0][0][3]=5.000000 W[1][0][3]=6.000000
W[0][0][4]=6.000000 W[1][0][4]=6.000000
W[0][1][0]=0.000000 W[1][1][0]=6.000000
W[0][1][1]=1.000000 W[1][1][1]=6.000000
W[0][1][2]=5.000000 W[1][1][2]=5.000000
W[0][1][3]=6.000000 W[1][1][3]=5.000000
W[0][1][4]=5.500000 W[1][1][4]=5.500000
W[0][2][0]=1.320044 W[1][2][0]=3.266798
W[0][2][1]=2.114362 W[1][2][1]=4.810238
W[0][2][2]=5.193171 W[1][2][2]=4.807382
W[0][2][3]=5.243454 W[1][2][3]=4.807361
W[0][2][4]=5.225576 W[1][2][4]=4.228176
W[0][3][0]=4.745697 W[1][3][0]=1.301179

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and Cooperation: luzhenbo@yahoo.com.cn

W[0][3][1]=5.000000 W[1][3][1]=1.000000
W[0][3][2]=5.387721 W[1][3][2]=0.565731
W[0][3][3]=5.577927 W[1][3][3]=0.563596
W[0][3][4]=6.000000 W[1][3][4]=1.000000
W[0][4][0]=4.751555 W[1][4][0]=1.295121
W[0][4][1]=5.008562 W[1][4][1]=0.802011
W[0][4][2]=5.000000 W[1][4][2]=0.000000
W[0][4][3]=5.500000 W[1][4][3]=0.500000
W[0][4][4]=6.000000 W[1][4][4]=0.000000

图 8.16 表示了由特征映射形成的类别相应的权值矢量的分布情况。为了强调说明权值矩阵在几何意义上已经与聚类中心相对应,图 8.17 示出了按照相应的权值矢量所代表的聚类中心而划分的权值矩阵。

8.6 学习矢量量化

学习矢量量化(LVQ)[Kohonen,1989,1990a]是一种前而已研究过的 Kohonen 网络方法的有监督学习的扩展形式。它允许对输入将被分到哪一类进行指定。对于为训练集合指定的种类预先知晓并且也属于训练集合的一部分。LVQ 网络结构除了输出层的每个神经元被指定为属于几个类别之一外,与我们已熟悉的 SOFM 是完全相同的。图 8.18 说明了这一点。一般地,每个类别将分得几个输出神经元。像以前一样,对于一给定输出单元的权值矢量,它

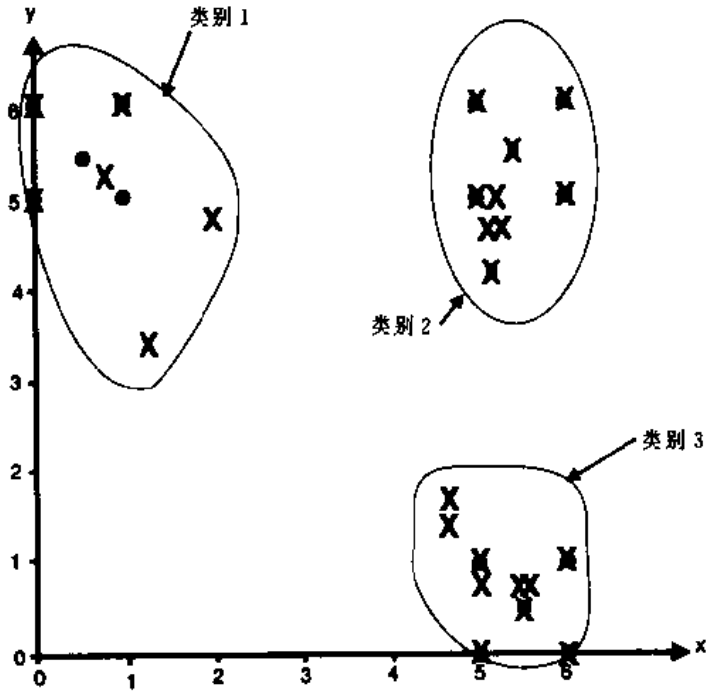


图 8.16 由 SOFM 检测出的类别及其中心

代表在输出端的响应最大的输入矢量的样本。在 LVQ 的方法中,这样的权值矢量有时被看作是一参考或码书矢量。当一种输入模式 x 输入到网络时,有最近(欧氏范数)的权值矢量的

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/u/1212121183
Communication & Cooperation: luzhenbo@yahoo.com.cn

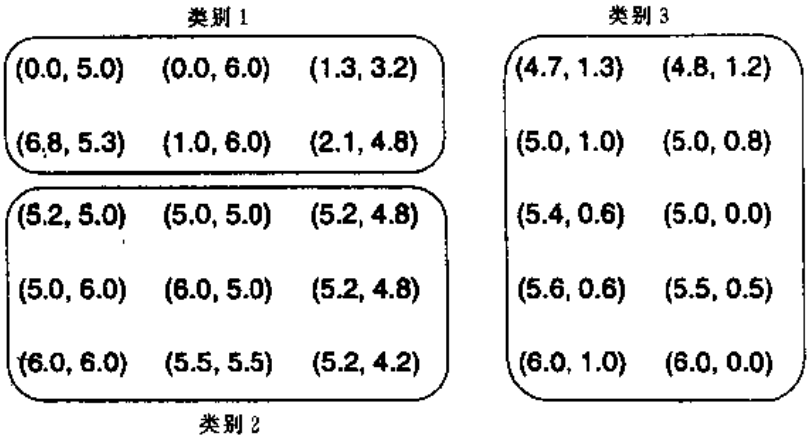


图 8.17 矢量矩阵所表明的二维阵列 SOFM 的类别情况

神经元被选定为获胜者。训练过程利用类似于较早期的 Kohonen 学习的一种规则，只有获胜的神经元被修改。公式(8-19)表明其训练调整为：

$$\begin{aligned} \mathbf{W}_{ij}^{新} &= \mathbf{W}_{ij}^{旧} + \eta(x_i - \mathbf{W}_{ij}^{旧}) \quad \text{分类正确时} \\ \mathbf{W}_{ij}^{新} &= \mathbf{W}_{ij}^{旧} - \eta(x_i - \mathbf{W}_{ij}^{旧}) \quad \text{分类不正确时} \end{aligned} \tag{8-19}$$

很明显，这一训练方法可获得一获胜神经元。如果它属于正确的类别，那么它将朝着输入矢量的方向移动；反之，如果获胜神经元不属于正确的类别，它将被迫向远离输入矢量的方向移动。这一性能如图 8.19 所示。在提出 LVQ 算法以前，我们先介绍几个必要的术语。令

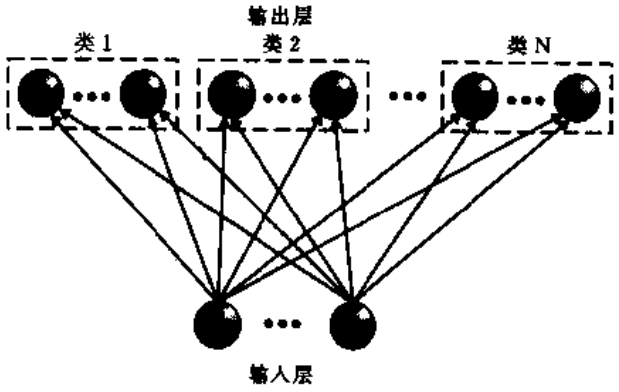


图 8.18 学习矢量量化网络结构

$\mathbf{x}^{(p)}$代表第 p 个训练模式矢量，$T^{(p)}$代表 $\mathbf{x}^{(p)}$所属的类别，最后令 C_j 为第 j 个输出神经元代表的类别。现在，LVQ 算法可用下面的步骤来表达：

步骤 1. 初始化：

初始化权值矢量。权值矢量可随机地初始化。然面，随后我们将讨论还有一些其它的选择。同时，初始化学习率。

步骤 2. 对训练集合中的每个矢量 $\mathbf{x}^{(p)}$执行步 2a 和 2b。

步骤 2a. 用如下方法寻找获胜神经元 k。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication E-mail: luzhenbo@yahoo.com.cn

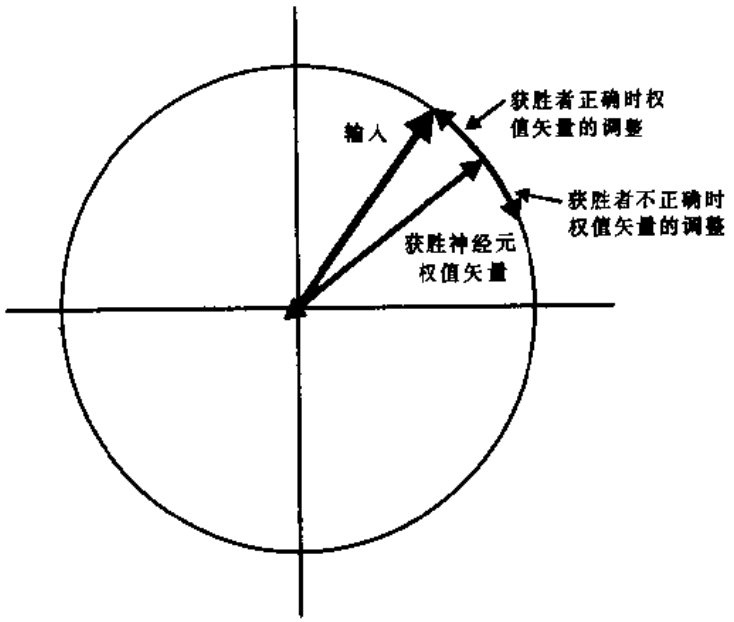


图 8.19 LVQ 网络的训练

$$i(\mathbf{x}^{(p)})=k \quad \text{其中,} \| \mathbf{W}_k-\mathbf{x}^{(p)} \| < \| \mathbf{W}_j-\mathbf{x}^{(p)} \| \quad j=1,2,\cdots,n \tag{8-20}$$

步骤 2b. 修改权值 $\mathbf{W}_k$ 如下:

$$\mathbf{W}_k^{\text{新}}=\begin{cases}\mathbf{W}_k^{\text{旧}}+\eta(\mathbf{x}^{(p)}-\mathbf{W}_k^{\text{旧}}) & \text{如果 } T=C_j \\ \mathbf{W}_k^{\text{旧}}-\eta(\mathbf{x}^{(p)}-\mathbf{W}_k^{\text{旧}}) & \text{如果 } T\neq C_j\end{cases} \tag{8-21}$$

步骤 3. 调整学习率:

学习率作为一迭代函数减小。

步骤 4. 检查终止条件:

如果满足终止条件就退出,否则返到步骤 2。

完整的学习矢量量化算法的 $C++$ 实现程序,见清单 8.4。

LVQ 网络已被应用于字符识别问题中。Baykal 和 Yalabki[1992]已利用一个 Kohonen LVQ 网络与前馈网络相连接,用于多字体字符识别中。据报道,即使在有失真、移位、转动的字符的情况下,识别率也达到 87%。

清单 8.4

```
//*************************************************************
// LEARNING VECTOR QUANTIZATION
//*************************************************************

#include <stdlib.h>
#include <stdio.h>
#include <math.h>

#define  MAXPATS     100
#define  MAXNEURONSIN 50
#define  MAXNEURONS   15

#define  MAXEPOCHS    1500
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
#define  ETAMIN     .001

unsigned int Random(int N) {
 unsigned int j;
j= (N*rand())/RAND_MAX;
if (j>=N) j=N;
return j;
}

class TPATTERN {
 friend class LVQ;
private:
 double      P[MAXPATS][MAXNEURONSIN];
 int         PClass[MAXPATS];
 int         NumPatterns;
 int         NumClasses;
 int         Shuffle[MAXPATS];
 int         SizeVector;
public:
 TPATTERN();
 int GetPatterns(char *);            //load pattern form file
 int GetRandPats(int,int);           //random patterns arg1=# of patterns, arg2=dimension
 double Query(int,int);              //returns P[arg1][arg2]
 double QueryR(int,int);             //returns P[Shuffle[arg1]][arg2]
 int   QueryClass(int);
 void ReShuffle(int N);
};

TPATTERN::TPATTERN(){
int i;
for (i=0; i<MAXPATS; i++)
  Shuffle[i]=i;
}
int TPATTERN::GetPatterns(char *fname) {
  FILE *fp;
  int i,j,k;
  double x;
fp=fopen(fname,"r");
if (fp==NULL) return 0;  // Test for failure.
fscanf(fp,"%d",&NumPatterns);
fscanf(fp,"%d",&SizeVector);
fscanf(fp,"%d",&NumClasses);
for (i=0; i<NumPatterns; i++) {                  // For each vector
  for (j=0; j<SizeVector; j++) {                 // create a pattern
    fscanf(fp,"%lg",&x);                         // consisting of all elements
    P[i][j]=x;
    } /* endfor */
  fscanf(fp,"%d",&k);
  PClass[i]=k;
  } /* endfor */
fcfose(fp);
return 1;
}
int TPATTERN::GetRandPats(int n1,int n2) {
  int i,j;
  double x;
NumPatterns=n1;
SizeVector=n2;
for (i=0; i<NumPatterns; i++) {                  // For each vector
  for (j=0; j<SizeVector; j++) {                 // create a pattern
    x=(double)rand()/RAND_MAX;                   // consisting of random elements
    P[i][j]=x;                                   // between 0 and 1
    } /* endfor */
  } /* endfor */
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication address: luzhenbo@yahoo.com.cn


```
return 1;
}

void TPATTERN::ReShuffle(int N) {
int i,a1,a2,tmp;
for (i=0; i<N ;i++) {
  a1=Random(NumPatterns);
  a2=Random(NumPatterns);
  tmp=Shuffle[a1];
  Shuffle[a1]=Shuffle[a2];
  Shuffle[a2]=tmp;
   }
}

double TPATTERN::Query(int pat,int j) {
return P[pat][j];
}

double TPATTERN::QueryR(int pat,int j) {
return P[Shuffle[pat]][j];
}

int TPATTERN::QueryClass(int pat) {
return PClass[pat];
}

class LVQ {
private:
 double W[MAXNEURONSIN][MAXNEURONS];   //The weight matrix
 int    zClass[MAXNEURONS];            //Neuron Class assignment
 double Yout[MAXNEURONS];              //The output layer neurons
 double Yin[MAXNEURONSIN];             //The input layer neurons
 int    YinSize;                       //input layer dimensions
 int    YoutSize;                      //outlayer dimensions

 int    epoch;
 double eta;                           //The learning rate
 double delta_eta;                     //Amount to change l.r. each epoch
 int    StochFlg;                      //Present vectors in rand order if 1
 TPATTERN *Pattern;

 int    LoadInLayer(int);              //pattern->input layer
 double EucNorm(int);                  //Calc Euclidean distance
 int    FindWinner();                  //get coords of winning neuron
 void   Train(int,int);
 void   AdaptParms();
public:
 LVQ();
 void SetPattern(TPATTERN *);
 void SetParms(int, double);
 void PrintWeights();
 void PrintWinner();
 void RunTrn();
 void Run();
};
LVQ::LVQ(){
StochFlg=0;
}

void LVQ::SetPattern(TPATTERN *p) {
  Pattern=p;
  YinSize=p->SizeVector;
}

void LVQ::SetParms(int X, double LR){
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@hit.edu.cn


```
 int i,k,m;
YoutSize=X;
eta=LR;
delta_eta=0.002;
for (i=0; i<YoutSize; i++) {
  for (k=0; k<YinSize; k++) {
    W[k][i]= (double)rand()/(10.0 * (double)RAND_MAX);
    //W[k][i]= (double)rand()/((double)RAND_MAX);
     } /* endfor */
  m=YoutSize/Pattern->NumClasses;
  zClass[i]= i/m;
  } /* endfor */
}

int LVQ::LoadInLayer(int P){
  int i;
for (i=0; i<YinSize; i++){
  if (StochFlg){
    Yin[i]=Pattern->QueryR(P,i);
     }
   else {
    Yin[i]=Pattern->Query(P,i);
     }
  }
return 1;
}

void LVQ::AdaptParms(){
eta=eta-delta_eta;
if (eta<ETAMIN)
  eta=ETAMIN;
}

void LVQ::PrintWeights() {
  int i,k;
for (i=0; i<YoutSize; i++) {
   printf("W[%d]=",i);
   for (k=0; k<YinSize; k++) {
     printf("%f ",W[k][i]);
     } /* endfor */
   printf("\n");
   } /* endfor */
}

void LVQ::RunTrn(){
int pat,np;
int k,z;
int Winner;
epoch=0;
np=Pattern->NumPatterns;
while (epoch<=MAXEPOCHS){
  if( (epoch<=50) || (25*(epoch/25)==epoch) ) {          //output control
    printf("EPOCH=%d\n",epoch);
    printf("eta=%f\n",eta);
     }
  for (pat=0; pat<np; pat++){                            //Traverse all patterns
   LoadInLayer(pat);
   Winner=FindWinner();
   if( (epoch<=50) || (25*(epoch/25)==epoch) ) {         //output control
     printf("winner=%d/pat=%d\n",Winner,pat);
     printf("winner class=%d/pat class=%d\n",zClass[Winner],Pattern->QueryClass(pat));
      }
   Train(Winner,pat);
   if( (epoch<=50) || (25*(epoch/25)==epoch) ) {         //output control
     printf("W[%d]=",Winner);
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
        z=1;
        for (k=0; k<YinSize; k++) {
          printf("%f ",W[k][Winner]);
          if (z>4){
            printf("\n     ");
            z=1;
            }
          else
            z++;
          } /* endfor */
        printf("\n\n");
        }
      //printf("\n");
      }
// if(1*(epoch/1)==epoch) {                          //output control
//  printf("Epoch=%d\n",epoch);
//  PrintWeights();
//  }
  epoch++;                                          //keep track of epochs
  if (StochFlg)                                     // if desired
    Pattern->ReShuffle(np);                         //  reorder training patterns
  AdaptParms();                                     //Adjust the learning rate
  }
}
void LVQ::Run(){
 int pat,np,i;
 int Winner;
printf("\n");
np=Pattern->NumPatterns;
for (pat=0; pat<np; pat++){                         //Traverse all patterns
  LoadInLayer(pat);
  Winner=FindWinner();
  printf("Responding neuron %d is of class %d \n",Winner,zClass[Winner]);
  printf("The desired class for pattern %d is: %d\n",pat,Pattern->QueryClass(pat));
  printf("The distances to each of the output layer neurons are:\n");
  for (i=0;i<YoutSize ; i++) {
    printf("distance from pattern %d to neuron %d is: %f\n",pat,i,EucNorm(i));
    } /* endfor */
  printf("\n");
  }
}

void LVQ::Train(int Winner, int pat){
  int c,k;
c=Pattern->QueryClass(pat);
if (c==zClass[Winner]) {
  for (k=0; k<YinSize; k++){
    W[k][Winner]=W[k][Winner]+eta*(Yin[k]-W[k][Winner]);
    } /*endfor*/
  }
 else {
  for (k=0; k<YinSize; k++){
    W[k][Winner]=W[k][Winner]-eta*(Yin[k]-W[k][Winner]);
    } /*endfor*/
  } /* endif */
}

int LVQ::FindWinner(){
  int i;
  double d,best;
  int Winner;
best=1.0e99;
Winner=-1;
for (i=0; i<YoutSize; i++){
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@hotmail.com


```
 d=EucNorm(i);
 if (d<best) {
   best=d;
   Winner=i;
   } // endif
 } // endfor
return Winner;
}

double LVQ::EucNorm(int x){                    // Calc Euclidean norm of vector dif
int i;
double dist;
dist=0;
for (i=0; i< YinSize;i++){
  dist += (W[i][x]-Yin[i]) * (W[i][x]-Yin[i]);
  } /* endfor */
dist=sqrt(dist);
return dist;
}

//=============================================================
// GLOBAL OBJECTS
//=============================================================
TPATTERN InPat;
TPATTERN InPat2;
LVQ   net;
//=============================================================
// Main()
//=============================================================

main(int argc, char *argv[]) {
//srand(17);
if (argc>1) {
  InPat.GetPatterns(argv[1]);                  //Establish training pattern
  net.SetPattern(&InPat);                      //Inform the net about the pattern
// net.SetParms(8, 0.2500);                    //Init fm parms
  net.SetParms(4, 0.2500);                     //Init fm parms
// net.SetParms(4, 0.3000);                    //Init fm parms
  net.RunTrn();                                //Run the FM w/ training enabled
  InPat2.GetPatterns(argv[2]);                 //Establish test pattern
  net.SetPattern(&InPat2);                     //Inform the net about the new pattern
  net.Run();
  }
 else {
  printf("USAGE: LVQ PATTERN_FILE");
  }
}
```

8.6.1 LVQ举例

为了说明学习矢量量化，我们选择了一个字符识别的例子。对应于4个不同类的8个训练模式被应用于LVQ网络。训练模式和它们相应的类，如图8.20所示。完成训练之后，将用两个测试矢量对网络进行测试，如图8.21所示。根据输入的维数的需要，本例中的两络拓扑采用 x 个神经元的输入层。输出层由四个神经元的一维矩阵组成，每一类对应一个神经元。在典型情况下，LVQ网络对于每一类往往对应一个以上神经元。在本例中，由于示例的有限范围，一个就足够了。与图8.20相应的输入训练矢量将随后介绍。

训练网络的准备工作已经完成。对于每种模式，获胜输出层神经元已表示出来。然后获胜神经元类别将在模式类别旁边表示出来。最后权值或码书矢量在训练之后用获胜神经元表

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

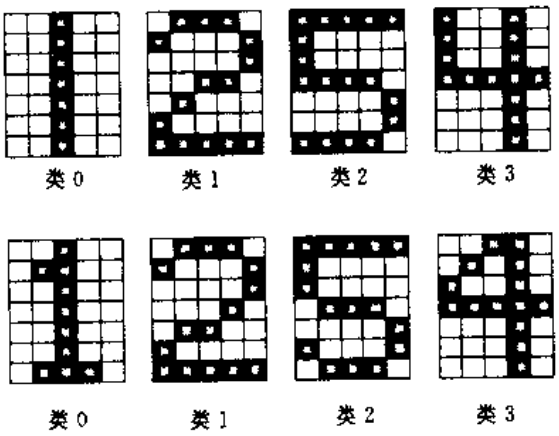


图 8.20 提供给 LVQ 网络的样本训练矢量

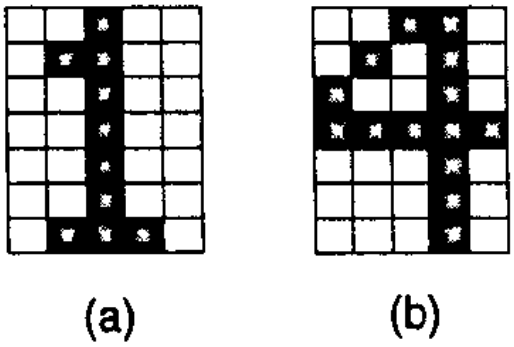


图 8.21 样本测试矢量,相应的类别判别正是预期的测试结果

示。第一次迭代的计算如下:

迭代=0 $\eta=0.250000$

获胜者=1/模式=0
获胜类别=1/模式类别=0

W[1]=0.072737 0.048654 −0.205546 0.025029 0.103366
0.051988 0.057939 −0.127602 0.015805 0.026578
0.119808 0.092185 −0.198870 0.097514 0.094737
0.119606 0.003510 −0.210158 0.094619 0.030374
0.073695 0.005425 −0.130493 0.039892 0.007420
0.055235 0.114380 −0.178468 0.014855 0.071223
0.031507 0.061983 −0.220408 0.059622 0.050760

获胜者=3/模式=1
获胜类别=3/模式类别=1

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xaut.edu.cn

W[3]= 0.025998 −0.232505 −0.213176 −0.149648 0.027367
−0.179613 0.089450 0.024693 0.123730 −0.218745
0.053827 0.094409 0.107616 0.111850 −0.127739
0.049425 0.054025 −0.234107 −0.192789 0.029729
0.123257 −0.168397 0.075529 0.030236 0.056860
−0.151254 0.009850 0.059549 0.019074 0.030717
−0.131874 −0.173246 −0.126476 −0.190340 −0.150037

获胜者=0/模式=2
获胜类别=0/模式类别=2
W[0]= −0.185766 −0.228034 −0.211421 −0.183184 −0.131546
−0.228534 0.087779 0.028302 0.061846 0.015587
−0.239513 0.048704 0.034654 0.046007 0.122932
−0.183077 −0.154290 −0.169191 −0.154107 0.097530
0.102870 0.018990 0.078185 0.039335 −0.206637
0.114650 0.064970 0.050146 0.075846 −0.151822
−0.133557 −0.141259 −0.141682 −0.165685 0.094802

获胜者=2/模式=3
获胜类别=2/模式类别=3
W[2]= −0.140873 0.053369 0.044778 −0.202254 0.005394
−0.229926 0.065294 0.087073 −0.237865 0.050104
−0.153321 0.030602 0.042852 −0.221252 0.037233
−0.211932 −0.139100 −0.245418 −0.168607 −0.200175
0.084536 0.091575 0.117225 −0.220840 0.104812
0.120903 0.097331 0.053938 −0.165738 0.101173
0.019845 0.034986 0.016915 −0.141976 0.093776

获胜者=2/模式=4
获胜类别=2/模式类别=0
W[2]= −0.176092 0.066712 −0.194027 −0.252817 0.006743
−0.287408 −0.168382 −0.141159 −0.297331 0.062630
−0.191651 0.038253 −0.196435 −0.276564 0.046541
−0.264915 −0.173874 −0.556773 −0.210759 −0.250218
0.105670 0.114468 −0.103468 −0.276049 0.131015
0.151129 0.121664 −0.182578 −0.207173 0.126466
0.024806 −0.206268 −0.228856 −0.427470 0.117220

获胜者=1/模式=5
获胜类别=1/模式类别=1

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2

Communication: E-mail: luzhenbo@yahoo.com.cn

W[1]=0.054553 0.286491 0.095841 0.268772 0.077525
0.288991 0.043455 −0.096701 0.011854 0.269933
0.089856 0.069139 −0.149153 0.073136 0.321053
0.089704 0.002632 −0.157619 0.320964 0.022780
0.055271 0.254068 0.152130 0.029919 0.005565
0.291426 0.085785 −0.133851 0.011141 0.053417
0.273630 0.296487 0.084694 0.294716 0.288070

获胜者=1/模式=6
获胜类别=1/模式类别=2
W[1]= −0.181809 0.108113 −0.130199 0.085965 −0.153094
0.111239 0.054318 −0.119627 0.014817 0.337417
−0.137680 0.086423 −0.186441 0.091420 0.401316
0.112130 −0.246710 −0.447023 0.151205 0.028475
0.069089 0.317586 0.190163 0.037398 −0.243044
0.114283 0.107231 −0.167314 0.013926 −0.183229
0.342037 0.120609 −0.144133 0.118395 0.360087

获胜者=3/模式=7
获胜类别=3/模式类别=3
W[3]= 0.019498 −0.174379 0.090118 0.137764 0.020526
−0.134710 0.317087 0.018520 0.342797 −0.164059
0.290370 0.070807 0.080712 0.333888 −0.095804
0.287069 0.290519 0.074419 0.105408 0.272297
0.092442 −0.126298 0.056647 0.272677 0.042645
−0.113440 0.007387 0.044662 0.264306 0.023038
−0.098906 −0.129934 −0.094857 0.107254 −0.11527

注意到在第一次迭代期间,获胜神经元的类别经常与训练模式类别不匹配。在这种情况下,训练权值以便获胜神经元权值矢量与公式(8-21)中所指明的模式矢量有更多的不同。

到第 10 次迭代为止,训练效果已经较好,达到获胜神经元类别几乎经常与训练矢量类别相一致。第 10 次迭代的计算如下:

迭代=10 η=0.230000

获胜者=0/模式=0
获胜类别=0/模式类别=0
W[0]= −0.182456 −0.183187 0.979025 −0.182411 −0.181517
−0.183196 0.532081 1.162414 0.001071 0.000270
−0.183386 0.000843 1.162524 0.000797 0.002129
−0.003170 −0.181911 0.979756 −0.181907 0.001689
0.001781 0.000329 1.163278 0.000681 −0.182817

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xaut.edu.cn

-0.177254 0.001125 1.162793 0.001313 -0.181868
-0.002312 0.348876 0.980232 0.348453 0.001641

获胜者=1/模式=1
获胜类别=1/模式类别=1
W[1]= -0.326132 0.977788 0.971853 0.977236 -0.325417
0.977866 0.001353 -0.002979 0.000369 1.305103
-0.325033 0.002152 -0.004643 0.002277 1.306695
-0.224235 -0.327749 0.389607 0.978861 0.000709
0.001721 1.304609 0.579091 0.000931 -0.327657
1.204969 0.002671 -0.004167 0.000347 -0.326168
1.078190 0.978099 0.971506 0.978044 1.305668

获胜者=1/模式=2
获胜类别=1/模式类别=2
W[1]= -0.631143 0.972679 0.965379 0.972001 -0.630263
0.972775 0.001664 -0.003664 0.000454 1.605277
-0.629791 0.002647 -0.005711 0.002800 1.607234
-0.505809 -0.633131 0.249216 0.973999 0.000872
0.002116 1.604669 0.712283 0.001146 -0.633019
1.482112 0.003285 -0.005125 0.000427 -0.631186
1.096174 0.973062 0.964952 0.972994 1.605971

获胜者=3/模式=3
获胜类别=3/模式类别=3
W[3]= 0.385936 -0.341704 0.250249 0.979510 -0.335685
0.381174 0.599714 -0.005565 1.322161 -0.005066
0.984223 0.002187 -0.003644 1.0321885 -0.002959
1.158810 0.984227 0.971417 0.978511 1.319983
0.002855 -0.003900 -0.004387 1.319995 -0.335002
-0.178193 0.000228 -0.004757 1.319737 -0.335608
-0.164684 -0.340332 -0.345385 0.978567 -0.003475

获胜者=0/模式=4
获胜类别=0/模式类别=0
W[0]= -0.140491 -0.141054 0.983849 -0.140456 -0.139768
-0.141061 0.639702 1.125059 0.000825 0.000208
-0.141207 0.000649 1.125144 0.000613 0.001639
-0.002441 -0.140071 0.984412 -0.140069 0.001300
0.001371 0.000253 1.125724 0.000524 -0.140769

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Copyright: luzhenbo@yahoo.com.cn

-0.136486 0.000866 1.125350 0.001011 -0.140038
-0.001781 0.498635 0.984779 0.498309 0.001264

获胜者=1/模式=5
获胜类别=1/模式类别=1
W[1]= -0.485980 0.978963 0.973342 0.978441 -0.485303
0.979037 0.001281 -0.002822 0.000349 1.466063
-0.484939 0.002038 -0.004398 0.002156 1.467570
-0.389473 -0.487511 0.191897 0.979979 0.000672
0.001630 1.465595 0.778458 0.000882 -0.487424
1.371227 0.002529 -0.003946 0.000328 -0.486014
1.074054 0.979258 0.973013 0.979205 1.466598

获胜者=0/模式=6
获胜类别=0/模式类别=2
W[0]= -0.402804 -0.403497 0.980134 -0.402761 -0.401914
-0.403505 0.786834 1.383823 0.001014 0.000256
-0.403685 0.000799 1.383927 0.000754 0.002016
-0.003002 -0.402287 0.980827 -0.402284 0.001599
0.001687 0.000311 1.384641 0.000645 -0.403146
-0.397877 0.001065 1.384181 0.001244 -0.402247
-0.002190 0.383321 0.981278 0.382920 0.001555
获胜者=3/模式=7
获胜类别=3/模式类别=3
W[3]= 0.297171 -0.263112 0.422692 0.984223 -0.258478
0.293504 0.691780 -0.004285 1.248064 -0.003901
0.987851 0.001684 -0.002806 1.247852 -0.002278
1.122284 0.987855 0.977991 0.983453 1.246387
0.002198 -0.003003 -0.003378 1.246396 -0.257952
-0.137209 0.000176 -0.003663 1.246197 -0.258418
-0.126806 -0.262055 -0.265946 0.983497 -0.002676

到第 20 次迭代为止,在整个迭代过程中,获胜神经元和模式矢量类别已经都一致了。第 20 次迭代的训练如下:

迭代=20 η=0.210000

获胜者=0/模式=0
获胜类别=0/模式类别=0
W[0]= -0.065621 -0.065633 0.999673 -0.065621 -0.0065607
-0.065633 0.475598 1.065309 0.000017 0.000004
-0.065636 0.000013 1.065311 0.000012 0.000033

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

−0.031805	−0.065613	0.999684	−0.065613	0.000026
0.000028	0.000005	1.065322	0.000011	−0.065627
−0.033785	0.000018	1.065315	0.000020	−0.065612
−0.031792	0.409964	0.999692	0.409958	0.000026

获胜者＝1/模式＝1
获胜类别＝1/模式类别＝1

W[1]＝

−0.121791	0.999482	0.999344	0.999469	−0.121774
0.999484	0.000032	−0.000069	0.000009	1.211301
−0.121765	0.000050	−0.000108	0.000053	1.121338
−0.040615	−0.121829	0.493460	0.999507	0.000017
0.000040	1.121289	0.505810	0.000022	−0.121827
1.040166	0.000062	−0.000097	0.000008	−0.121792
1.080624	0.999489	0.999336	0.999488	1.121314

获胜者＝2/模式＝2
获胜类别＝2/模式类别＝2

W[2]＝

0.720564	0.778254	0.716303	0.702335	0.764005
0.694116	−0.040007	−0.033539	−0.070645	0.014881
0.716867	0.009089	−0.046672	−0.065711	0.011058
0.361543	0.721091	0.630116	0.712327	−0.059451
0.025107	0.027197	−0.024584	−0.065588	0.793532
0.373825	0.028907	−0.043380	−0.049224	0.792451
0.430380	0.713395	0.708028	0.660838	0.027851

获胜者＝3/模式＝3
获胜类别＝3/模式类别＝3

W[3]＝

0.510088	−0.086869	0.402623	0.999611	−0.086754
0.509998	0.489639	−0.000106	1.086498	−0.000096
0.999700	0.000042	−0.000069	1.086492	−0.000056
1.013832	0.999701	0.999457	0.999592	1.086456
0.000054	−0.000074	−0.000083	1.086456	−0.086741
−0.014200	0.000004	−0.000090	1.086452	−0.086753
−0.072690	−0.086843	−0.086938	0.999593	−0.000066

获胜者＝0/模式＝4
获胜类别＝0/模式类别＝0

W[0]＝

−0.051841	−0.051850	0.999742	−0.051840	−0.051829
−0.051850	0.585722	1.051594	0.000013	0.000003
−0.051852	0.000010	1.051595	0.000010	0.000026

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication E-mail: luzhenbo@yahoo.com.cn

−0.025126	−0.051834	0.999751	−0.051834	0.000021
0.000022	0.000004	1.051605	0.000008	−0.051845
−0.026690	0.000014	1.051599	0.000016	−0.051834
−0.025115	0.533872	0.999757	0.533867	0.000020

获胜者＝1/模式＝5
获胜类别＝1/模式类别＝1

W[1]＝ −0.096215	0.999591	0.999482	0.999581	−0.096202
0.999592	0.000025	−0.000055	0.000007	1.095828
−0.096195	0.000040	−0.000086	0.000042	1.095857
−0.032086	−0.096245	0.389834	0.999611	0.000013
0.000032	1.095818	0.609590	0.000017	−0.096243
1.031731	0.000049	−0.000077	0.000006	−0.096216
1.063693	0.999597	0.999475	0.999596	1.095838

获胜者＝2/模式＝6
获胜类别＝2/模式类别＝2

W[2]＝0.779246	0.824820	0.775879	0.764844	0.813564
0.758352	−0.031606	−0.026496	−0.055810	0.011756
0.776325	0.007180	−0.036871	−0.051912	0.008736
0.285619	0.779662	0.707791	0.772739	−0.046966
0.019834	0.021486	−0.019421	−0.051815	0.836890
0.505322	0.022837	−0.034270	−0.038887	0.836036
0.340000	0.773582	0.769342	0.732062	0.022002

获胜者＝3/模式＝7
获胜类别＝3/模式类别＝3

W[3]＝ 0.402970	−0.068626	0.528072	0.999693	−0.068536
0.402898	0.596815	−0.000083	1.068333	−0.000076
0.999763	0.000033	−0.000055	1.068329	−0.000044
1.010928	0.999763	0.999571	0.999678	1.068300
0.000043	−0.000058	−0.000066	1.068301	−0.068526
−0.011218	0.000003	−0.000071	1.068297	−0.068535
−0.057425	−0.068606	−0.068681	0.999679	−0.000052

这时，训练必须持续进行，直到对每个类别都形成其聚类中心为止。注意到随着训练过程的进行，η 值在减小。最后一次迭代的过程如下：

迭代＝1500　η＝0.001000

获胜者＝0/模式＝0
获胜类别＝0/模式类别＝0

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xaut.edu.cn

W[0]= -0.000000 -0.000000 1.000000 -0.000000 -0.000000
-0.000000 0.500377 1.000000 0.000000 0.000000
-0.000000 0.000000 1.000000 0.000000 0.000000
-0.000000 -0.000000 1.000000 -0.000000 0.0000000
0.000000 0.000000 1.000000 0.000000 -0.000000
-0.000000 0.000000 1.000000 0.000000 -0.000000
-0.000000 0.500377 1.000000 0.500377 0.000000

获胜者=1/模式=1
获胜类别=1/模式类别=1
W[1]= -0.000000 1.000000 1.000000 1.000000 -0.000000
1.000000 0.000000 -0.000000 0.000000 1.000000
-0.000000 0.000000 -0.000000 0.000000 1.000000
-0.000000 -0.000000 0.499623 1.000000 0.000000
0.000000 1.000000 0.500377 0.000000 -0.000000
1.000000 0.000000 -0.000000 0.000000 -0.000000
1.000000 1.000000 1.000000 1.000000 1.000000

获胜者=2/模式=2
获胜类别=2/模式类别=2
W[2]=1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 -0.000000 -0.000000 -0.000000 0.000000
1.000000 0.000000 -0.000000 -0.000000 0.000000
0.499623 1.000000 1.000000 1.000000 -0.000000
0.000000 0.000000 -0.000000 -0.000000 1.000000
0.500377 0.000000 -0.000000 -0.000000 1.000000
0.499623 1.000000 1.000000 1.000000 0.000000

获胜者=3/模式=3
获胜类别=3/模式类别=3
W[3]= 0.499623 -0.000000 0.500377 1.000000 -0.000000
0.499623 0.500377 -0.0000001.000000 -0.000000
1.000000 0.000000 -0.0000001.000000 -0.000000
1.000000 1.000000 1.0000001.000000 1.000000
0.000000 -0.000000 -0.0000001.000000 -0.000000
-0.000000 0.000000 -0.0000001.000000 -0.000000
-0.000000 -0.000000 -0.0000001.000000 -0.000000

获胜者=0/模式=4
获胜类别=0/模式类别=0

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication or Cooperation: luzhenbo@yahoo.com.cn

W[0] = −0.000000 −0.000000 1.000000 −0.000000 −0.000000
−0.000000 0.500877 1.000000 0.000000 0.000000
−0.000000 0.000000 1.000000 0.000000 0.000000
−0.000000 −0.000000 1.000000 −0.000000 0.000000
0.000000 0.000000 1.000000 0.000000 −0.000000
−0.000000 0.000000 1.000000 0.000000 −0.000000
−0.000000 0.500877 1.000000 0.500877 0.000000

获胜者＝1/模式＝5
获胜类别＝1/模式类别＝1
W[1] = −0.000000 1.000000 1.000000 1.000000 −0.000000
1.000000 0.000000 −0.000000 0.000000 1.000000
−0.000000 0.000000 −0.000000 0.000000 1.000000
−0.000000 −0.000000 0.499123 1.000000 0.000000
0.000000 1.000000 0.500877 0.000000 −0.000000
1.000000 0.000000 −0.000000 0.000000 −0.000000
1.000000 1.000000 1.000000 1.000000 1.000000

获胜者＝2/模式＝6
获胜类别＝2/模式类别＝2
W[2] = 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 −0.000000 −0.000000 −0.000000 0.000000
1.000000 0.000000 −0.000000 −0.000000 0.000000
0.499123 1.000000 1.000000 1.000000 −0.000000
0.000000 0.000000 −0.000000 −0.000000 1.000000
0.500877 0.000000 −0.000000 −0.000000 1.000000
0.499123 1.000000 1.000000 1.000000 0.000000

获胜者＝3/模式＝7
获胜类别＝3/模式类别＝3
W[3] = 0.499123 −0.000000 0.500877 1.000000 −0.000000
0.499123 0.500877 −0.000000 1.000000 −0.000000
1.000000 0.000000 −0.000000 1.000000 −0.000000
1.000000 1.000000 1.000000 1.000000 1.000000
0.000000 −0.000000 −0.000000 1.000000 −0.000000
−0.000000 0.000000 −0.000000 1.000000 −0.000000
−0.000000 −0.000000 −0.000000 1.000000 −0.000000

在网络训练完之后，下面可以应用测试矢量。从下面的与图 8.21(a)相应的模式开始这一过程：

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/u/1501...
Communication & Cooperation: lzhenbo@yahoo.com.cn

0.0 0.0 1.0 0.0 0.0
0.0 1.0 1.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0
0.0 1.0 1.0 1.0 0.0

网络计算出对该测试矢量的测试结果。到每个输出层神经元的距离是:

从模式 0 到神经元 0 的距离是:0.864507
从模式 0 到神经元 1 的距离是:3.807887
从模式 0 到神经元 2 的距离是:3.968517
从模式 0 到神经元 3 的距离是:4.122681

神经元 0(代表类别 0)对该模式有反应,因为它有最短的欧氏距离。对于模式 0 所预计的类别也是类别 0,因而网络已正确地完成了对该模式的分类,它属于类别 0。

下面我们应用与图 8.21(b)相应的第二个测试矢量:

0.0 0.0 1.0 1.0 0.0
0.0 1.0 0.0 1.0 0.0
1.0 0.0 0.0 1.0 0.0
1.0 1.0 1.0 1.0 1.0
0.0 0.0 0.0 1.0 0.0
0.0 0.0 0.0 1.0 0.0
0.0 0.0 0.0 1.0 0.0

网络计算出对该测试矢量的响应,结果如下:
神经元 3 的响应结果是类别 3
模式 1 的预计类别是类别 3。
每个输出层神经元的距离是:

从模式 1 到神经元 0 的距离是:3.968517
从模式 1 到神经元 1 的距离是:4.416079
从模式 1 到神经元 2 的距离是:3.840687
从模式 1 到神经元 3 的距离是:0.998247

神经元 3(代表类别 3)对该模式有反应,因为它有最短的欧氏距离。对模式 2 所预计的类别是 3。因而网络已正确地完成了对该模式的分类,它属于类别 3。这样,网络又正确地评测了该测试矢量。

8.7 LVQ 的改进

下面看一看已经研究过的有关基本 LVQ 方法的几种改进形式,尤其是我们将涉及由 Kohonen [1990a]提出的对该算法的几种改进算法:LVQ2,LVQ2.1 和 LVQ3。所有这些算

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication mailbox: luzhenbo@yahoo.com.cn

法,都围绕着如何训练网络这一问题来进行变化,而网络结构将保留不变。特别地,训练将不再仅仅限制于获胜神经元。

8.7.1 LVQ2

学习矢量量化可通过在适当条件下训练获胜矢量和紧接着的次获胜者来扩展。主要思想是当获胜神经元不代表正确类别而次获胜神经元代表正确类别时,我们希望它们都可以得到训练。获胜者远离样本模式矢量,而次获胜者被训练得更近。获胜者被惩罚,与正确类别中最近的矢量得到奖励。LVQ2 需满足下列条件:从输入矢量到获胜者的距离 d_c 与输入矢量到次获胜者之间的距离 d_r 没有很大差别(当上面的条件不满足时,训练过程与以前的 LVQ 方法相同)。

d_c 和 d_r 之间没有多大差别这一条件,可用一个"窗"来表示,在这个"窗"内认为满足条件。这个"窗"定义为:

$$\frac{d_c}{d_r}=1-\varepsilon$$
$$\frac{d_c}{d_r}=1+\varepsilon \tag{8-22}$$

当上面的所有条件都满足时,LVQ2 的训练过程如下:

$$\mathbf{W}_R^{新}=\mathbf{W}_R^{旧}+\eta(\mathbf{x}^{(p)}-\mathbf{W}_R^{旧})$$
$$\mathbf{W}_C^{新}=\mathbf{W}_C^{旧}-\eta(\mathbf{x}^{(p)}-\mathbf{W}_C^{旧}) \tag{8-23}$$

这里 $\mathbf{W}_R$ 是次获胜者的参考矢量,它和训练矢量 $\mathbf{x}^{(p)}$ 属于同一类别,而 $\mathbf{W}_C$ 是获胜神经元的参考矢量(其中获胜神经元不属于 $\mathbf{x}^{(p)}$ 的正确类别)。

8.7.2 LVQ2.1

对学习矢量量化做进一步的修改[Kohonen,1990a],获胜神经元不是训练矢量的正确类别的要求(在 LVQ2 中)可以被忽略。在这里,最好的两个参考矢量被训练。假定它们之一属于正确类别,而另一个不属于,而不考虑是最佳的还是次获胜者代表正确类别。LVQ2.1 对"窗"的条件作了略微的修改,其形式如下:

$$\min\left[\frac{d_{C_1}}{d_{C_2}},\frac{d_{C_2}}{d_{C_1}}\right]>1-\varepsilon \tag{8-24}$$

和

$$\max\left[\frac{d_{C_1}}{d_{C_2}},\frac{d_{C_2}}{d_{C_1}}\right]<1+\varepsilon \tag{8-25}$$

这里 d_{C_1} 是从 $\mathbf{x}^{(p)}$ 到 $\mathbf{W}_{C_1}$ 的距离,d_{C_2} 是从 $\mathbf{x}^{(p)}$ 到 $\mathbf{W}_{C_2}$ 的距离。$\mathbf{W}_{C_2}$ 和 $\mathbf{W}_{C_2}$ 是两个最佳参考矢量。当以上条件满足时,训练过程如下:

$$\mathbf{W}_{C_1}^{新}=\mathbf{W}_{C_1}^{旧}+\eta(\mathbf{x}^{(p)}-\mathbf{W}_{C_1}^{旧})$$
$$\mathbf{W}_{C_2}^{新}=\mathbf{W}_{C_2}^{旧}-\eta(\mathbf{x}^{(p)}-\mathbf{W}_{C_2}^{旧}) \tag{8-26}$$

这里已把 $\mathbf{W}_{C_1}$ 作为代表正确类别的神经元。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

8.7.3 LVQ3

Kohonen[1990a]提出了 LVQ 的另一种变形 LVQ3。LVQ3 算法又改变了"窗"的条件，对其进行了扩展。在这一条件下，获胜神经元和次获胜神经元都可以得到训练。这里，当获胜神经元和次获胜神经元属于不同的类别时，训练过程和 LVQ2.1 的相同。当获胜神经元和次获胜神经元属于同一类，并且满足"窗"的条件时，获胜者和次获胜者均被训练。LVQ3 中的"窗"的标准如下：

$$\min\left[\frac{d_{C_1}}{d_{C_2}},\frac{d_{C_2}}{d_{C_1}}\right]>\frac{1-\varepsilon}{1+\varepsilon} \tag{8-27}$$

当满足以上标准时，训练如下完成：

$$\mathbf{W}_C^{新}=\mathbf{W}_C^{旧}+\beta(\mathbf{x}^{(p)}-\mathbf{W}_C^{旧}) \tag{8-28}$$

其中

$$\beta=m\eta \tag{8-29}$$

乘数 m 的范围是 $0.1<m<0.5$。ε 的典型值为 0.2 左右。乘数 m 也可以变化，以便于一个更狭窄的"窗"来产生一个更小的乘数。

8.7.4 LVQ的最后变形

Yizhak 和 Chevalier[1991]提出了一种 Kohonen 网络的有监督学习改进形式(该变形不同于 LVQ 学习)，将其用于手写数字识别。在这一变形中，两组神经元集合被连接到一个 Kohonen 层中。第一组是常规输入层 I，第二组 T 在训练期间把预计的输出/目标模式分别送到 Kohonen 层中，然后后者将作为网络的输出层。权值有两组，一组为 I 层的，另一组为 T 层的。

I 层总是接收输入模式矢量，它所关联的权值被指定为 $\mathbf{W}_{ij}$。与 I 层相连的权值 $\mathbf{W}_{ij}$ 相对于输入模式矢量进行训练。在训练期间，T 层作为一个输入而起作用，并且接收预计的输出模式。因而，T 层权值(被指定为 $\mathbf{W}_{ik}$)相对于预计的输出进行训练。

训练之后，在识别阶段期间，T 层的作用反过来了，T 层神经元变成有权值 $\mathbf{W}_{ik}$ 的输出了。训练过程中，获胜 Kohonen 层的神经元邻域范围内的所有神经元都同获胜者一起被修改。定义这一邻域的半径由获胜神经元的响应程度来建立。获胜神经元对输入矢量的响应越强，邻域的范围就越小(因为已经存在一个神经元，该神经元很好地标志了这一矢量)。

在这一实现过程中，进一步提出了一种在操作模式中判别网络结果不确定性的准则。建立三个参数 σ_1，σ_2，和 σ_3。如果包括所有参数或者临界值在内都满足准则的话，那么网络仅仅被认为识别了一个字符(否则，网络将被认为处于混乱状态)。σ_1 是一临界值，以便于获胜输出神经元的强度比 σ_1 更大。第二个最佳神经元的强度必须小于 σ_2。最后，最佳与次佳神经元强度的差异一定要比 σ_3 大。

本研究中用到的样本尺寸较小，为 735 个训练模式和 265 个测试模式。据报导，对于这两种训练集合和测试集合，最佳的识别率分别为 90.7 和 75.5%。在同样的条件下，对于该训练集合和测试集合，网络对输入模式的混淆率各自为 4.9 和 14.3%。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Email: luzhenbo@yahoo.com.cn

参考书及文献

Caudill, M., "A little knowledge is a dangerous thing", *AI Expert*, pp. 16–22, June 1993.

Baykal, N. and Yalabik, N., "Object orientation detection and character recognition using optimal feed-forward network and Kohonen's feature map," *SPIE*, vol. 1709, pp. 292–303, 1992.

Kohonen, T., "Self-organized formation of topographically correct feature maps." *Biol. Cybern.*, Vol. 43, pp. 59–69, 1982.

Kohonen, T., *Self-organization and Associative Memory*, 3rd ed., Springer-Verlag, Berlin, 1989.

Kohonen, T., "Improved versions of learning vector quantizations," *Int. Joint Conf. Neural Networks*, I, pp. 545–550, 1990(a).

Kohonen, T., "The self-organizing map," *Proc. IEEE*, vol. 78 no. 9, 1464–1480, 1990(b).

Yizhak, I. and Chevalier, R. C., "Handwritten digit recognition by a supervised Kohonen-like learning algorithm," *1991 IEEE Joint Conf. Neural Networks IJCNN*, pp. 1576–1581, 1991.

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

第九章 神经联想记忆和 Hopfield 网络

9.1 概 述

本章主要讨论神经联想记忆。如字面意思所示，神经联想记忆就是通过学习每对模式之间的关联来进行工作。也就是说，这种网络输入一模式时，它能够给出一个联想的响应状态。由此可看出，神经联想记忆是按内容存取记忆，而且从某种意义上讲，我们可认为联想神经网络能够像人类那样来进行学习。当网络输入一个和所有已学的标本模式不同的模式时，网络能够通过联想来输出和输入模式最为相似(按照某个标准)的标本模式。正是由于这个特性，我们发现这种网络的功能和生物学习过程很类似。我们将一个模式提供给人看时，例如字母“A”，此模式不是必须和他(她)之前所学的样本字母“A”一模一样。它只要和他(她)经验中的“A”足够相象，能够提醒他(她)想起以前遇到过的其它形式的“A”，它就能够被正确的识别出来。除了基本类似和非常相似之外，对于使用简化的人类或生物的记忆模型时，这种联想记忆网络产生的似是而非的现象这里不予讨论。如果读者对这种网络的生物似是而非现象感兴趣的话，请参阅 Hopfield[1982 年，1984 年]，Kohonen[1977 年]，或 Levine[1990 年]的著作或者文章。

几位学者包括 Nakano[1972]，Kohonen[1977，1980]以及 Anderson 和 Bower[1973]在联想记忆网络方面做了开拓性的工作。Hopfield 记忆模型在人工神经网络的复兴中起了非常重要的作用。Hopfield 提出了新的离散和连续的神经网络模型，并将它应用到神经竞争的优化问题上。例如“巡回推销商“(TSP)问题的求解，这给神经网络模型领域的科学界起了非常深远以及强有力的激励作用[Hopfield 和 Tank，1985；Tank 和 Hopfield，1987]。另外的几本书中(例如 Zurada[1992]，Freeman 和 Skapura[1991]，Fausett[1994]等)，也非常详细地讨论了这些主题。本文将讨论限制在离散的 Hopfield 记忆模型的详细描述上，以及它作为联想记忆的性能上。

在典型的数字计算机存储器当中，数据是通过它在存储器当中的地址来进行存储的。相反，在联想记忆当中并不存在可以使用的地址，因为存储的信息是空间分布的并且在整个网络中是叠加的。因此数字计算机的存储器被称作按地址存储记忆，联想记忆被称作按内容存储记忆(CAM)。

联想记忆可分为有自联想和异联想两种形式。对于自联想网络记忆，训练输入和目标输出是等同的。我们可以认为网络是通过将模式和它自己进行联想来记忆此模式的。通过模式矢量和它自己来构成外积，我们可得到一个自联想记忆的矩阵。如果我们将许多这种排列放在其它这种排列的上面，便能设计出一种能存储几种模式的存储器，并且它能够提供自联想记忆。相反，异联想网络记忆是通过将输入模式和由教师提供的不同并且是截然不同的训练模式进行联想的。

另外，联想记忆也可分为前馈和回归两种形式。在前馈网络中，信息只是从输入流向输

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

出。回归网络含有构成环路的神经元之间的连接,因此网络能够进行迭代并最后收敛到与所需的联想相对应的最小能量状态。

本章中主要考虑几种最为常用的联想记忆网络。我们首先讨论线性联想记忆(LAM),它是一单层的前馈网络。同时我们也举出一个自联想 LAM 的例子。

Hopfield 网络,或许是最为著名的自联想神经网络记忆,将在 9.3 节讨论。Hopfield 网络是回归并且是自联想的。在这种自联想记忆当中,能够存储许多不同的模式。因此如果这些模式中的任何一种被输入到网络(也就是联想记忆被设置到预存状态其一)时,网络仍将稳定在预存的那个状态。当向网络输入一扭曲的存储模式时,网络从此状态逐渐演化,最后收敛到稳定的预存模式。同时在 9.3 节中也讨论了 Hopfield 网络的收敛性和网络的存储容量。最后我们在 9.7 节讨论了双向联想记忆(BAM),在 9.8 节举了一个回归异联想形式的 BAM 的例子。

LAM 模型也称作前馈型,而 Hopfield 网络和 BAM 因为含有利于回归运转的反馈连接,它们被称作反馈型。回归模型不同于前馈模型,它需要许多次迭代才能收敛到最后的模式。

本章并不是想详尽无遗地讨论联想记忆。而是想讨论它的基本概念以及它的潜在优点和它的一些限制。

9.2 线性联想记忆(LAM)

线性联想记忆是一单层的前馈网络,它将一 K 维输入空间映射到一 N 维输出空间。映射的目的是从输入模式中的全部信息或者其中的部分信息恢复出完整的输出模式。LAM 可以是自联想的,也可以是异联想的。

(根据定义,自联想的 LAM 中的输入和输出训练集模式是相同的;因此输入空间和输出空间的维数也应是等同的,即 $K=N$。)线性联想记忆是从 1968 年到 1972 年这段期间里,由几位学者分别独立工作所发明的。Amari[1972],Anderson[1972]以及 Kohonen[1972]所写的文章,被认为是线性联想记忆方面的经典文献。

线性联想记忆可由图 9.1 描述。

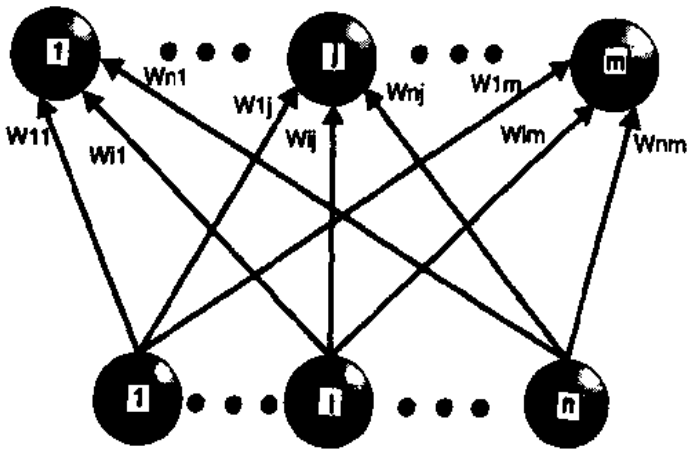


图 9.1 线性联想记忆的结构

通常映射由权值 W 以及阈值 θ 所决定,等式(9-1)到等式(9-4)指出了 LAM 网络的输出

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

是怎样确定的。

$$\hat{\mathbf{a}}^{(m)} = \mathbf{W}\mathbf{b}^{(m)} \tag{9-1}$$

或者：

$$\hat{a}_i = \sum_j W_{ij} b_j \tag{9-2}$$

这里 **a** 表示一个矢量，而 a_i 是 **a** 的第 i 个分量。$\hat{a}$表示其实际值。a_i 表示其二进制的值。

$$a_i^{(m)} = \begin{cases} 1 & \hat{a}_i^{(m)} > \theta_i \\ 0 & \hat{a}_i^{(m)} \geq \theta_i \end{cases} \tag{9-3}$$

这里对应于第 m 个模式的输入矢量 **b** 和输出矢量 **a**，由等式(9-4)给出：

$$a^{(m)} = \begin{bmatrix} a_1^{(m)} \\ a_2^{(m)} \\ \vdots \\ a_N^{(m)} \end{bmatrix} \qquad b^{(m)} = \begin{bmatrix} b_1^{(m)} \\ b_2^{(m)} \\ \vdots \\ b_K^{(m)} \end{bmatrix} \tag{9-4}$$

LAM 网络很容易以单步进行训练。训练 LAM 网络只是简单地求出输入矢量和输出矢量之间的相关性，即它们之间的权值。对于双极型取值(即输入只在[1，－1]中取值)，权值可由等式(9-5)求出：

$$W_{ij} = \sum_m a_i^{(m)} b_j^{(m)} \tag{9-5}$$

对于二值取值输入(即输入只在[0,1]中取值)，网络必须按照等式(9-6)来进行训练，因为等式(9-6)考虑到了由于这种表述所引起的 0.5 的偏差。

$$W_{ij} = \sum_m (2a_i^{(m)} - 1)(2b_j^{(m)} - 1) \tag{9-6}$$

为了得到二值输出，阈值 θ(参考等式 3)应按照等式(9-7)来进行计算：

$$\theta_i = \sum_{j=1}^{K} W_{ij} \tag{9-7}$$

上式中 K 是输入矢量 $b^{(m)}$的维数。

9.2.1 一个自联想 LAM 例子

为了描述线性联想记忆的工作机理，我们考虑如图 9.2 所示的代表数字 1 到 4 的四个模板。

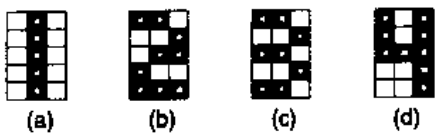


图 9.2 LAM 的输入模式

图 9.2 的这些 3×5 的图像，可以映射成 15 维的矢量。这样我们可以设计一个具有 15 个输入神经元和 15 个输出神经元的自联想网络。网络中有 225(15×15)个权来连接这些神经元，有 15 个阈值分别与 15 个输出神经元相对应。

我们将此看成矩阵记忆，那么矩阵的元素就是由外积法所得到的矢积。从神经网络的观点看，矢积被看成是连接两个神经元的权值。因此在两个不同的地方(Community)存在两个

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication E-mail: luzhenbo@yahoo.com.cn

不同但相关的观点。Kohonen[1977]在关于联想记忆的书中,详细地描述了它们的关系。

对于图 9.2 所示的四个数字,其相应的二值矢量表示如下:

$$\mathbf{b}^{(0)}=\begin{pmatrix}0\\1\\0\\0\\1\\0\\0\\1\\0\\0\\1\\0\end{pmatrix}\quad \mathbf{b}^{(1)}=\begin{pmatrix}1\\1\\0\\0\\0\\1\\0\\1\\1\\1\\0\\0\end{pmatrix}\quad \mathbf{b}^{(2)}=\begin{pmatrix}1\\1\\0\\0\\0\\1\\1\\1\\0\\0\\0\\1\\1\\1\\0\end{pmatrix}\quad \mathbf{b}^{(3)}=\begin{pmatrix}1\\0\\1\\1\\0\\1\\1\\1\\1\\0\\0\\1\\0\\0\\1\end{pmatrix} \tag{9-8}$$

对应于这些训练模式的权矩阵可由等式(9-6)如下计算(注意:自联想情况下训练集合的约束条件 $a_i=b_i$):

W[0,0]=-1*-1+1*1+1*1+1*1=4
W[0,1]=-1*1+1*1+1*1+1*-1=0
W[0,2]=-1*-1+1*-1+1*-1+1*1=0
W[0,3]=-1*-1+1*-1+1*-1+1*1=0
W[0,4]=-1*1+1*-1+1*-1+1*-1=-4
W[0,5]=-1*-1+1*1+1*1+1*1=4
W[0,6]=-1*-1+1*-1+1*1+1*1=2
W[0,7]=-1*1+1*1+1*1+1*1=2
W[0,8]=-1*-1+1*1+1*-1+1*1=2
W[0,9]=-1*-1+1*1+1*-1+1*-1=0
W[0,10]=-1*1+1*-1+1*-1+1*-1=-4
W[0,11]=-1*-1+1*-1+1*1+1*1=2
W[0,12]=-1*-1+1*1+1*1+1*-1=2
W[0,13]=-1*1+1*1+1*1+1*-1=0
W[0,14]=-1*-1+1*1+1*-1+1*1=2
W[1,0]=1*-1+1*1+1*1+-1*1=0
W[1,1]=1*1+1*1+1*1+-1*-1=4
W[1,2]=1*-1+1*-1+1*-1+-1*1=-4
W[1,3]=1*-1+1*-1+1*-1+-1*1=-4
W[1,4]=1*1+1*-1+1*-1+-1*-1=0
W[1,5]=1*-1+1*1+1*1+-1*1=0
W[1,6]=1*-1+1*-1+1*1+-1*1=-2
W[1,7]=1*1+1*1+1*1+-1*1=2
W[1,8]=1*-1+1*1+1*-1+-1*1=-2
W[1,9]=1*-1+1*1+1*-1+-1*-1=0
W[1,10]=1*1+1*-1+1*-1+-1*-1=0
W[1,11]=1*-1+1*-1+1*1+-1*1=-2
W[1,12]=1*-1+1*1+1*1+-1*-1=2
W[1,13]=1*1+1*1+1*1+-1*-1=4
W[1,14]=1*-1+1*1+1*-1+-1*1=-2
W[2,0]=-1*-1+-1*1+-1*1+1*1=0
W[2,1]=-1*1+-1*1+-1*1+1*-1=-4
W[2,2]=1*-1+-1*-1+-1*-1+1*1=4
W[2,3]=-1*-1+-1*-1+-1*-1+1*1=4
W[2,4]=-1*1+-1*-1+-1*-1+1*-1=0
W[2,5]=-1*-1+-1*1+-1*1+1*1=0
W[2,6]=-1*-1+-1*-1+-1*1+1*1=2
W[2,7]=-1*1+-1*1+-1*1+1*1=-2
W[2,8]=-1*-1+-1*1+-1*-1+1*1=2
W[2,9]=-1*-1+-1*1+-1*-1+1*-1=0
W[2,10]=-1*1+-1*-1+-1*-1+1*-1=0
W[2,11]=-1*-1+-1*-1+-1*1+1*1=2

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

W[2,12]=-1*-1+-1*1+-1*1+1*-1=-2
W[2,13]=-1*1+-1*1+-1*1+1*-1=-4
W[2,14]=-1*-1+-1*1+-1*-1+1*1=2
W[3,0]=1*-1+-1*1+-1*1+1*1=0
W[3,1]=-1*1+-1*1+-1*1+1*-1=-4
W[3,2]=1*-1+-1*-1+-1*-1+1*1=4
W[3,3]=-1*-1+-1*-1+-1*-1+1*1=4
W[3,4]=-1*1+-1*-1+-1*-1+1*-1=0
W[3,5]=-1*-1+-1*1+-1*1+1*1=0
W[3,6]=-1*-1+-1*-1+-1*1+1*1=2
W[3,7]=-1*1+-1*1+-1*1+1*1=-2
W[3,8]=-1*-1+-1*1+-1*-1+1*1=2
W[3,9]=-1*-1+-1*1+-1*-1+1*-1=0
W[3,10]=-1*1+-1*-1+-1*-1+1*-1=0
W[3,11]=-1*-1+-1*-1+-1*1+1*1=2
W[3,12]=-1*-1+-1*1+-1*1+1*-1=-2
W[3,13]=-1*1+-1*1+-1*1+1*-1=-4
W[3,14]=-1*-1+-1*1+-1*-1+1*1=2
W[4,0]=1*-1+-1*1+-1*1+-1*1=-4
W[4,1]=1*1+-1*1+-1*1+-1*-1=0
W[4,2]=1*-1+-1*-1+-1*-1+-1*1=0
W[4,3]=1*-1+-1*-1+-1*-1+-1*1=0
W[4,4]=1*1+-1*-1+-1*-1+-1*-1=4
W[4,5]=1*-1+-1*1+-1*1+-1*1=-4
W[4,6]=1*-1+-1*-1+-1*1+-1*1=-2
W[4,7]=1*1+-1*1+-1*1+-1*1=-2
W[4,8]=1*-1+-1*1+-1*-1+-1*1=-2
W[4,9]=1*-1+-1*1+-1*-1+-1*-1=0
W[4,10]=1*1+-1*-1+-1*-1+-1*-1=4
W[4,11]=1*-1+-1*-1+-1*1+-1*1=-2
W[4,12]=1*-1+-1*1+-1*1+-1*-1=-2
W[4,13]=1*1+-1*1+-1*1+-1*-1=0
W[4,14]=1*-1+-1*1+-1*-1+-1*1=-2
W[5,0]=-1*-1+1*1+1*1+1*1=4
W[5,1]=-1*1+1*1+1*1+1*-1=0
W[5,2]=-1*-1+1*-1+1*-1+1*1=0
W[5,3]=-1*-1+1*-1+1*-1+1*1=0
W[5,4]=-1*1+1*-1+1*-1+1*-1=-4
W[5,5]=-1*-1+1*1+1*1+1*1=4
W[5,6]=-1*-1+1*-1+1*1+1*1=2
W[5,7]=-1*1+1*1+1*1+1*1=2
W[5,8]=-1*-1+1*1+1*-1+1*1=2
W[5,9]=-1*-1+1*1+1*-1+1*-1=0
W[5,10]=-1*1+1*-1+1*-1+1*-1=-4
W[5,11]=-1*-1+1*-1+1*1+1*1=2
W[5,12]=-1*-1+1*1+1*1+1*-1=2
W[5,13]=-1*1+1*1+1*1+1*-1=0
W[5,14]=-1*-1+1*1+1*-1+1*1=2
W[6,0]=-1*-1+-1*1+1*1+1*1=2
W[6,1]=-1*1+-1*1+1*1+1*-1=-2
W[6,2]=-1*-1+-1*-1+1*-1+1*1=2
W[6,3]=-1*-1+-1*-1+1*-1+1*1=2
W[6,4]=-1*1+-1*-1+1*-1+1*-1=-2
W[6,5]=-1*-1+-1*1+1*1+1*1=2
W[6,6]=-1*-1+-1*-1+1*1+1*1=4
W[6,7]=-1*1+-1*1+1*1+1*1=0
W[6,8]=-1*-1+-1*1+1*-1+1*1=0
W[6,9]=-1*-1+-1*1+1*-1+1*-1=-2
W[6,10]=-1*1+-1*-1+1*-1+1*-1=-2
W[6,11]=-1*-1+-1*-1+1*1+1*1=4
W[6,12]=-1*-1+-1*1+1*1+1*-1=0
W[6,13]=-1*1+-1*1+1*1+1*-1=-2
W[6,14]=-1*-1+-1*1+1*-1+1*1=0
W[7,0]=1*-1+1*1+1*1+1*1=2
W[7,1]=1*1+1*1+1*1+1*-1=2
W[7,2]=1*-1+1*-1+1*-1+1*1=-2
W[7,3]=1*-1+1*-1+1*-1+1*1=-2
W[7,4]=1*1+1*-1+1*-1+1*-1=-2
W[7,5]=1*-1+1*1+1*1+1*1=2
W[7,6]=1*-1+1*-1+1*1+1*1=0
W[7,7]=1*1+1*1+1*1+1*1=4
W[7,8]=1*-1+1*1+1*-1+1*1=0
W[7,9]=1*-1+1*1+1*-1+1*-1=-2
W[7,10]=1*1+1*-1+1*-1+1*-1=-2
W[7,11]=1*-1+1*-1+1*1+1*1=0
W[7,12]=1*-1+1*1+1*1+1*-1=0
W[7,13]=1*1+1*1+1*1+1*-1=2
W[7,14]=1*-1+1*1+1*-1+1*1=0
W[8,0]=-1*-1+1*1+-1*1+1*1=2
W[8,1]=-1*1+1*1+-1*1+1*-1=-2

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

W[8,2]=-1*-1+1*-1+-1*-1+1*1=2
W[8,3]=-1*-1+1*-1+-1*-1+1*1=2
W[8,4]=-1*1+1*-1+-1*-1+1*-1=-2
W[8,5]=-1*-1-1*1+-1*1+1*1=2
W[8,6]=-1*-1+1*-1+-1*1+1*1=0
W[8,7]=-1*1+1*1+-1*1+1*1=0
W[8,8]=-1*-1+1*1+-1*-1+1*1=4
W[8,9]=-1*-1+1*1+-1*-1+1*-1=2
W[8,10]=-1*1+1*-1+-1*-1+1*-1=-2
W[8,11]=-1*-1+1*-1+-1*1+1*1=0
W[8,12]=-1*-1+1*1+-1*1+1*-1=0
W[8,13]=-1*1+1*1+-1*1+1*-1=-2
W[8,14]=-1*-1+1*1+-1*-1+1*1=4
W[9,0]=-1*-1+1*1+-1*1+-1*1=0
W[9,1]=-1*1+1*1+-1*1+-1*-1=0
W[9,2]=-1*-1+1*-1+-1*-1+-1*1=0
w[9,3]=-1*-1+1*-1+-1*-1+-1*1=0
W[9,4]=-1*1+1*-1+-1*-1+-1*-1=0
W[9,5]=-1*-1+1*1+-1*1+-1*1=0
W[9,6]=-1*-1+1*-1+-1*1+-1*1=-2
W[9,7]=-1*1+1*1+-1*1+-1*1=-2
W[9,8]=-1*-1+1*1+-1*-1+-1*1=2
W[9,9]=-1*-1+1*1+-1*-1+-1*-1=4
W[9,10]=-1*1+1*-1+-1*-1+-1*-1=0
W[9,11]=-1*-1+1*-1+-1*1+-1*1=-2
W[9,12]=-1*-1+1*1+-1*1+-1*-1=2
W[9,13]=-1*1+1*1+-1*1+-1*-1=0
W[9,14]=-1*-1+1*1+-1*-1+-1*1=2
W[10,0]=1*-1+-1*1+-1*1+-1*1=-4
W[10,1]=1*1+-1*1+-1*1+-1*-1=0
W[10,2]=1*-1+-1*-1+-1*-1+-1*1=0
W[10,3]=1*-1+-1*-1+-1*-1+-1*1=0
W[10,4]=1*1+-1*-1+-1*-1+-1*-1=4
W[10,5]=1*-1+-1*1+-1*1+-1*1=-4
W[10,6]=1*-1+-1*-1+-1*1+-1*1=-2
W[10,7]=1*1+-1*1+-1*1+-1*1=-2
W[10,8]=1*-1+-1*1+-1*-1+-1*1=-2
W[10,9]=1*-1--1*1+-1*-1+-1*-1=0
W[10,10]=1*1+-1*-1+-1*-1+-1*-1=4
W[10,11]=1*-1+-1*-1+-1*1+-1*1=-2
W[10,12]=1*-1+-1*1+-1*1+-1*-1=-2
W[10,13]=1*1+-1*1+-1*1+-1*-1=0
W[10,14]=1*-1+-1*1+-1*-1*-1+1=-2
W[11,0]=-1*-1+-1*1+1*1+1*1=2
W[11,1]=-1*1+-1*1+1*1+1*-1=-2
W[11,2]=-1*-1+-1*-1+1*-1+1*1=2
W[11,3]=-1*-1+-1*-1+1*-1+1*1=2
W[11,4]=-1*1+-1*-1+1*-1+1*-1=-2
W[11,5]=-1*-1+-1*1+1*1+1*1=2
W[11,6]=-1*-1+-1*-1+1*1+1*1=4
W[11,7]=-1*1+-1*1+1*1+1*1=0
W[11,8]=-1*-1+-1*1+1*-1+1*1=0
W[11,9]=-1*-1+-1*1+1*-1+1*-1=-2
W[11,10]=-1*1+-1*-1+1*-1+1*-1=-2
W[11,11]=-1*-1+-1*-1+1*1+1*1=4
W[11,12]=-1*-1+-1*1+1*1+1*-1=0
W[11,13]=-1*1+-1*1+1*1+1*-1=-2
W[11,14]=-1*-1+-1*1+1*-1+1*1=0
W[12,0]=-1*-1+1*1+1*1+-1*1=2
W[12,1]=-1*1+1*1+1*1+-1*-1=2
W[12,2]=-1*-1+1*-1+1*-1+-1*1=-2
W[12,3]=-1*-1+1*-1+1*-1+-1*1=-2
W[12,4]=-1*1+1*-1+1*-1+1*1=-2
W[12,5]=1*-1+1*1+1*1+-1*1=2
W[12,6]=-1*-1+1*-1+1*1+-1*1=0
W[12,7]=1*1+1*1+1*1+-1*1=0
W[12,8]=-1*-1+1*1+1*-1+-1*1=0
W[12,9]=1*-1+1*1+1*-1+-1*-1=2
W[12,10]=-1*1+1*-1+1*-1+1*1=-2
W[12,11]=1*-1+1*-1+1*1+-1*1=0
W[12,12]=-1*-1+1*1+1*1+-1*-1=4
W[12,13]=1*1+1*1+1*1+-1*-1=2
W[12,14]=-1*-1+1*1+1*-1+-1*1=0
W[13,0]=1*1+1*1+1*1+-1*1=0
W[13,1]=1*1+1*1+1*1+-1*-1=4
W[13,2]=1*1+1*-1+1*-1+-1*1=-4
W[13,3]=1*-1+1*-1+1*-1+-1*1=-4
W[13,4]=1*1+1*-1+1*-1+-1*-1=0
W[13,5]=1*-1+1*1+1*1+-1*1=0
W[13,6]=1*-1+1*-1+1*1+-1*1=-2

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

W[13,7]=1*1+1*1+1*1+-1*1=2　　W[13,8]=1*-1+1*1+1*-1+-1*1=-2
W[13,9]=1*-1+1*1+1*-1+-1*-1=0　　W[13,10]=1*1+1*-1+1*-1+-1*-1=0
W[13,11]=1*-1+1*-1+1*1+-1*1=-2　　W[13,12]=1*-1+1*1+1*1+-1*-1=2
W[13,13]=1*1+1*1+1*1+-1*-1=4　　W[13,14]=1*-1+1*1+1*-1+-1*1=-2
W[14,0]=-1*-1+1*1+-1*1+1*1=2　　W[14,1]=-1*1-1*1+-1*1+1*-1=-2
W[14,2]=-1*-1+1*-1+-1*-1+1*1=2　　W[14,3]-1*-1+1*1+-1*-1+1*1=2
W[14,4]=-1*1+1*-1+-1*-1+1*-1=-2　　W[14,5]=-1*-1+1*1+-1*1+1*1=2
W[14,6]=-1*-1+1*-1+-1*1+1*1=0　　W[14,7]=-1*1+1*1+-1*1+1*1=0
W[14,8]=-1*-1+1*1+-1*-1+1*1=4　　W[14,9]=-1*-1+1*1+-1*-1+1*-1=2
W[14,10]=-1*1+1*-1+-1*-1+1*-1=-2　　W[14,11]=-1*-1+1*-1+-1*1+1*1=0
W[14,12]=-1*-1+1*1+-1*1+1*-1=0　　W[14,13]=-1*1+1*1+-1*1+1*-1=-2
W[14,14]=-1*-1+1*1+-1*-1+1*1=4

由上可算出权矩阵如下：

$$W=\begin{pmatrix}
4 & 0 & 0 & 0 & -4 & 4 & 2 & 2 & 2 & 0 & -4 & 2 & 2 & 0 & 2\\
0 & 4 & -4 & -4 & 0 & 0 & -2 & 2 & -2 & 0 & 0 & -2 & 2 & 4 & -2\\
0 & -4 & 4 & 4 & 0 & 0 & 2 & -2 & 2 & 0 & 0 & 2 & -2 & -4 & 2\\
0 & -4 & 4 & 4 & 0 & 0 & 2 & -2 & 2 & 0 & 0 & 2 & -2 & -4 & 2\\
-4 & 0 & 0 & 0 & 4 & -4 & -2 & -2 & -2 & 0 & 4 & -2 & -2 & 0 & -2\\
4 & 0 & 0 & 0 & -4 & 4 & 2 & 2 & 2 & 0 & -4 & 2 & 2 & 0 & 2\\
2 & -2 & 2 & 2 & -2 & 2 & 4 & 0 & 0 & -2 & -2 & 4 & 0 & -2 & 0\\
2 & 2 & -2 & -2 & -2 & 2 & 0 & 4 & 0 & -2 & -2 & 0 & 0 & 2 & 0\\
2 & -2 & 2 & 2 & -2 & 2 & 0 & 0 & 4 & 2 & -2 & 0 & 0 & -2 & 4\\
0 & 0 & 0 & 0 & 0 & 0 & -2 & -2 & 2 & 4 & 0 & -2 & 2 & 0 & 2\\
-4 & 0 & 0 & 0 & 4 & -4 & -2 & -2 & -2 & 0 & 4 & -2 & -2 & 0 & -2\\
2 & -2 & 2 & 2 & -2 & 2 & 4 & 0 & 0 & -2 & -2 & 4 & 0 & -2 & 0\\
2 & 2 & -2 & -2 & -2 & 2 & 0 & 0 & 0 & 2 & -2 & 0 & 4 & 2 & 0\\
0 & 4 & -4 & -4 & 0 & 0 & -2 & 2 & -2 & 0 & 0 & -2 & 2 & 4 & -2\\
2 & -2 & 2 & 2 & -2 & 2 & 0 & 0 & 4 & 2 & -2 & 0 & 0 & -2 & 4
\end{pmatrix} \tag{9-9}$$

相应的阈值，θ，由式(9-10)计算，结果如下：

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication: luzhenbo@yahoo.com.cn

$$\theta=\begin{bmatrix}6\\-2\\2\\2\\-6\\6\\3\\1\\5\\2\\-6\\3\\3\\-2\\5\end{bmatrix} \tag{9-10}$$

注意:如果输入模式和训练模式完全匹配时,网络输出也完全正确:

输入:0100100100100 10
输出:0100100100100 10

输入:1100010111001 11
输出:1100010111001 11

输入:1100011100011 10
输出:1100011100011 10

输入:1011011110010 01
输出:1011011110010 01

如下所示,我们将一个输入矢量做某些改动。那么这个改动的矢量能被联想成训练集中的最相近矢量。

输入:1111011110110 01
输出:1011011110010 01

输入模式和输出模式将在 9.3 节中进行讨论。注意到:当图 9.3(a) 的输入是图 9.2(a) 的改动版本时,网络输出也能够完全正确。如图 9.3(b) 。

然而当输入一个和训练集中所有矢量都截然不同的矢量时,它就不能映射到训练矢量集中的任何矢量。这种情况描述如下:

输入:0101011111011 01
输出:1011011111011 01

注意输出不能和任何存储的模式相匹配,因此恢复失败。清单 9.1 是 LAM 网络的C++

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@hotmail.com

图 9.3 LAM 例子:输入(a) 和成功恢复的输出(b)

图 9.4 LAM 例子:输入(a) 和不能成功恢复的输出(b)

实现。

清单 9.1

```
/*************************************************************************
*                                                                        *
* LAM                                                                    *
*                                                                        *
*************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <math.h>

// FUNCTION PROTOTYPES

//void  ShowResults(int, long int,double temp, double alpha, double eta);
// DEFINES
#define MAXNEURONS 100      // MAX NUMBER OF NEURONS PER LAYER
#define MAXPATTERNS 80      // MAX NUMBER OF PATTERNS IN A TRAINING SET

class LAME
{

private:
 int W[MAXNEURONS][MAXNEURONS];          // WEIGHTS MATRIX
 int Thresh[MAXNEURONS];
 int TrnSet[MAXPATTERNS][MAXNEURONS];    //TRAINING SET
 int TstSet[MAXPATTERNS][MAXNEURONS];    //TRAINING SET
 int inVect[MAXNEURONS];
 int outVect[MAXNEURONS];
 int   Neurons;
 int   TrnPatterns;                      // # of training set patterns
 int   TstPatterns;                      // # of test set patterns
public:
 LAME(void);
 void  GetTrnSet(char *Fname);
 int   GetTstSet(char *Fname);
 void  Train();
 void  Run(int i);
 void  ShowWeights();
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and Cooperation: luzhenbo@yahoo.com.cn


```
 void  ShowThresholds();
 void  ShowInVect();
 void  ShowOutVect();
};

//---------------------------------------------------------------
// METHOD DEFINITIONS

LAME::LAME(){
Neurons=0;
TrnPatterns=0;
TstPatterns=0;
}

void LAME::GetTrnSet(char *Fname){
FILE  *PFILE;
int  i,j,k;

PFILE = fopen(Fname,"r");                    // batch
if (PFILE==NULL){
  printf("\nUnable to open file %s\n",Fname);
  exit(0);
  }
fscanf(PFILE,"%d",&TrnPatterns);
fscanf(PFILE,"%d",&Neurons);
for (i=0; i<TrnPatterns; i++) {
  for (j=0; j<Neurons; j++) {
    fscanf(PFILE,"%d",&k);
    TrnSet[i][j]=k;
    } /* endfor */
  } /* endfor */
 }

 int  LAME::GetTstSet(char *Fname){
   FILE  *PFILE;
   int i,j,k;
 PFILE = fopen(Fname,"r");                    // batch
 if (PFILE==NULL){
   printf("\nUnable to open file %s\n",Fname);
   exit(0);
   }
 fscanf(PFILE,"%d",&TstPatterns);
 for (i=0; i<TstPatterns; i++) {
   for (j=0; j<Neurons; j++) {
     fscanf(PFILE,"%d",&k);
     TstSet[i][j]=k;
     } /* endfor */
   } /* endfor */
 return(TstPatterns);
 }

 void LAME::Train(){
   int i,j,p;
 //Calc weight matrix

 for (i=0; i<Neurons; i++) {
   for (j=0; j<Neurons; j++) {
     W[i][j]=0;
     for (p=0; p<TrnPatterns; p++) {
       W[i][j] += (2*TrnSet[p][i]-1) *(2*TrnSet[p][j]-1);
       } /* endfor */
     } /* endfor */
   } /* endfor */

 //Calc Thresholds
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/lu-zhenbo2
Communication & Cooperation: lu_zhenbo@hotmail.com.cn


```
for (i=0; i<Neurons; i++) {
  for (j=0; j<Neurons; j++) {
     Thresh[i] += W[i][j];
     } /* endfor */
  Thresh[i]=Thresh[i]/2;
   } /* endfor */
}

 void LAME::Run(int tp){
   int i,j;
   int RawOutVect[MAXNEURONS];
 for (i=0; i<Neurons; i++) {
   inVect[i]=TstSet[tp][i];
   } /* endfor */
 for (i=0; i<Neurons; i++) {
    RawOutVect[i] = 0;
    for (j=0; j<Neurons; j++) {
      RawOutVect[i]+=W[i][j] * inVect[j];          //Calc Raw output vect
      } /* endfor */
    } /* endfor */
 //apply threshold
 for (i=0; i<Neurons; i++) {
   if (RawOutVect[i]>Thresh[i]) {
     outVect[i]=1;
   } else {
     outVect[i]=0;
   } /* endif */
   } /* endfor */

 }
 void LAME::ShowWeights(){
    Int i,j;
 for (i=0; i<Neurons; i++) {
    for (j=0; j<Neurons; j++) {
       printf("%d ",W[i][j]);
       } /* endfor */
    printf("\n");
    } /* endfor */
 }
 void LAME::ShowThresholds(){
    int i;
 for (i=0; i<Neurons; i++) {
   printf("%d ",Thresh[i]);
    } /* endfor */
 printf("\n");
 }

 void LAME::ShowInVect(){
    Int i;
 printf("IN:  ");
 for (i=0; i<Neurons; i++) {
    printf("%d ",inVect[i]);
    } /* endfor */
 printf("\n");
 }

 void LAME::ShowOutVect(){
    int i;
 printf("OUT: ");
 for (i=0; i<Neurons; i++) {
    printf("%d ",outVect[i]);
    } /* endfor */
 printf("\n");
 }
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
//-------------------------------------------------------------------

LAME  LAM;

/*****************************************************************
*                                                                *
* MAIN                                                           *
*****************************************************************/
int main(int argc, char *argv[])
{
  int TstSetSize;
  int i;
if (argc>2) {
  LAM.GetTrnSet(argv[1]);
  TstSetSize=LAM.GetTstSet(argv[2]);
  LAM.Train();
  LAM.ShowWeights();
  for (i=0; i<TstSetSize; i++) {
    LAM.Run(i);                          //Evaluate ith test pattern
    printf("\n");
    LAM.ShowInVect();
    LAM.ShowOutVect();
  } /* endfor */
  }
 else {
  printf("USAGE: LAM TRAINING_FILE TEST_FILE\n");
  exit(0);
  }


 return 0;
 }
```

9.3 Hopfield 网络

Hopfield 网络[Hopfield,1982;1984]是一个单层回归自联想记忆网络。它的处理单元(神经元)是全连的,并且都能作为一个联想记忆器。Hopfield 的贡献得到了相当的重视。因为他将记忆用能量函数(Liapunov)来表示,并且他在每个处理单元(神经元)中引入了异步工作方式。

除了它们的自适应特性以外,神经网络的另一个显著特点就是:许多已有的结构,不管是信息处理单元还是信息存储单元,它们都是并行工作的。因此,所有的处理单元(神经元)都是同步(同时)进行运算。同时,它们的结构也被设计成并行分布记忆。正因为如此,所有的存储单元(权值)一起才代表有意义的信息。对于存储单元和处理单元,这些特点能够提供很大的容错性。随着适合互连、独立、并行的基本处理器的硬件的可实现性的提高,这些神经网络模型正重新受到重视。

从信息处理的角度看,Hopfield 网络是非常有趣的。和通常的神经网络模型一样,它的确支持并行分布信息存储,但是信息处理并不是真的以并行方式进行进行。处理单元(神经元)是以异步方式执行运算。这样可使网络避免在整个网络中必须传递同步信号所遇到的困难。另一个优点是:通常的神经网络要求在所有的时间里,每一个处理单元都要有整个网络的全局信息,而 Hopfield 网络就减少了这种要求。

对于离散的 Hopfield 网络,处理单元或取值二值(1/0)或取值双值(1/-1)。我们把有 N 个离散的二值序列构成的模式 u_i 输入到网络时,就能够产生一个系统的状态矢量。每个处理

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/lzhenbo1982
Communication & Cooperation: luzhenbo@yahoo.com.cn

单元(神经元)都和其它的另一个处理单元相连,从这种意义上说,网络是全互联的。权值,以T_{ij}表示,是对称的(示于等式 11);并且神经元和自己不相连(示于等式 12):

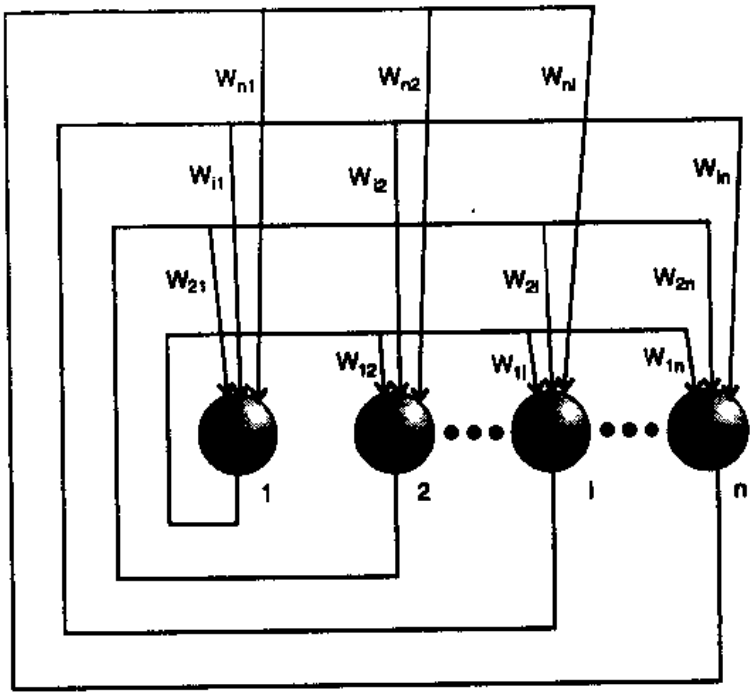


图 9.5 有 η 个神经元的 Hopfield 网络的结构

$$T_{ij} = T_{ji} \tag{9-11}$$

$$T_{ii} = 0 \tag{9-12}$$

通常情况下的具有 N 个神经元的网络结构示于图 9.5。图 9.6(a)给出了有 3 个神经元的 Hopfield 网络,图 9.6(b)给出了有 4 个神经元的 Hopfield 网络。在图 9.7(a)和图 9.7(b)中,我们给出了相对于这两种神经网络的能量流动立方体,以便于用来解释在这两种网络中的可能状态改变。

网络的状态就是由兴奋级别和有序处理单元的状态所组成的矢量。状态有一个联想能量(Liapunov)函数,由下式给出。

$$E = -\frac{1}{2}\sum_{j}\sum_{\substack{i \\ i\neq j}} T_{ji}u_j u_i \tag{9-13}$$

上式中,T_{ij}是从单元 i 到单元 j 的权值,并且 u_i 是网络中第 i 单元的输出。作为一个迭代网络,神经元被容许一次一个地进行更新直到收敛。当能量函数达到最小值并且再计算时没有神经元状态改变,那么我们就认为网络已经收敛。我们可以看到,当按规定方式进行更新时,Hopfield 网络总是收敛到最小能量状态。并且,一旦进入这种状态,任一神经元的状态都不会改变。

图 9.8 画出了具有几个预存的全局极小值状态的二维能量地形图;通常能量地形图是一个超空间,在此空间里由位于超立方体的各个顶角的二值状态矢量来表示各合理状态。离散的 Hopfield 联想记忆模型迭代过程为不断地从一顶角转向具有更小能量的另一顶角,直至稳定于具有最小能量的一个顶角。与之相对应,连续的 Hopfield 联想记忆模型迭代过程,从一个起始状态到另一个全局极小值(预存状态)来穿越超空间,如图 9.8 所示。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication: luzhenbo@yahoo.com.cn

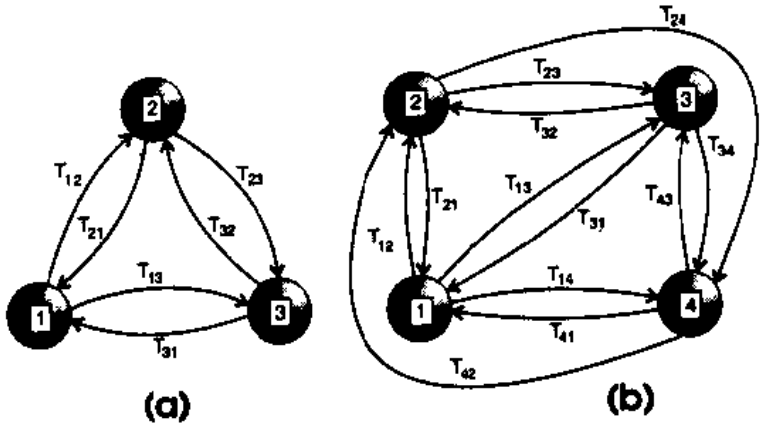


图 9.6 (a)具有 3 神经元和(b)具有 4 神经元的 Hopfield 网络的结构

注意到神经元的状态，可以被认为是短期记忆(STM)。当通过改变这些 u_i 的值时，联想记忆能够短时间记录并且保持任何模式。突触的强度(权值) T_{ij} 可以被认为是长期记忆(LTM)，因为网络能够通过改变这些权值来进行学习。

我们现在讨论一种方法，以这种方法来更新 Hopfield 网络中指定神经元。第 j 个神经元的激励是输入 u_i 和权值 T_{ij} 的乘积的和，如下式：

$$S_j = \sum_{\substack{i=1 \\ i\neq j}}^{n} u_i T_{ji} \tag{9-14}$$

第 j 个神经元的输出结果为：

$$u_j = \begin{cases} 1 & S_j \geq 0 \\ 0 & S_j < 0 \end{cases} \tag{9-15}$$

让我们再以能量函数的形式，仔细讨论一下等式(9-14)和等式(9-15)。尤其是我们必须指出：当我们以这种方式更新神经元时，网络必须收敛到具有最小能量的状态。Hopfield 网络是一迭代网络，并且一次只有一个神经元得到更新，所以我们从单个神经元更新的结果来考虑能量改变就足够了。

首先，单个神经元 j 改变后的能量由下式给出：

$$E = -\frac{1}{2} \sum_{\substack{i \\ i\neq j}} T_{ij} u_i u_j \tag{9-16}$$

上式的和只对 i 进行，所以我们可重新书写等式(9-16)如下式：

$$E_j = u_j \left(-\frac{1}{2} \sum_{\substack{i \\ i\neq j}} T_{ij} u_i \right) \tag{9-17}$$

因此，当神经元 j 从先前值 $u_j^{更}$ 更新到一个新值 $u_j^{新}$，对能量改变就是 ΔE_j：

$$\Delta E_j = E_j^{新} - E_j^{旧} \tag{9-18}$$

这样将等式(9-17)代入上式，我们得到

$$E_j = u_j^{新} \left[-\frac{1}{2} \sum_{\substack{i \\ i\neq j}} T_{ij} u_i \right] - u_j^{旧} \left[-\frac{1}{2} \sum_{\substack{i \\ i\neq j}} T_{ij} u_i \right] \tag{9-19}$$

$$\Delta E_j = (u_j^{新} - u_j^{旧}) \left[-\frac{1}{2} \sum_{\substack{i \\ i\neq j}} T_{ij} u_i \right] \tag{9-20}$$

我们将 u_j 定义如下：

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xidian.edu.cn

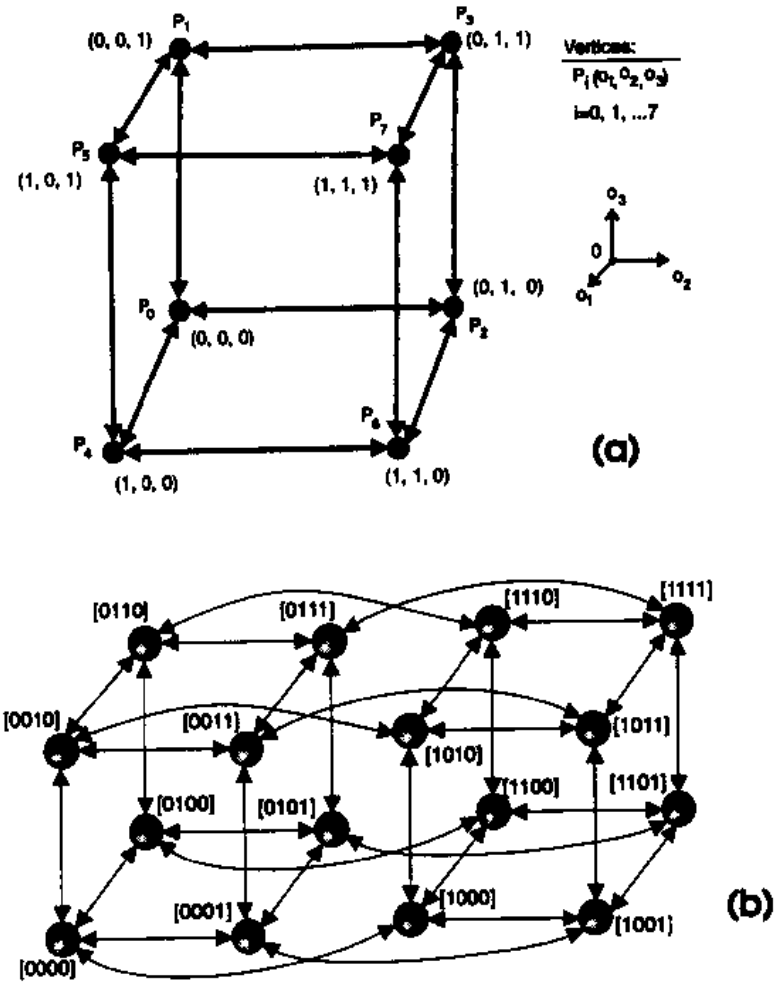


图 9.7　用来解释具有(a)3 神经元和(b)4 神经元的 Hopfield 网络状态改变的能量流的立方体

$$\Delta u_j = u_j^{新} - u_j^{旧} \tag{9-21}$$

于是我们得到

$$\Delta E_j = \Delta u_j \left(-\frac{1}{2} \sum_{\substack{i \\ i \neq j}} T_{ij} u_i \right) \tag{9-22}$$

我们将上式代入到等式(9-14),得到

$$\Delta E_j = -\frac{1}{2} \Delta u_j S_j \tag{9-23}$$

现在我们必须考虑以下三种情况:

1. 如果第 j 个神经元没有改变状态,那么 $\Delta u_j = 0$;因此由式(9-23),能量改变 $\Delta E_j = 0$。

2. 如果第 j 个神经元起始状态为 1,并且跃迁至 0 状态,那么

$$\begin{aligned} u_j^{旧} &= 1 \\ u_j^{新} &= 0 \\ \therefore \Delta u_j &= -1 < 0 \end{aligned} \tag{9-24}$$

应注意到由等式(9-15),当一个神经元的状态由 1 变为 0,那么 S_j 必须小于 0。因此,$\Delta u_j S_j$ 的乘积必须是正的,因此我们可以得出:

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xaut.edu.cn

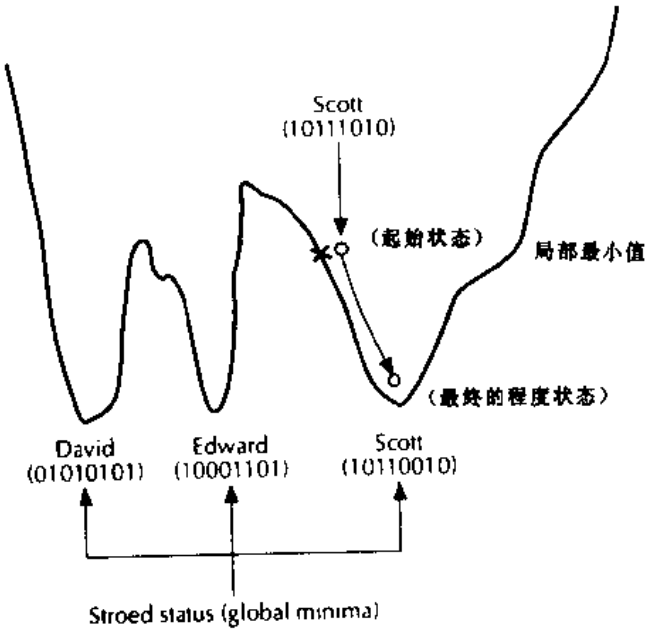


图 9.8 具有多个吸引子的 Hopfield 网络的二维能量地形图

$$\Delta E_j = 0 \tag{9-25}$$

3. 如果第 j 个神经元由起始状态 0 变为 1:

$$\begin{aligned} u_j^{旧} &= 1 \\ u_j^{新} &= 0 \\ \therefore \Delta u_j &= 1 > 0 \end{aligned} \tag{9-26}$$

同样由等式(9-15)我们可以看到:

$$S_j \geq 0 \tag{9-27}$$

那么 $\Delta u_j S_j$ 的乘积大于等于 0,由此我们得出:

$$\Delta E_j \leq 0 \tag{9-28}$$

因此,我们可以得出这样一个结论:对于 Hopfield 网络的任意一个神经元的可能状态改变,能量变化不是减小就是维持不变。更进一步,如果所有的神经元状态不再变化,那么就达到了一个稳定的全局最小值。

计算此网络,我们可随机选取神经元。当所有的神经元都被访问到,并且没有神经元再改变状态(见式 9-14 和 9-15)时,我们就认为网络已经收敛。

现在我们必须讨论训练网络的问题,以便于网络的极小值能和我们希望网络记忆的样本矢量集合相对应。给定网络需记忆的 m 个训练矢量集合 $\mathbf{A}^p$,下面的过程可用来固定网络的权值,以达到这样一个目的:每一预存的状态矢量能对应能量极小值,并且网络对应于每一个这样预存值都有一个稳定的状态:

$$\begin{aligned} T_{ji} &= \sum_{p=1}^{m} (2a_{pi} - 1)(2a_{pj} - 1) \qquad (i \neq j) \\ T_{ii} &= 0, \; T_{ij} = T_{ji} \end{aligned} \tag{9-29}$$

上式中 a_{pi} 是第 p 个训练矢量的第 i 个元素(分量),T_{ij} 是第 i 个和第 j 个神经元的权值。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xidian.edu.cn

9.4 Hopfield 网络的一个范例

我们现在讨论一个简单的例子。本章后面将对一个更为详细的例子的结果进行讨论。假定,我们希望在图 9.9 所示的 Hopfield 网络中存储两个模式。要存储的模式如下:

$$a^{(1)}=\begin{bmatrix}1\\0\\1\\0\end{bmatrix}\quad a^{(2)}=\begin{bmatrix}0\\1\\0\\1\end{bmatrix} \tag{9-30}$$

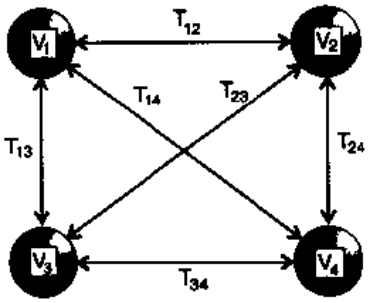


图 9.9 一个 4 神经元的 Hopfield 网络

因此训练等式变成:

$$T_{ij}=\sum_{p=1}^{2}(2a_i^{(p)}-1)(2a_j^{(p)}-1)\quad(i\neq j)$$
$$T_{ij}=0,T_{ij}=T_{ji} \tag{9-31}$$

于是,从上式我们可计算出权矩阵的每个元素如下:

$$\begin{aligned}T_{12}&=(2a_1^{(1)}-1)(2a_2^{(1)}-1)+(2a_1^{(2)}-1)(2a_2^{(2)}-1)\\&=(1)(-1)+(-1)1=-2\end{aligned} \tag{9-32}$$

$$\begin{aligned}T_{13}&=(2a_1^{(1)}-1)(2a_3^{(1)}-1)+(2a_1^{(2)}-1)(2a_3^{(2)}-1)\\&=(1)(1)+(-1)(-1)=2\end{aligned} \tag{9-33}$$

$$\begin{aligned}T_{14}&=(2a_1^{(1)}-1)(2a_4^{(1)}-1)+(2a_1^{(2)}-1)(2a_4^{(2)}-1)\\&=1(-1)+(-1)1=-2\end{aligned} \tag{9-34}$$

$$\begin{aligned}T_{23}&=(2a_2^{(1)}-1)(2a_3^{(1)}-1)+(2a_2^{(2)}-1)(2a_3^{(2)}-1)\\&=(-1)(1)+(1)(-1)=-2\end{aligned} \tag{9-35}$$

$$\begin{aligned}T_{24}&=(2a_2^{(1)}-1)(2a_4^{(1)}-1)+(2a_2^{(2)}-1)(2a_4^{(2)}-1)\\&=(-1)(-1)+(1)1=-2\end{aligned} \tag{9-36}$$

$$\begin{aligned}T_{34}&=(2a_3^{(1)}-1)(2a_4^{(1)}-1)+(2a_3^{(2)}-1)(2a_4^{(2)}-1)\\&=1(-1)+(-1)1=-2\end{aligned} \tag{9-37}$$

因此完整的权矩阵由式(9-38)给出:

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

$$T=\begin{bmatrix} 0 & -2 & 2 & -2 \\ -2 & 0 & -2 & 2 \\ 2 & -2 & 0 & -2 \\ -2 & 2 & -2 & 0 \end{bmatrix} \tag{9-38}$$

现在假设给定上面训练完的网络，我们输入一个矢量$[1110]^T$。这即意味着网络的初始状态为：

$$\begin{aligned} v_1 &= 1 \\ v_2 &= 1 \\ v_3 &= 1 \\ v_4 &= 0 \end{aligned} \tag{9-39}$$

请注意，a_i 通常表示输入训练模式，而 v_i 通常表示网络的状态。

如果神经元 v_2 首先被更新，那么我们将得到：

$$\begin{aligned} S_2 &= \sum_{j=1}^{4} T_{2j} v_j \\ &= (-2)(1)+(0)(1)+(-2)(1)+(2)(0) \\ &= -4<0 \qquad \therefore v_2=0 \end{aligned} \tag{9-40}$$

注意到 v_2 已经改变了状态。同时我们也应注意到，这时网络的状态和预存的模式$[1010]^T$相匹配。剩下的就只是对其它的每个神经元都处理一遍，直到没有状态改变。这可很容易地如下完成：

$$\begin{aligned} S_4 &= \sum_{j=1}^{4} T_{4j} v_j \\ &= (-2)(1)+(0)(1)+(-2)(1)+(2)(0) \\ &= -4<0 \qquad \therefore v_4=0 \end{aligned} \tag{9-41}$$

$$\begin{aligned} S_1 &= \sum_{j=1}^{4} T_{1j} v_j \\ &= (0)(1)+(-2)(0)+(2)(1)+(-2)(0) \\ &= 2>0 \qquad \therefore v_1=1 \end{aligned} \tag{9-42}$$

$$\begin{aligned} S_3 &= \sum_{j=1}^{4} T_{3j} v_j \\ &= (2)(1)+(-2)(0)+(0)(1)+(-2)(0) \\ &= 2>0 \qquad \therefore v_3=0 \end{aligned} \tag{9-43}$$

$$\begin{aligned} S_2 &= \sum_{j=1}^{4} T_{2j} v_j \\ &= (-2)(1)+(0)(0)+(-2)(1)+(2)(0) \\ &= -4<0 \qquad \therefore v_2=0 \end{aligned} \tag{9-44}$$

注意从式(9-41)到式(9-44)，所有的神经元都计算一遍，网络的状态$[1010]^T$没有改变，因此网络已收敛。

9.5 讨　论

正如我们所讨论的一样，Hopfield 网络可容易地进行单步训练。因此，网络的权值被预

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xaut.edu.cn

置,并且我们把 Hopfield 网络分类为固定权值的神经网络。与之相对应的有:有监督或无监督模型。在这些模型中,网络分别使用不同的学习算法来进行训练。

训练完成后,我们向网络输入一要识别的输入矢量。随机选取神经元来按照等式(9-14)、(9-15)来进行更新。当所有的神经元都遍历过并且没有发生变化时,网络就已收敛。注意第一个被更新的方向,能够影响网络最后收敛所达到的状态。网络更新也有采取并行(同步)方式的,但这种方式比串行(异步)方式更容易受到伪状态问题的困扰,下面详细论述此问题。对于 Hopfield 网络进行同步更新方式的详细讨论,见 Kung 于 1993 年的文献。

在 Hopfield 模型中,能量函数的每一个局部极小值一定是一个吸引子。吸引子就是一个平衡状态。当达到这样一个状态,那么网络就一直保此那种状态不变。Hopfield 网络可以完全在状态空间中进行描述,但是它的行为(Behavior)可以用被称作能量(也就是适当的 Liapunov 函数的值)的单个标量来表述。了解一个按内容存取的联想记忆模型,如 Hopfield 模型,对于一任意输入的模式怎样进行响应是非常重要的。给定一个输出模式,怎样才能保证网络向与输入模式最为相近的网络预存模式演化呢?这种网络的信息存储能量是多少呢(也就是有多少稳定的平衡状态(预存模式)能够同时存在呢)? Hopfield 和 Jank 于 1985 年由经验得出:在保证回想记忆的误差不是非常严重的前提下,含有 N 个神经元的 Hopfield 网络中可预存 0.15N 个状态。

我们所遇到的一个困难,就是训练集合中的非正交模式能够耦合产生伪吸引子,也就是和训练矢量都不对应的稳定极小能量状态。输入模式的负模式也是一个吸引子,这种特性更加剧了这种状况。Kang 于 1993 年提出的这种特性,可由等式中的对称性很容易得到。当 $a_{pi}=a_{pj}$ 时,t_{ji} 增加;而 $a_{pi}\neq a_{pj}$ 时,t_{ji} 减小。因此,t_{ij} 权值只对 a_{pi} 和 a_{pj} 的比较值敏感(而它们相应的值就被丢失)。这也必然导致负的或相反的模式被储存。也就是说,对于每一个预存模式,同时有一个不正确相反的模式也被预存。对于图像的负片(底片),对于这个特征引起的不变性来说,接受这种特征是令人感兴趣的,甚至可能令人非常欢迎这个特征。可是非常遗憾的是这些负模式也能够和其它的模式耦合来产生伪极小值,这就近一步阻碍了试图对输入模式进行正交。这些问题以及 Hopfield 网络的相对较低的存储容量,选取 Hopfield 网络来进行模式识别似乎不太令人信服。Xun-Jing 于 1992 年的一份研究表明:对 Hopfield 网络、Hamming 网络以及神经认知机三种网络进行比较,对于字符识别来说,Hopfield 网络不适合。

和本书中讨论的许多用于模式识别的其它神经网络相比,Hopfield 网络在文献中出现的次数并不多。1991 年 Gee 等将 Hopfield 网络用于对旋转手写数字识别问题上。在这个应用中,它将图形看成是沿着字符轮廓的一系列 n 个点。1987 年,Tank 和 Hopfield 对连续的 Hopfield 模型做了改进用于解决优化问题,其中包含有将实验或者未知的字符图形中的这些特征点映射到响应的样本模式或模板上。

1991 年,Chen 等将二值(0/1)Hopfield 网络改进后应用于数字识别问题上。在这个改进中使用了单误差检测,单误差校验(SEDSEI)编码来提高 Hopfield 网络的训练和识别精度。SEDSEC 编码经常用于硬件存储器中以提高可靠性。据 Chen 等的报道,这种改进的结果是激动人心,并且令人吃惊的。对于这类数据集合来说,通常的 Hopfield 网络只能达到 5% 的识别精度,而改进后的网络能达到并超过 90% 的识别精度。SEDSEC 编码是通过在原始图形的信息位后加上校验位来完成的。这些校验位是局部的,但同时它们也是跨越信息位与其自身的重叠运算符。对于长度为 n 的信息比特序列的校验位的数目可以得到。例如一个字节 8 比特的信息序列需要 4 个校验比特。校验位的计算是非常快的,它只需使用异或(XOR)运算符。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

网络的结构是蜂窝状的(六角形的),网络不再是全互联。神经元只和它最相近的三个神经元相连。在网络训练过程中,通过异或(XOR)运算产生了校验比特(位)。在回忆过程中找出并且改正不正确的输入信息。

9.6 位 图 范 例

假设我们希望构造一个联想记忆模型,这个模型能够识别出图 9.10 所示 4 个数字。即使给定的要识别的数字以相对简单的表述方法表述,也需要一个 56 神经元的网络,并且也需要进行所需计算的自动元素。清单 9.2 包含有运行二值 Hopfield 模型所需的程序代码。

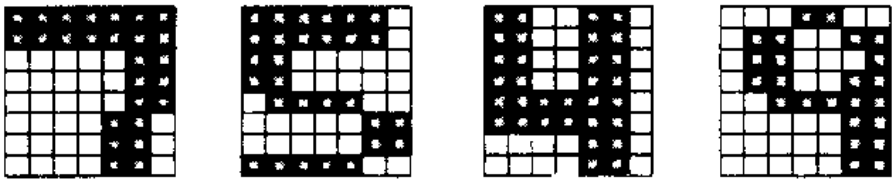


图 9.10 Hopfield 网络输入模式的范例

清单 9.2

```
/*************************************************************************
*                                                                        *
* Binary Hopfield Network                                                *
*                                                                        *
*************************************************************************/

#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
//#include <dos.h>

//------------------------------------------------------------------------

// DEFINES
#define MAXNEURONS    64
#define MAXPATTERNS   10
#define MAXITER       600000
#define TRUE          1
#define FALSE         0

//------------------------------------------------------------------------

// FUNCTION PROTOTYPES
// network fns
void InitNetRand(void);                    // Scramble net state
void LoadTrainingSet(char *Fname);         // Get training set from file
void TrainNet();                           // Train net to recognize patterns
void RunNet(void);                         // Update net til convergence or MAXITER
int UpdateNeuron(int j);                   // Update jTH neuron
                                           // Return TRUE if state changed

void LoadUserPattern(char *Fname);
int QueryAllClean(void);                   // Return TRUE if all neurons were visited
void SetAllToDirty(void);                  // Set all neurons to NOT visited
// utility fns
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
void Initialize(int cnt, char *Name);              // housekeeping
void DisplayPatVect(int V[MAXNEURONS]);            // show Net/Pattern (human-eye view)
void SavePatVect(int V[MAXNEURONS],                // store Net/Pattern (human-eye view)
                 char *txt, int i);                // to the archive file
void DisplayWeights(void);                         // show the weight matrix
void DisplayPatterns(void);                        // Display all trained patterns
int  QueryUserInt(char *msg);
void KeyWait(void);
void KeyWait2(char *txt);
int  random(int N);
//-------------------------------------------------------------------------

// GLOBALS
int PatVec[MAXNEURONS];                            // Pattern Vector
int PatMatrix[MAXPATTERNS][MAXNEURONS];            // Pattern Vector
int NEURON[MAXNEURONS];                            // Network
int T[MAXNEURONS][MAXNEURONS];                     // Weight matrix
int NumNeurons;                                    // Actual size of net
int NumPatterns;                                   // Actual number of patterns
int PatternX;                                      // X-dimension for human viewing
int PatternY;                                      // Y-dimension for human viewing
int Dirty[MAXNEURONS];                             // TRUE if neuron has not been updated
                                                   // FALSE otherwise

FILE *ARCHIVE;

//-------------------------------------------------------------------------

int  random(int N){
int x;
x=N*rand();
return (x/RAND_MAX);
}

void DisplayPatVect(int V[MAXNEURONS]){
  int x,y,indx;
indx=0;
for (y=0; y<PatternY; y++) {
  for (x=0; x<PatternX; x++) {
    if (V[indx]==1) {
      printf("X");
    } else {
       printf(".");
     } /* endif */
     indx++;
     } /* endfor */
   printf("\n");
   } /* endfor */
  printf("\n");
}

void SavePatVect(int V[MAXNEURONS], char *txt, int i) {
  int x,y,indx;
  indx=0;
  fprintf(ARCHIVE,"\n");
  for (y=0; y<PatternY; y++) {
    for (x=0; x<PatternX; x++) {
      if (V[indx]==1) {
        fprintf(ARCHIVE,"X");
      } else {
        fprintf(ARCHIVE,".");
      } /* endif */
      indx++;
      } /* endfor */
    fprintf(ARCHIVE,"\n");
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
    } /* endfor */
  fprintf(ARCHIVE,"\n%s ",txt);
  if (i>=0) fprintf(ARCHIVE,"%d ",i);
  fprintf(ARCHIVE,"\n\n ");
}

void DisplayWeights(){
  int i,j;
fprintf(ARCHIVE,"WEIGHTS:\n");
for (i=0; i<NumNeurons; i++) {
  fprintf(ARCHIVE,"[");
  for (j=0; j<NumNeurons; j++) {
    fprintf(ARCHIVE, " %d",T[j][i]);
    } /* endfor */
  fprintf(ARCHIVE,"]\n");
  } /* endfor */
}

void DisplayPatterns() {
  int i,p;
for (p=0; p<NumPatterns; p++) {
  for (i=0; i<NumNeurons; i++) {
    PatVec[i] =PatMatrix[p][i];
    } /* endfor */
  DisplayPatVect(PatVec);                    // show 1st training pattern
  SavePatVect(PatVec, "Training Pattern", p+1);
  printf("\n\nTraining Pattern %d of %d\n\n",p+1,NumPatterns);
  KeyWait();
  } /* endfor */
}

int  QueryUserInt(char *msg){
int rv;
printf("Enter %s ==> ",msg);
scanf("%d",&rv);
return rv;
}

void KeyWait(void){
printf("Press any key to continue.\n");
while (!kbhit()) { } /* endwhile */
getch();
}

void KeyWait2(char *txt){
printf("\n\n%s\n",txt);
KeyWait();
}

void InitNetRand() {
  int i,r;

fprintf(ARCHIVE,"Creating test pattern\n");
srand(5);
//randomize();

for (i=0; i<NumNeurons; i++) {
  r=random(100);
  if (r >= 50) {
    NEURON[i]=0;
    }
  else {
    NEURON[i]=1;
    } /* endif */
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@hotmail.com


```
  } /* endfor */

}

void LoadTrainingSet(char *Fname) {
  int pat,j, InVal;
  FILE *PATTERNFILE;

printf("Loading training set from default file: %s\n",Fname);
fprintf(ARCHIVE,"Loading training set from default file: %s\n",Fname);
PATTERNFILE = fopen(Fname,"r");
if (PATTERNFILE==NULL){
  printf("Unable to open default training Set file: %s",Fname);
  exit(0);
  }

//printf("\n");
fscanf(PATTERNFILE,"%d",&NumNeurons);          // Get number of neurons
fscanf(PATTERNFILE,"%d",&NumPatterns);         // Get number of patterns
fscanf(PATTERNFILE,"%d",&PatternX);            // X-dimension for human viewing
fscanf(PATTERNFILE,"%d",&PatternY);            // Y-dimension for human viewing
printf("%d Patterns Loaded\n",NumPatterns);
fprintf(ARCHIVE,"%d Patterns Loaded\n",NumPatterns);
for (pat=0; pat<NumPatterns; pat++) {
  for (j=0; j<NumNeurons; j++) {
    fscanf(PATTERNFILE,"%d",&InVal);
    PatMatrix[pat][j] =InVal;
    } // endfor
  } // endfor
fclose(PATTERNFILE);
}
void LoadUserPattern(char *Fname) {
  int j, InVal;
FILE *PATTERNFILE;

printf("Loading pattern from file: %s\n", Fname);
fprintf(ARCHIVE,"Loading pattern from file: %s\n", Fname);
PATTERNFILE = fopen(Fname,"r");
if (PATTERNFILE==NULL){
  printf("Unable to open file: %s",Fname);
  exit(0);
  }

printf("\n");
for (j=0; j<NumNeurons; j++) {
  fscanf(PATTERNFILE,"%d",&InVal);
  NEURON[j] =InVal;
  } // endfor
fclose(PATTERNFILE);
}
void TrainNet(){
  int i,j,pat;
  int Sum;
for (i=0; i<NumNeurons; i++) {
  for (j=0; j<NumNeurons; j++) {
    Sum=0;
    for (pat=0; pat<NumPatterns; pat++) {
      Sum += (2*PatMatrix[pat][i]-1) * (2*PatMatrix[pat][j]-1);
      //Sum += PatMatrix[pat][i] * PatMatrix[pat][j];
      } /* endfor */
    T[j][i] = T[i][j] = Sum;
    } /* endfor */
  } /* endfor */
for (i=0; i<NumNeurons; i++) {                 // Get rid of the diagonal...
  T[i][i]=0;                                   //  ...so it doesn't cause trouble later
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
  } /* endfor */
}
int QueryAllClean() {
  int i;
for (i=0; i<NumNeurons; i++) {
  if (Dirty[i]==TRUE) return FALSE;
  } // endfor
return TRUE;
}


void SetAllToDirty() {
  int i;
for (i=0; i<NumNeurons; i++) {
  Dirty[i]=TRUE;
  } // endfor
}


int UpdateNeuron(int j) {
  int i;
  int Sum = 0;
  int OldState = NEURON[j];
for (i=0; i<NumNeurons; i++) {              // accumulate all contributions
  Sum += T[j][i] * NEURON[i];               // remember we set diagonal of matrix T to 0 ..
                                            //       .. so no need to tast for i==j
  } /* endfor */

if (Sum < 0) {
  NEURON[j] = 0;
  }
 else {
  if (Sum>0)
    NEURON[j] = 1;
  } /* endif */

if (NEURON[j] == OldState) {
  return 0;
  }
 else {
  return 1;
  } /* endif */
}


void RunNet(void) {
  int j;
  int Converged = FALSE;
  int ChngCount = 0;
  unsigned long int IterCount = 0;
  int ArchCnt=0;
SetAllToDirty();
while ( (!Converged) && (IterCount < MAXITER) ) {
  j = random(NumNeurons);                   //next updating neuron j);
  ChngCount += UpdateNeuron(j);             // increment if neuron changed state
  DisplayPatVect(NEURON);
    printf("RUNNING...  Iteration=%d \n",IterCount);
  if (ArchCnt>=9) {
   SavePatVect(NEURON, "Net output at iteration =", IterCount+1);
    ArchCnt=0;
     } else {
     ArchCnt++;
     } /* endif */
   Dirty[j] = FALSE;                        // Record that we've  covered this neuron
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@sina.com


```
  if (QueryAllClean()) {                    // Check if we hit all neurons at least once
    // here if we have hit all at least once
    //DisplayPatVect(NEURON);
    if (ChngCount == 0) {                   // Check if any neurons changed this pass
      // if we're here then were converged
      Converged = TRUE;
      printf("\nCONVERGED");
     SavePatVect(NEURON, "Net after convergence at iteration=", IterCount);
      }
     else {
      // if here then NOT converged so reinit for another pass
      SetAllToDirty();
      ChngCount=0;
      } /* endif */
    } /* endif */
  IterCount++;                              // Increment iteration counter
  } /* endwhile */
}


void Initialize(int cnt, char *Name) {      // housekeeping
  char TrnName[50];
  char TstName[50];
ARCHIVE = fopen("ARCHIVE.LST","w");
if (ARCHIVE==NULL){
  printf("Unable to open default Training Set file: ARCHIVE.LST");
  exit(0);
  }
if (cnt>1) {
  // Get test pattern from file specified by command line arg
  strcpy(TrnName,Name);
  strcpy(TstName,Name); strcat(TstName,".tst");
  TrnName[4]=0; strcat(TrnName,"n4.trn");
  LoadTrainingSet(TrnName);
  if (cnt==2) LoadUserPattern(TstName);
  }
 else {
  // Initialize net with random test pattern
  // no command line parms
  LoadTrainingSet("HOPNET1.TRN");           // Use default training set
  InitNetRand();                            // use random pattern
  } /* endif */

if (cnt>2) InitNetRand();                   // parm count >2 —ignore pat file & randomize
KeyWait();
}

int main(int argc, char *argv[]) {
   int i;
Initialize(argc, argv[1]);
DisplayPatVect(NEURON);                     // show net is set to test pattern
SavePatVect(NEURON, "Test Pattern", -1);
KeyWait2("TEST PATTERN");
DisplayPatterns();
TrainNet();
DisplayWeights();
RunNet();
fclose(ARCHIVE);
}
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

图 9.11 示出了网络(如图 9.10 所示那样进行训练完的)对数字 7 的畸变样本进行识别的过程。每当有 30 个神经元更新时,画一次网络的状态。注意,经过 150 次迭代网络就已达到稳定状态,此状态对应于已训练过的样本"7"。

注意到恢复(见图 9.11)是成功的,因为我们先前预存的模式之一被成功地恢复出来。与之相反的是,图 9.12 示出了网络收敛到相反的吸引子情况。

我们观察到经过 300 次迭代,试验样本就收敛到一个稳定状态。而且,我们也可以观察到这个稳定状态(吸引子)是训练样本数字"9"的相反样本。很明显我们的实验样本是令人生疑的,因为它和我们用以训练网络的输入样本相差得太多。但是,它只是用来说明 Hopfield 网络的训练特性。我们也不难找到其它一些收敛到伪吸引子的试验样本。

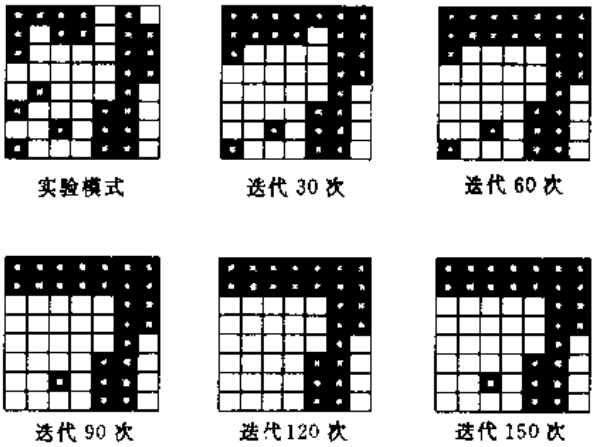


图 9.11 Hopfield 网络的收敛同时说明恢复成功

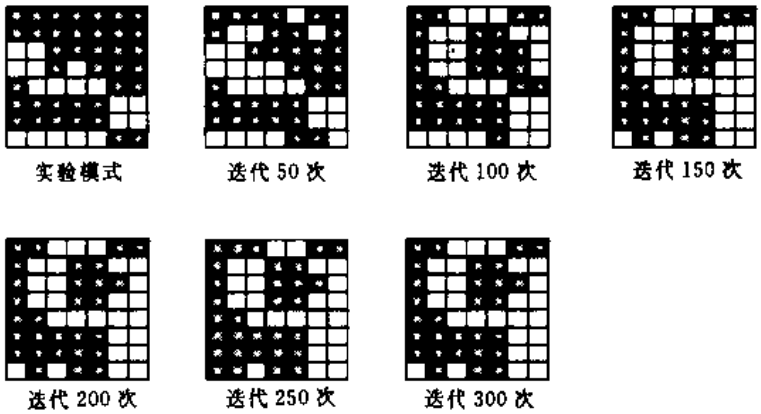


图 9.12 Hopfield 网络收敛到一相反吸引子

9.7 BAM 网络

双向联想记忆(BAM)预存了一系列联想对[Kosko ,1987;1988],像 Hopfield 网络一样,它们也具有反馈连接并且也需要许多次迭代才能稳定至平衡状态,所以它也是回归的。和

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

Hopfield 网络不一样的是,它们也可以是异联想的。图 9.13 示出了一个 BAM 网络的结构。

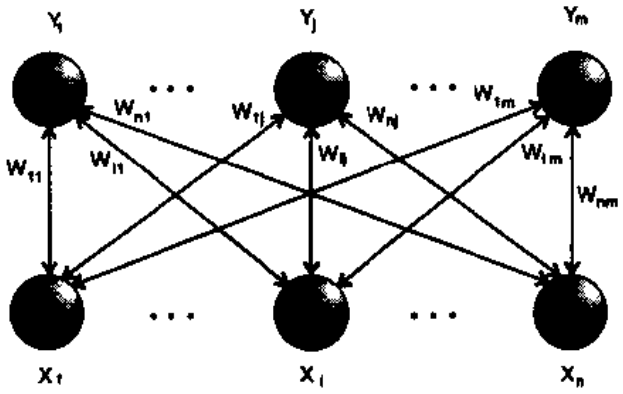


图 9.13 BAM 网络的结构

注意到网络含有二层,而每层可以包含不同数量的神经元。在 BAM 网络中,两层都不能被定义成固定的输入层或输出层,一个样本可以输入到两层中的任何一层。当网络达到平衡状态时,在理论上最后都应在另一层输出适应的相应模式。因此,我们将这两层分别定义成 X 层和 Y 层。注意到 X 层和 Y 层是全互连的,我们也应注意到在同一层的神经元之间没有连接。层之间的连接是双向的,如果我们将从 X 层到 Y 层的权值矩阵定义成 W,那么从 Y 层到 X 层的权值矩阵是 $\mathbf{W}$ 的转置矩阵,即 $\mathbf{W}^T$。网络的运转情况是:信息在两层之间来回迭代传送,直到网络达到平衡状态,也就是说网络在前向过程($X\rightarrow T$)和反向过程($Y\rightarrow X$)中交替进行迭代,直到没有网络变化为止。

在前向过程中,Y 层的第 j 个神经元的输入计算如下:

$$net_j^{(y)}=\sum_{i=1}^{n} w_{ij}x_i \tag{9-45}$$

并且 X 层神经元的兴奋性如下式计算:

$$y_j=\begin{cases}1 & net_j^{(y)}>\theta_j\\ y_j & net_j^{(y)}>\theta_j\\ -1 & net_j^{(y)}>\theta_j\end{cases} \tag{9-46}$$

在反向过程中 X 层神经元计算如下:

$$net_j^{(x)}=\sum_{j=1}^{n} w_{ij}x_i \tag{9-47}$$

并且 X 层神经元的兴奋性如下式计算:

$$x_i=\begin{cases}1 & net_i^{(x)}>\theta_j\\ x_i & net_i^{(x)}=\theta_j\\ -1 & net_i^{(x)}=\theta_j\end{cases} \tag{9-48}$$

BAM 网络更新及完成收敛的整个过程,示于图 9.14。

BAM 网络的训练需要一训练系列,此系列含有如下式那样的矢量联想对:

$$\{(\mathbf{a}^{(1)},\mathbf{b}^{(1)}),(\mathbf{a}^{(2)},\mathbf{b}^{(2)}),\cdots,(\mathbf{a}^{(p)},\mathbf{b}^{(p)})\} \tag{9-49}$$

上式中:

$$\mathbf{a}^{(p)}=a_1^{(p)},a_2^{(p)},\cdots,a_n^{(p)} \tag{9-50}$$

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

以及：

$$\mathbf{b}^{(p)} = b_1^{(p)}, b_2^{(p)}, \cdots, b_n^{(p)} \tag{9-51}$$

对于双极性数据权值可按式(9-52)那样计算：

$$W_{ij} = \sum_p a_i^{(p)} b_j^{(p)} \tag{9-52}$$

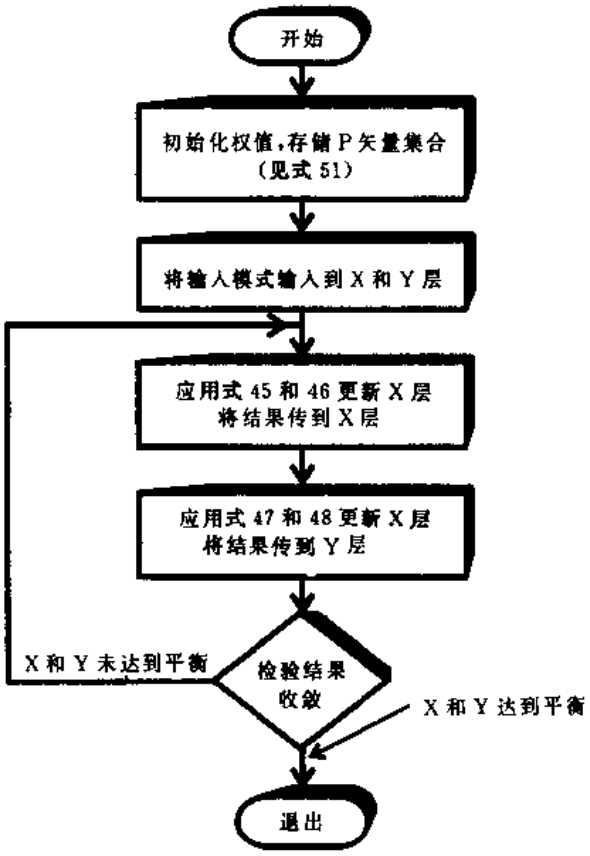


图 9.14 BAM 网络的流程图

如果数据是二进制的权值，将如下式计算：

$$W_{ij} = \sum_p (2a_i^{(p)} - 1)(2b_j^{(p)} - 1) \tag{9-53}$$

现在我们举一个例子，用以解释 BAM 网络的工作情况。

9.8 一个 BAM 网络范例

图 9.15 给出了二个位图模式，以及我们希望用于联想相应模式的双极性数码，相应的 BAM 网络结构由图 9.16 给出。

因此我们有两组联想对$\{(\mathbf{a}^{(1)}, \mathbf{b}^{(1)}), (\mathbf{a}^{(2)}, \mathbf{b}^{(2)})$，联想对的具体取值如下：

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

图 9.15 BAM 网络输入模式的例子

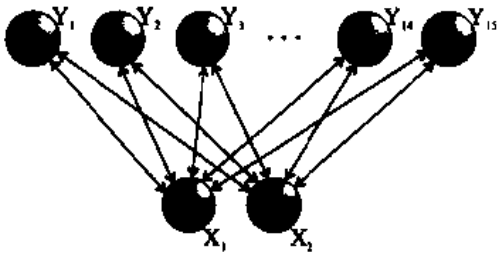


图 9.16 具有 2 个层神经元以及 15 个层神经元的 BAM 网络例子

$$\mathbf{a}^{(1)}=\begin{pmatrix}1\\1\\1\\-1\\1\\-1\\-1\\1\\-1\\-1\\1\\-1\\1\\1\\1\end{pmatrix}^T \quad \mathbf{b}^{(1)}=\begin{pmatrix}-1\\1\end{pmatrix}^T \quad \mathbf{a}^{(2)}=\begin{pmatrix}1\\-1\\1\\1\\-1\\1\\1\\1\\1\\1\\-1\\1\\1\\-1\\1\end{pmatrix}^T \quad \mathbf{b}^{(2)}=\begin{pmatrix}1\\1\end{pmatrix}^T \tag{9-54}$$

相应的权值矩阵按式(9-53)计算结果如下：

More documents and datum download website
LuZhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

$$\mathbf{W}=\begin{pmatrix}-1&1\\-1&1\\-1&1\\1&-1\\-1&1\\1&-1\\1&-1\\-1&1\\1&-1\\1&-1\\-1&1\\1&-1\\-1&1\\-1&1\\-1&1\end{pmatrix}+\begin{pmatrix}1&1\\-1&-1\\1&1\\1&1\\-1&-1\\1&1\\1&1\\1&1\\-1&-1\\1&1\\1&1\\-1&-1\\1&1\\-1&-1\\1&1\end{pmatrix}=\begin{pmatrix}0&2\\-2&0\\0&2\\2&0\\-2&0\\2&0\\2&0\\0&2\\2&0\\2&0\\-2&0\\2&0\\0&2\\-2&0\\0&2\end{pmatrix} \tag{9-55}$$

现在，我们检验如下事实：即当我们将训练样本逐一输入到网络时，BAM 网络能正确重新生成相应的联想训练模式。这时，我们将 $\mathbf{b}^{(1)}$ 输入到 X 层，并且设定 Y 层为零（设定 BAM 网络的一层为零，事实上是为了不准对 BAM 网络的这一层提供任何联想对的信息。如果需要，我们可能对两层同时提供矢量的部分信息）。计算如下：

$$\begin{aligned}net^{(x)}&=(-1\quad 1)\begin{pmatrix}0&-2&0&2&-2&2&2&0&2&2&-2&2&0&-2&0\\2&0&2&0&0&0&0&2&0&0&0&0&2&0&2\end{pmatrix}\\&=(2\quad 2\quad 2\quad -2\quad 2\quad -2\quad -2\quad 2\quad -2\quad -2\quad 2\quad -2\quad 2\quad 2\quad 2)\end{aligned} \tag{9-56}$$

$$f(net^{(x)})=(1\quad 1\quad 1\quad -1\quad 1\quad -1\quad -1\quad 1\quad -1\quad -1\quad 1\quad -1\quad 1\quad 1\quad 1) \tag{9-57}$$

注意矢量 $\mathbf{a}^{(1)}$ 已按需要出现于 Y 层。

现在，为了说明 BAM 网络的双向特性，以及对噪声的容错性，我们将 $\mathbf{a}^{(2)}$ 模式进行微小的改动，然后输入到 Y 层，并且 X 层再次设置为零，那么输入到 X 层的矢量 $\mathbf{c}$ 如下所示：

$$\mathbf{c}=(1\quad \mathbf{0}\quad 1\quad -1\quad 1\quad -1\quad -1\quad 1\quad -1\quad -1\quad 1\quad -1\quad 1\quad \mathbf{1}\quad 1) \tag{9-58}$$

现在我们可以计算 $net^{(y)}$ 结果如下：

$$net^{(y)}=\mathbf{c}\mathbf{W} \tag{9-59}$$

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

$$net^{(y)} = (1 \quad 0 \quad 1 \quad -1 \quad 1 \quad -1 \quad -1 \quad 1 \quad -1 \quad -1 \quad 1 \quad -1 \quad 1 \quad 1 \quad 1) \begin{pmatrix} 0 & 2 \\ -2 & 0 \\ 0 & 2 \\ 2 & 0 \\ 2 & 0 \\ 0 & 2 \\ 2 & 0 \\ 2 & 0 \\ -2 & 0 \\ 2 & 0 \\ 0 & 2 \\ -2 & 0 \\ 0 & 2 \end{pmatrix} \tag{9-60}$$

$$net^{(y)} = (18 \quad 10) \tag{9-61}$$

将兴奋函数代入，然后我们得到：

$$b_{ij} < \frac{1}{(L-1+m)} \tag{9-62}$$

考虑到网络已经正确收敛，并在 X 层向我们提供了模式 $\mathbf{b}^{(2)}$。

参考书与文献

Amari, S. I., "Learning patterns and pattern sequences by self-organizing nets," *IEEE Trans. Comput.*, vol. 21, pp. 1197–1206, 1972.

Anderson, J. A., "A simple neural network generating an interactive memory," *Math. Biosci.*, vol. 14, pp. 197–220, 1972.

Anderson, J. A. and Bower, G. H., *Human Associative Memory*, V. H. Vincent, Washington, D.C., 1973.

Carpenter, G. A., "Neural network models for pattern recognition and associative memory," *Neural Networks*, vol. 2, pp. 243–257, 1989.

Chen, L. C., Fan, J. L., and Chen, Y. S., "A high speed modified Hopfield neural network and a design of character recognition system," *Proc. 1991 25th Annual 1991 IEEE Int. Carnahan Conf. Security Technology*, pp. 308–314, 1991.

Fausett, L., *Fundamentals of Neural Networks*, Prentice-Hall, Englewood Cliffs, NJ, 1994.

Freeman, J. A. and Skapura, D. M., *Neural Networks: Algorithms, Applications, and Programming Techniques*, Addison-Wesley, Reading, MA, 1991.

Gee, A. H., Aiyer, S. V. B., and Prager, R. W., "A Subspace Approach to Invariant Pattern Recognition using Hopfield Networks," Cambridge Univ. Eng. Dept. Tech. Report No. CUED/F-INFENG/TR62, 1991.

Grossberg, S., "Nonlinear neural networks: principles, mechanisms and architectures," *Neural Networks*, Vol. 1, pp. 17–61, 1988.

Hopfield, J. J., "Neural networks and physical systems with emergent collective computational abilities," *Proc. Natl. Sci.*, vol. 79, pp. 2554–2558, 1982.

Hopfield, J. J., "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Natl. Sci.*, vol. 81, pp. 3088–3092, 1984.

Hopfield, J. J., "Neuron computation decisions in optimizing problems," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.

Hopfield, J. J. and Tank, D. W., "Neural computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.

Kohonen, T., "Correlation matrix memories," *IEEE Trans. Comput.*, vol. C-21, no. 4, pp. 353–359, 1972.

Kohonen, T., *Associative Memory: A System-Theoretical Approach*, Springer-Verlag, Berlin, 1977.

Kohonen, T., *Content-Addressable Memories*, Springer-Verlag, Berlin, 1980.

Kosko, B., "Adaptive bidirectional associative memories," *Appl. Optics*, vol. 26, no. 23, pp. 4947–4959, 1987.

Kosko, B., "Bidirectional associative memories," *IEEE Trans. Syst. Man Cyber.*, vol. 18, no. 1, pp. 49–60, 1988.

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

Kung, S. Y., *Digital Neural Networks*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

Levine, D. S., *Introduction to Neural and Cognitive Modeling*, LEA Publishers, Hillside, NJ, 1990.

Nakano, K., "Associatron — A model of associative memory," *IEEE Trans. Syst. Man Cybern.*, pp. 380–388, 1972.

Pao, Y. H. and Merat, F. L., "Distributed associative memory for patterns," *IEEE Trans. Syst. Man Cybern.*, vol. 5, pp. 620–625, 1975.

Tank, D. W. and Hopfield, J. J., "Neural computation by concentrating information in time," *Proc. Natl. Acad. Sci.*, vol. 84, pp. 1896–1990, 1987.

Xuan-Jing, S., "A study of neural network application in handwritten digit recognition," *SPIE*, vol. 1766, pp. 684–689, 1992.

Zurada, J. M., *Introduction to Artificial Neural Systems*, West Publ., New York, 1992.

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation:

第十章 自适应共振理论(ART)

10.1 概 述

计算机科学的一个主要目的,就是研究一种智能机器,能在复杂的环境中、在没有任何帮助的情况下令人满意地工作。在无监督方式下学习的自适应共振理论(ART),就是设法实现这一目标的例子。

正如 Grossberg[1976]所研究的,自适应共振理论的结构是基于两层节点之间的自适应谐振反馈理论。Carpenter 和 Grossberg[1987a]所描述的 ART1 模型,是被设计用来对二进制输入模式进行聚类的。Carpenter 和 Grossberg[1987b]所提出的 ART2 模型,则是用于模拟输入模式的聚类。这些模型的进一步改进,即 ART3,也是由 Carpenter 和 Grossberg[1990]研究出来的。

在自适应共振理论中,以稳定的方式对任意输入模式序列进行自主学习和模式识别。在这种示例中,将自调节控制结构引入到了竞争学习模型中。

10.2 寻求聚类结构

模式可以看作是 N 维特征空间中几个点。我们期望模式能在分类隶属度或其它属性值的基础上有某些相识的方面,这些模式在模式空间中应是相互靠近的。这样,属于类别 C_i 的模式,应比属于类别 C_j 的模式聚集得更紧密。当然,在许多实际应用中,这些类别是相互重叠的。图 10.1 给出了在距离测度基础上形成的各种模式类的模式空间。

无监督学习算法,能设法识别出作为聚类中心的各种原型或样本。原型可以是实际的模式或一个位于对应类中心的合成模式矢量。K-均值算法(参阅第八章)、ISODATA 算法、矢量量化技术(参阅下节)(有关这些算法的详细讨论,参考 Pao[1989]或 Tou 和 Gonzales [1974]的文献)是用于聚类的判决理论方法。自适应共振理论结构,是一种在无监督学习领域用于聚类的一种神经网络方法。

在许多无监督学习的实际应用中,类别的数目可能是事先未知的。从面当决定两络结构时,也不能提前给出确切的输出节点数目。

10.3 矢 量 量 化

无监督学习的目标,就是把输入数据分到一定数目的有意义的类别中。解决这个问题最明显而直观的方法,就是根据欧式空间中的距离函数来对输入矢量进行聚类。在第八章中,我们已经提到了 K-均值算法,这种算法恰好是根据欧式距离进行聚类的方法。K-均值算法要求预先知道聚类中心的数目。然面,在实际应用中的无监督学习里,预先知道有意义的类别的数目以便对输入数据进行描述是不可能的。出于这个原因,则极其需要一种自适应调整聚类中

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

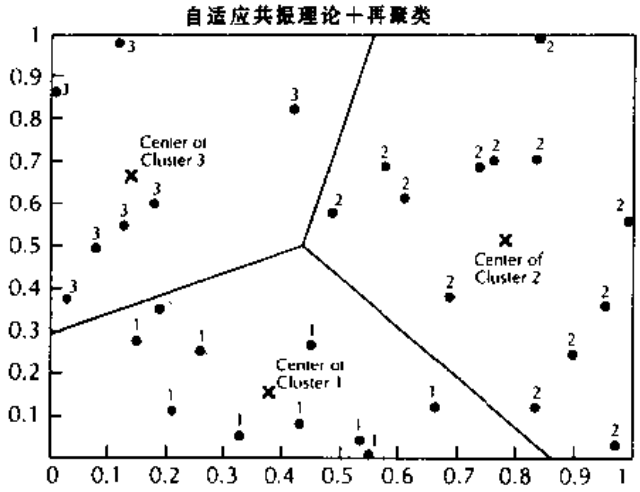


图 10.1 基于距离测度的聚类

心数目的方法。

本节讨论的 VQ 以及在本章的下一节中讨论的 ART,分别对聚类中心的动态分配问题给出了两种不同的方法。应注意的是,矢量量化是一种非神经的方法,而 ART 是一种神经网络方法。

矢量量化方法开始时并没有分配的类别(第一个模式会强迫创建出一个类以支持该模式)。以后,遇到每一个新的输入模式,系统将计算它和任何一个已分配的类别之间的欧式距离。如果指定第 P 个输入矢量为 $\mathbf{X}^{(p)}$以及第 j 个聚类中心为矢量$\mathbf{C}_j$,则欧式距离 d 可由等式(10-1)来计算:

$$d = \| \mathbf{X}^{(p)} - \mathbf{C}_j \| = \left[\sum_{i=1}^{N} (x_i^{(p)} - c_{ji})^2 \right]^{1/2} \tag{10-1}$$

其中 N 是 $\mathbf{X}$ 的维数,$\mathbf{C}$ 是矢量。

一旦现有模式和所有已分配的类别之间的距离已知,则和输入模式 $\mathbf{C}_k$ 最近的类别可由下式得出:

$$\| \mathbf{X}^{(p)} - \mathbf{C}_k \| < \| \mathbf{X}^{(p)} - \mathbf{C}_j \| = \begin{cases} j = 1, \cdots, M \\ j \neq k \end{cases} \tag{10-2}$$

式中,M 是已分配类别的数目。

在确定了最接近的类别之后,k 就已经确定了,距离 $\| \mathbf{X}^{(p)} - \mathbf{C}_k \|$ 必须和距离门限值 ρ 进行比较。在这种比较的基础上,会产生下面两种情况之一:

1. 当 $\| \mathbf{X}^{(p)} - \mathbf{C}_k \| < \rho$ 时,矢量 $\mathbf{X}^{(p)}$在允许的误差范围内。在这种情况下,该输入模式属于第 k 个类别。也就是说,如果用 S_k 表示第 k 个类别对应的模式集,则 $\mathbf{X}^{(p)} \in S_k$。

2. 当 $\| \mathbf{X}^{(p)} - \mathbf{C}_k \| > \rho$ 时,矢量 $\mathbf{X}^{(p)}$不在容许的误差范围内,从而不能分配到该类别中(即使这个类别是最近的)。在这种情况下,为 $\mathbf{X}^{(p)}$分配一个新的类别。

一旦一个模式属于某一类别,最新修改的类别的聚类中心就必须调整。新的聚类中心是通过求得所有成员矢量的平均值形成的(引入了 K-均值方法的思想)。更新聚类中心的过程,可通过下面的等式(10-3)给出:

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xaut.edu.cn

$$\mathbf{C}_k = \frac{1}{N_x} \sum_{x \in S_k} \mathbf{X} \tag{10-3}$$

在上面的等式中，k 表示现在输入模式所属的聚类中心，即便它是由这个模式的输入而新近创建的。

整个矢量量化过程在图10.2中说明。清单10.1给出了实现矢量量化方法的整个源程序。

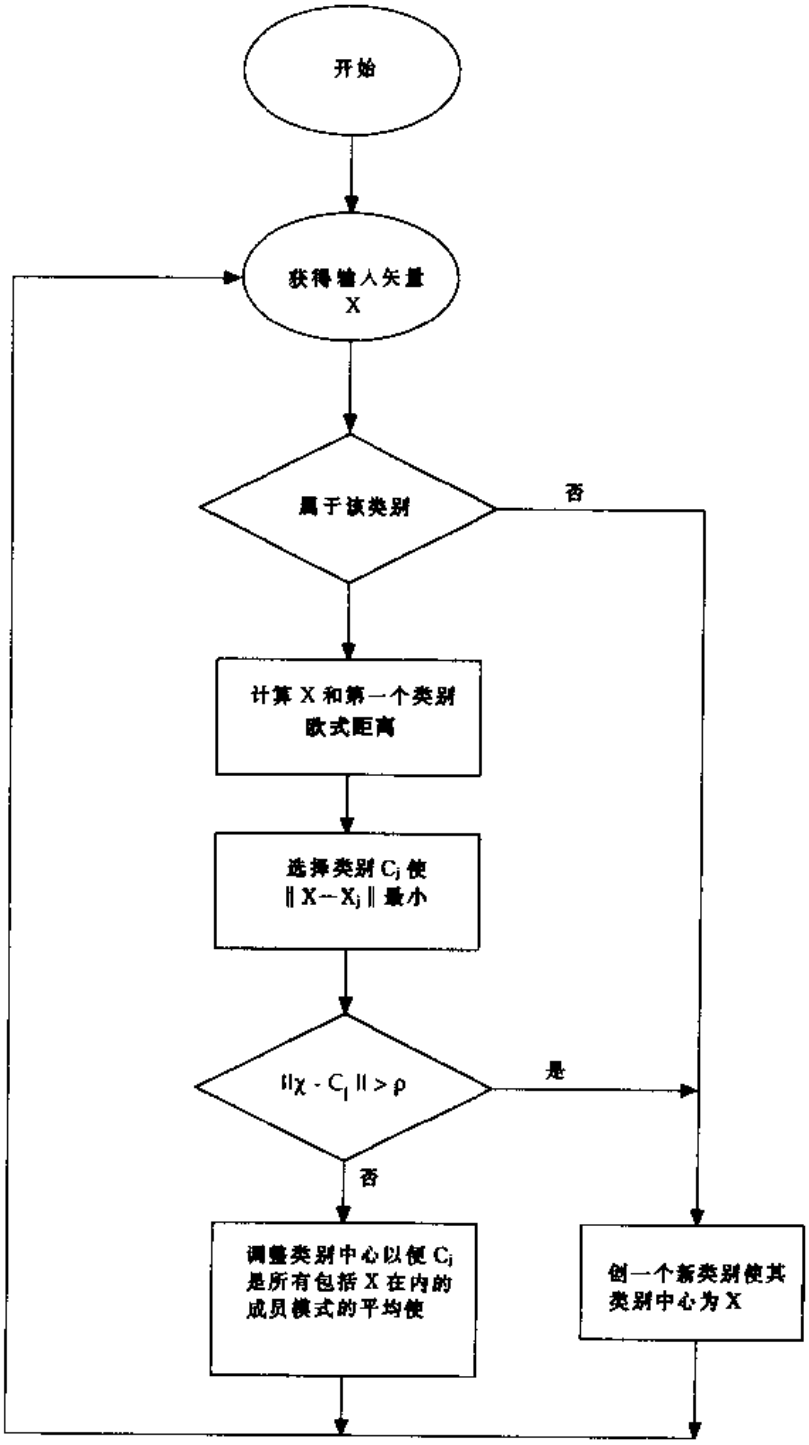


图 10.2 矢量量化过程

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

清单 10.1

```
/*************************************************************************
*                                                                        *
* VECTOR QUANTIZATION                                                    *
*                                                                        *
*************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <math.h>
// FUNCTION PROTOTYPES

// DEFINES
#define     SUCCESS      1
#define     FAILURE      0
#define     TRUE         1
#define     FALSE        0
#define     MAXVECTDIM   20
#define     MAXPATTERN   20
#define     MAXCLUSTER   10

// ***** Defined structures & classes *****
struct aCluster {
  double    Center[MAXVECTDIM];
  int       Member[MAXPATTERN];          //Index of Vectors belonging to this cluster
  int       NumMembers;
};

struct aVector {
  double    Center[MAXVECTDIM];
  int       Size;
};

class VQsyst {
private:
  double    Pattern[MAXPATTERN][MAXVECTDIM+1];
  aCluster  Cluster[MAXCLUSTER];
  int       NumPatterns;                 // Number of patterns
  int       SizeVector;                  // Number of dimensions in vector
  int       NumClusters;                 // Curr number of clusters
  double    Threshold;
  void      Attach(int,int);             // Add spec'd pattern to spec'd
                                         // Clusters membership list.
  int       AllocateCluster();           //
  void      CalcNewClustCenter(int);     // find cluster center for modified
                                         // clusters
  double    EucNorm(int, int);           // Calc Euclidean norm vector
  int       FindClosestCluster(int);     //ret indx of clust closest to pattern
                                         //whose index is arg
public:
  VQsyst();
  int LoadPatterns(char *fname);         // Get pattern data to be clustered
  void RunVQ();                          // Overall control Vector Quant process
  void ShowClusters();                   // Show results on screen
  void SaveClusters(char *fname);        // Save results to file
};

//================IMPLENTATION OF VQ METHODS=============================

//*************************************************************************
// Constructor
//*************************************************************************
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/lupaohe2
Communication & Cooperation: lupaohe@yahoo.com.cn


```
VQsyst::VQsyst(){
  NumClusters=0;
  }
//*************************************************************************
// LoadPatterns                                                           *
//  Loads pattern information from disk.                                  *
//     1) Number of patterns to be processed                              *
//     2) Size of the vectors to be processed                             *
//     3) Max dist permitted between cluster centers                      *
//     2) Pattern definitions                                             *
//*************************************************************************

int VQsyst::LoadPatterns(char *fname){
  FILE *InFilePtr;
  int   i,j;
  double x;
if((InFilePtr = fopen(fname, "r")) == NULL)
   return FAILURE;
fscanf(InFilePtr, "%d", &NumPatterns);          // Read # of patterns
fscanf(InFilePtr, "%d", &SizeVector);           // Read dimension of vector
fscanf(InFilePtr, "%lg", &Threshold);           // Read Euc dist Threshold.
for (i=0; i<NumPatterns; i++) {                 // For each vector
  for (j=0; j<SizeVector; j++) {                // create a pattern
    fscanf(InFilePtr,"%lg",&x);                 // consisting of all elements
    Pattern[i][j]=x;
    printf("Pattern[%d][%d]=%f\n",i,j,Pattern[i][j]);
    } /* endfor */
  } /* endfor */
 printf("\n");
 return SUCCESS;
 }

 //*************************************************************************
 // RunVQ                                                                  *
 //  Provides overall control of VQ algorithm.  K clusters                 *
 //  We choose the first K vectors to do this                              *
 //*************************************************************************

 void VQsyst::RunVQ(){
  int pat,Winner;
  double dist;
 for (pat=0; pat<NumPatterns; pat++) {
   Winner = FindClosestCluster(pat);            // Attach the curr input to closest
                                                // cluster
   if (Winner == -1) {
    Winner=AllocateCluster();
    }
   else {
    dist= EucNorm(pat,Winner);
    dist= sqrt(dist);                           // because eucnorm doesn't do this
    if (dist>Threshold) {
      Winner=AllocateCluster();                 // Above threshold so allocats new
      printf("Creating NEW cluster number:%d\n",Winner);
      } /* endif */                            // cluster
    } /* endif */
   printf("patern %d assignad to cluster %d\n",pat,Winner);
   Attach(Winner,pat);                          // Attach pettern to winner
   CalcNewClustCenter(Winner);                  // Adapt clust center
   } /* endfor */
 }
 //*************************************************************************
 // AllocateCluster                                                        *
 //   Designate the next free cluster as active                            *
 //*************************************************************************
```

More documents and datum download website
LuZhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
int VQsyst::AllocateCluster(){
  int n;
n=NumClusters;
NumClusters++;
return n;
}

//*************************************************************************
// EucNorm                                                                *
//  Returns the Euclidian norm between a pattern, p, and a cluster        *
//  center,c-1 he first K vectors to do this                              *
//*************************************************************************

double VQsyst::EucNorm(int p, int c){          // Calc Euclidean norm of vector difference
double dist;                                   // between pattern vector, p, and cluster
int i;                                         // center, c.
dist=0;
for (i=0; i<SizeVector ;i++){
  dist += (Cluster[c].Center[i]-Pattern[p][i])*(Cluster[c].Center[i]-Pattern[p][i]);
   } /* endfor */
return dist;
}

//*************************************************************************
// Find ClosestCluster                                                    *
//  Returns the index of the cluster to which the pattern, pat, has the   *
//  closest Euclidean distance.                                           *
//*************************************************************************

int VQsyst::FindClosestCluster(int pat){
  int i, ClustID;
  double MinDist, d;
MinDist =9.9e+99;
ClustID=-1;
for (i=0; i<NumClusters; i++) {
  d=EucNorm(pat,i);
  if (d<MinDist) {
    MinDist=d;
    ClustID=i;
    } /* endif */
  } /* endfor */
if (ClustID<0) {
  printf("No Clusters exist\n");
  } /* endif */
return ClustID;
}
//*************************************************************************
// Attach                                                                 *
//  Adds the pattern (whose index is p) to the cluster center whose index *
//  is p.                                                                 *
//*************************************************************************

void VQsyst::Attach(int c,int p){
int MemberIndex;

  MemberIndex=Cluster[c].NumMembers;
  Cluster[c].Member[MemberIndex]=p;
  Cluster[c].NumMembers++;
}

//*************************************************************************
// CalcNewClustCenter                                                     *
//  Calculate a new cluster center for the specified cluster,c            *
//                                                                        *
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
//*************************************************************************

void VQsyst::CalcNewClustCenter(int c){
  int VectID,j,k;
  double tmp[MAXVECTDIM];

  for (j=0; j<SizeVector; j++) {                    // clear workspace
    tmp[j]=0.0;
    } /* endfor */
  for (j=0; j<Cluster[c].NumMembers; j++) {         //traverse member vectors
    VectID=Cluster[c].Member[j];
    for (k=0; k<SizeVector; k++) {                  //traverse elements of vector
      printf("Cluster[%d]  Pattern[%d][%d]=%f,
Member_ID=%d\n",c,VectID,k,Pattern[VectID][k],VectID);
      tmp[k] += Pattern[VectID][k];                 // add (member) pattern element into temp
      } /* endfor */
    } /* endfor */
  for (k=0; k<SizeVector; k++) {                    //traverse elements of vector
    tmp[k]=tmp[k]/Cluster[c].NumMembers;
    Cluster[c].Center[k]=tmp[k];
    } /* endfor */
}
//*************************************************************************
// ShowClusters                                                           *
//  Display the cluster centers for each of the allocated clusters        *
//*************************************************************************

void VQsyst::ShowClusters(){
  int cl;
for (cl=0; cl<NumClusters; cl++) {
  printf("\nCLUSTER %d ==>[%f,%f]\n", cl,Cluster[cl].Center[0],Cluster[cl].Center[1]);
  } /* endfor */
}

//*************************************************************************
// SaveClusters                                                           *
//  Store the cluster centers for each of the allocated clusters          *
//  to the designated file                                                *
//*************************************************************************
void VQsyst::SaveClusters(char *fname){
  FILE *OutFilePtr;
  int cl;
if((OutFilePtr = fopen(fname, "r")) == NULL){
  printf("Unable to open file %s for output",fname);
  }
 else {
  for (cl=0; cl<NumClusters; cl++) {
    fprintf(OutFilePtr,"\nCLUSTER %d ==>[%f,%f]\n",
        cl,Cluster[cl].Center[0],Cluster[cl].Center[1]);
    } /* endfor */
  fclose(OutFilePtr);
  }
}

//*************************************************************************
// Main                                                                   *
//                                                                        *
//                                                                        *
//*************************************************************************
main(int argc, char *argv[]) {
  VQsyst VQ;
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & cooperation: luzhenbo@yahoo.com.cn


```
if (argc<3) {
  printf("USAGE: VQ PATTERN_FILE(input) CLUSTER_FILE(output)\n");
  exit(0);
  }
if (VQ.LoadPatterns(argv[1])==FAILURE){
  printf("UNABLE TO READ PATTERN_FILE:%s\n",argv[1]);
  exit(0);
  }
VQ.RunVQ();
VQ.ShowClusters();
}
```

注意,矢量量化有两个缺点:

1. 对输入矢量输入的顺序很敏感(可以注意到,K-均值算法对矢量输入的顺序不敏感。进一步可观察到,在 K-均值算法中引入距离门限值这种想法根本不难)。

2. 在创建新类别时门限距离必须提前任意选择好。遗憾的是,这个参数选择不利,会产生所不希望的结果。这点将在下面 10.3.1 和 10.3.2 节的例子中说明。

在把注意力转到矢量量化的例子之前,我们将注意到:已经描述过的方法要求必须保留每一个输入模式矢量,以用于计算新的聚类中心。当需要考虑存贮容量或速度的优先问题时,可以采用滑动窗口的方法来计算新的聚类中心。

10.3.1 VQ举例 1

图 10.3 中给出了二维欧式空间中的 12 个点。这个例子中,我们的任务是利用矢量量化算法进行聚类。

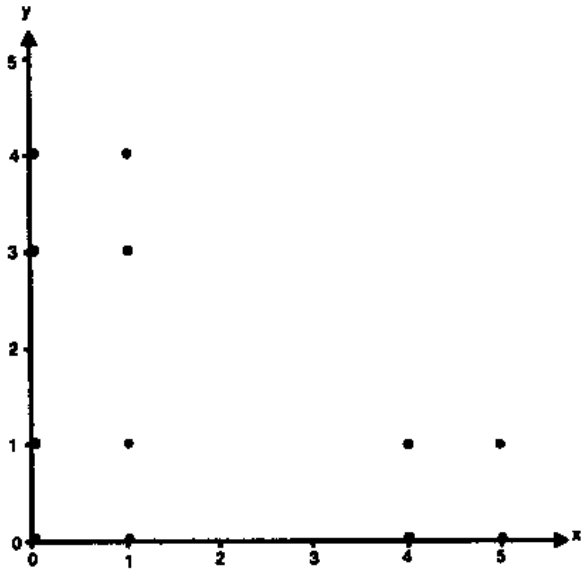


图 10.3 矢量量化举例 1 中的输入模式

在下面的计算中,给定距离门限值 $\rho=2.0$。输入模式为:

模式[0][0]=0.000000
模式[0][1]=0.000000
模式[1][0]=1.000000
模式[1][1]=0.000000

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xidian.edu.cn

模式[2][0]=0.000000
模式[2][1]=1.000000
模式[3][0]=1.000000
模式[3][1]=1.000000
模式[4][0]=0.000000
模式[4][1]=3.000000
模式[5][0]=1.000000
模式[5][1]=3.000000
模式[6][0]=0.000000
模式[6][1]=4.000000
模式[7][0]=1.000000
模式[7][1]=4.000000
模式[8][0]=4.000000
模式[8][1]=0.000000
模式[9][0]=5.000000
模式[9][1]=0.000000
模式[10][0]=4.000000
模式[10][1]=1.000000
模式[11][0]=5.000000
模式[11][1]=1.000000

模式 0:
无类别存在,因此分配一个新的类别 0,
模式 0 分到类别 0 中

新的聚类中心是:
类别 0 ==>[0.000000,0.000000]
类别成员:
  类别 0 ==> {0}

模式 1:
最近的类别是:0
距离 1.000000<2.000000
所以类别 0 通过距离测试
新的聚类中心是:
聚类 0 ==>[0.500000,0.000000]
类别成员:
  类别 0 ==> {0 1}

模式 2:

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

新的聚类中心是:0
距离 1.118034<2.000000
所以类别 0 通过距离测试
模式 2 分到类别 0 中
新的聚类中心是:
类别 0 ==>[0.333333,0.333333]
类别成员:
类别 0 ==> {0 1 2}

模式 3:
新的聚类中心是:0
距离 0.942809<2.000000
所以类别 0 通过距离测试
模式 3 分到类别 0 中

新的聚类中心是:
类别 0 ==>[0.500000,0.500000]
类别成员:
类别 0 ==> {0 1 2 3}

模式 4:
最近的类别是:0
距离 2.549510>2.000000
所以类别 0 未通过距离测试
因此创建新的类别:1
模式 4 分到类别 1 中

新的聚类中心是:
类别 0 ==>[0.500000,0.500000]
类别 1 ==>[0.000000,3.000000]
类别成员:
类别 0 ==> {0 1 2 3}

类别 1 ==> {4}

模式 5:
最近的类别是:1
距离 1.000000<2.000000
所以类别 1 通过距离测试
模式 5 分到类别 1 中

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

新的聚类中心是：
类别 0 ==>[0.500000,0.500000]
类别 1 ==>[0.500000,3.000000]
类别成员：
  类别 0 ==> {0 1 2 3}
  类别 1 ==> {4 5}

模式 6：
最近的类别是:1
距离 1.118034<2.000000
所以类别 1 通过距离测试
模式 6 分到类别 1 中

新的聚类中心是：
类别 0 ==>[0.500000,0.500000]
类别 1 ==>[0.333333,3.333333]
类别成员：
  类别 0 ==> {0 1 2 3}

  类别 1 ==> {4 5 6}

模式 7：
最近的类别是:1
距离 0.942809<2.000000
所以类别 1 通过距离测试
模式 7 分到类别 1 中

新的聚类中心是：
类别 0 ==>[0.500000,0.500000]
类别 1 ==>[0.500000,3.500000]
类别成员：
  类别 0 ==> {0 1 2 3}

  类别 1 ==> {4 5 6 7}

模式 8：
最近的类别是:0
距离 3.535534>2.000000
所以类别 0 未通过距离测试

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

因此创建新的类别:2
模式 8 分到类别 2 中

新的聚类中心是:
类别 0 ==>[0.500000,0.500000]
类别 1 ==>[0.500000,3.500000]
类别 2 ==>[4.000000,0.000000]
类别成员:

类别 0 ==> {0 1 2 3}

类别 1 ==> {4 5 6 7}

类别 2 ==>{8}

模式 9:
最近的类别是:2
距离 1.000000<2.000000
所以类别 2 通过距离测试
模式 9 分到类别 2 中

新的聚类中心是:
类别 0 ==>[0.500000,0.500000]
类别 1 ==>[0.500000,3.500000]
类别 2 ==>[4.500000,0.000000]
类别成员:
类别 0 ==> {0 1 2 3}

类别 1 ==> {4 5 6 7}

类别 2 ==>{8 9}

模式 10:
最近的类别是:2
距离 1.118034<2.000000
所以类别 2 通过距离测试
模式 10 分到类别 2 中

新的聚类中心是:
类别 0 ==>[0.500000,0.500000]

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/lzb1102
Communication & Cooperation: luzhenbo@yahoo.com.cn

类别 1 ==>[0.500000,3.500000]
类别 2 ==>[4.333333,0.333333]

类别成员:
类别 0 ==> {0 1 2 3}

类别 1 ==> {4 5 6 7}

类别 2 ==>{8 9 10}

模式 11:
最近的类别是:2
距离 0.942809<2.000000
所以类别 2 通过距离测试
模式 11 分到类别 2 中

新的聚类中心是:
类别 0 ==>[0.500000,0.500000]
类别 1 ==>[0.500000,3.500000]
类别 2 ==>[4.500000,0.500000]
类别成员:
类别 0 ==> {0 1 2 3}

类别 1 ==> {4 5 6 7}

类别 2 ==>{8 9 10 11}

注意,我们已经得出三个类别,聚类中心分别为:$C_1=(0.5,0.5)$,$C_2=(0.5,3.5)$,$C_3=(4.5,0.5)$。类别组成列表如下:$S(1)=\{0,1,2,3\}$,$S(2)=\{4,5,6,7\}$,$S(3)=\{8,9,10,11\}$。这些结果如图 10.4 所示。还可以注意到,已识别出的类别正是我们所观察到的这些模式应属于的类别。然而,我们现在要使用一个不同的距离门限值来重复这个例子。

10.3.2 VQ 举例 2

现在,我们将研究当距离门限值 $\rho=3.5$ 时,对于例 1 中所用的同样模式矢量量化将产生什么结果。不要奇怪,我们会得到一个不同的结果。计算如下:

输入模式为:
模式[0][0]=0.000000
模式[0][1]=0.000000
模式[1][0]=1.000000
模式[1][1]=0.000000

More documents and datum download website
LuZhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

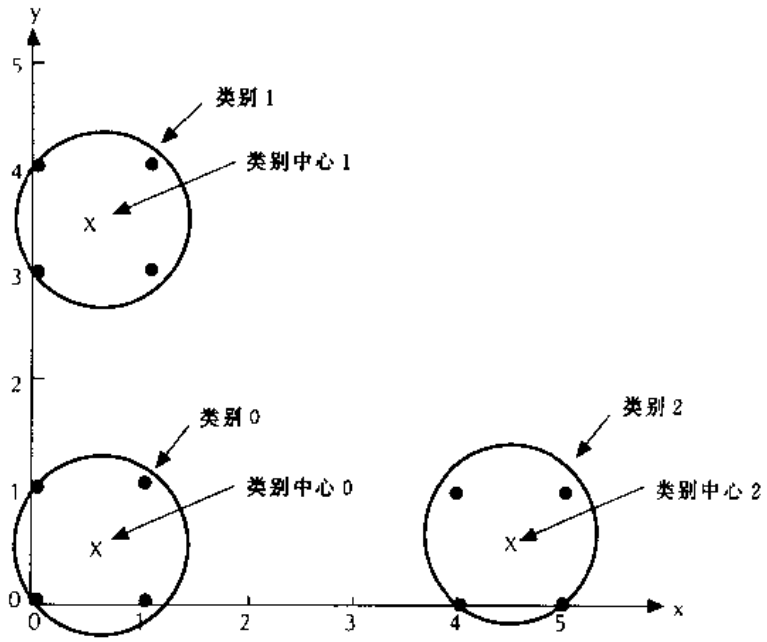


图 10.4 VQ形成的类别(ρ=2.0)

模式[2][0]=0.000000
模式[2][1]=1.000000
模式[3][0]=1.000000
模式[3][1]=1.000000
模式[4][0]=0.000000
模式[4][1]=3.000000
模式[5][0]=1.000000
模式[5][1]=3.000000
模式[6][0]=0.000000
模式[6][1]=4.000000
模式[7][0]=1.000000
模式[7][1]=4.000000
模式[8][0]=4.000000
模式[8][1]=0.000000
模式[9][0]=5.000000
模式[9][1]=0.000000
模式[10][0]=4.000000
模式[10][1]=1.000000
模式[11][0]=5.000000
模式[11][1]=1.000000

模式 0:
无类别存在,因此分配一个新的类别 0,

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@hotmail.com

模式 0 分到类别 0 中

新的聚类中心是:
类别 0 ==>[0.000000,0.000000]
类别成员:
  类别 0 ==> {0}

模式 1:
最近的类别是:0
距离 1.000000<3.500000
所以类别 0 通过距离测试
模式 1 分到类别 0 中

新的聚类中心是:
类别 0 ==>[0.500000,0.000000]
类别成员:
  类别 0 ==> {0 1}

  类别 1 ==> {0 1 2 3}

模式 2:
最近的类别是:0
距离 1.118034<3.500000
所以类别 0 通过距离测试
模式 2 分到类别 0 中

新的聚类中心是:
类别 0 ==>[0.333333,0.333333]
类别成员:
  类别 0 ==> {0 1 2}

模式 3:
最近的类别是:0
距离 0.942809<3.500000
所以类别 0 通过距离测试
模式 3 分到类别 0 中

新的聚类中心是:
类别 0 ==>[0.500000,0.500000]
类别成员

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

新的聚类中心是:
类别 0 ==>[0.500000,2.000000]
类别 1 ==>[4.000000,0.000000]
类别成员:
类别 0 ==>{0 1 2 3 4 5 6 7}

类别 1 ==>{8}

模式 4:
最近的类别是:0
距离 2.549510<3.500000
所以类别 0 通过距离测试
模式 4 分到类别 0 中

新的聚类中心是:
类别 0 ==>[0.400000,1.000000]
类别成员:
类别 0 ==> {0 1 2 3 4}

模式 5:
最近的类别是:0
距离 2.088061<3.500000
所以类别 0 通过距离测试
模式 5 分到类别 0 中

新的聚类中心是:
类别 0 ==>[0.500000,1.333333]
类别成员:
类别 0 ==> {0 1 2 3 4 5}

模式 6:
最近的类别是:0
距离 2.713137<3.500000
所以类别 0 通过距离测试
模式 6 分到类别 0 中

新的聚类中心是:
类别 0 ==>[0.428571,1.714286]
类别成员:

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xidian.edu.cn

类别 0 ==> {0 1 2 3 4 5 6}

模式 7:
最近的类别是:0
距离 2.356060<3.500000
所以类别 0 通过距离测试
模式 7 分到类别 0 中

新的聚类中心是:
类别 0 ==>[0.500000,2.000000]
类别成员:
类别 0 ==> {0 1 2 3 4 5 6 7}

模式 8:
最近的类别是:0
距离 4.031129>3.500000
所以类别 0 未通过距离测试
因此创建新的类别:1
模式 8 分到类别 1 中

模式 9:
最近的类别是:1
距离 1.000000<3.500000
所以类别 1 通过距离测试
模式 9 分到类别 1 中

新的聚类中心是:
类别 0 ==>[0.500000,2.000000]
类别 1 ==>[4.500000,0.000000]
类别成员:
类别 0 ==> {0 1 2 3 4 5 6 7}

类别 1 ==> {8 9}

模式 10:
最近的类别是:1
距离 1.118034<3.500000
所以类别 1 通过距离测试
模式 10 分到类别 1 中

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication: luzhenbo@yahoo.com.cn

新的聚类中心是:
类别 0 ==>[0.500000,2.000000]
类别 1 ==>[4.333333,0.333333]
类别成员:
类别 0 ==> {0 1 2 3 4 5 6 7}

类别 1 ==> {8 9 10}

模式 11:
最近的类别是:1
距离 0.942809<3.500000
所以类别 1 通过距离测试
模式 11 分到类别 1 中

新的聚类中心是:
类别 0 ==>[0.500000,2.000000]
类别 1 ==>[4.500000,0.500000]
类别成员:
类别 0 ==> {0 1 2 3 4 5 6 7}

类别 1 ==> {8 9 10 11}

注意到我们只得出了两个类别,其聚类中心为 $\mathbf{C}_1=(0.5,2.0)$,$\mathbf{C}_2=(0.5,3.5)$,类别组成如下:$S(1)=\{0,1,2,3,4,5,6,7\}$,$S(2)=\{8,9,10,11\}$。这些结果如图 10.5 所示。这时,所识别出的类别不再是我们所认为的类别。我们理智地看到,选择门限距离太大很容易掩盖有意义的类别。相反,选择门限距离太小会增加许多无用的类别。(在极限的情况下,每个输入模式矢量都有自己的类别。)

10.3.3 VQ 举例 3

作为最后一个矢量量化处理的例子,我们要说明聚类结果与矢量出现顺序的关系。对于这个例子,当重排输入数据的顺序时我们仍保持距离门限值 $\rho=3.5$ 不变。注意到虽然产生了同样数目的类别,但类别中成员和聚类中心的位置改变了许多。图 10.6 给出了在这种情况下应用矢量量化算法形成的模式(类别)组。对这个例子的计算如下:

输入模式为:
模式[0][0]=1.000000
模式[0][1]=0.000000
模式[1][0]=4.000000
模式[1][1]=0.000000
模式[2][0]=0.000000
模式[2][1]=1.000000

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

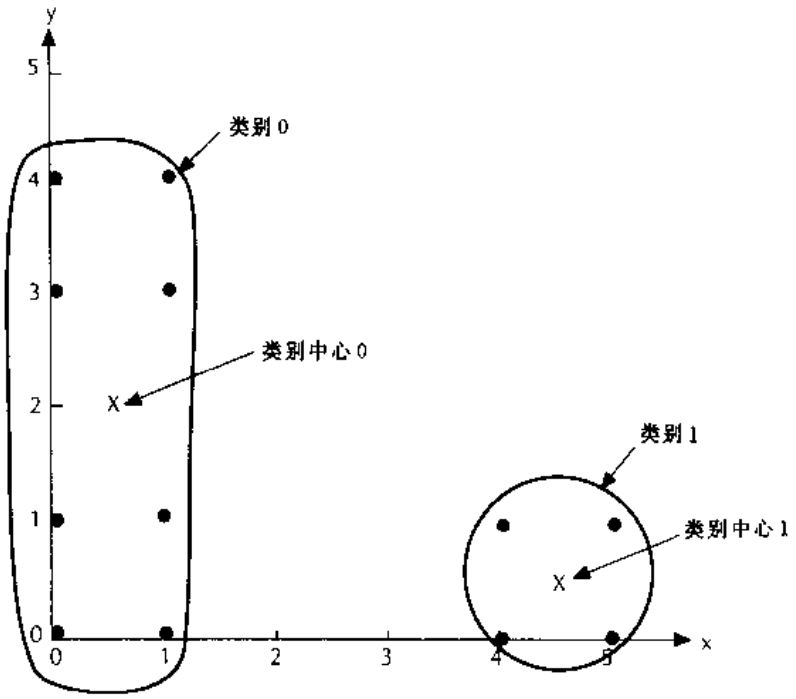


图 10.5 ρ=3 时 VQ 形成的类别

模式[3][0]=0.000000
模式[3][1]=3.000000
模式[4][0]=1.000000
模式[4][1]=1.000000
模式[5][0]=4.000000
模式[5][1]=3.000000
模式[6][0]=5.000000
模式[6][1]=1.000000
模式[7][0]=0.000000
模式[7][1]=0.000000
模式[8][0]=1.000000
模式[8][1]=3.000000
模式[9][0]=0.000000
模式[9][1]=4.000000
模式[10][0]=1.000000
模式[10][1]=4.000000
模式[11][0]=5.000000
模式[11][1]=0.000000

模式 0:
无类别存在,因此分配一个新的类别 0

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

模式 0 分到类别 0 中

新的聚类中心是:
类别 0 ==>[1.000000,0.000000]
类别成员:
类别 0 ==> {0}

模式 1:
最近的类别是:0
距离 3.000000<3.500000
所以类别 0 通过距离测试
模式 1 分到类别 0 中

新的聚类中心是:
类别 0 ==>[2.500000,0.000000]
类别成员:
类别 0 ==> {0 1}

模式 2:
最近的类别是:0
距离 2.692582<3.500000
所以类别 0 通过距离测试
模式 2 分到类别 0 中

新的聚类中心是:
类别 0 ==>[1.666667,0.333333]
类别成员:
类别 0 ==> {0 1 2}

模式 3:
最近的类别是:0
距离 3.144660942809<3.500000
所以类别 0 通过距离测试
模式 3 分到类别 0 中

新的聚类中心是:
类别 0 ==>[1.250000,1.000000]
类别成员:
类别 0 ==> {0 1 2 3}

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

模式 4:
最近的类别是:0
距离 0.2250000<3.500000
所以类别 0 通过距离测试
模式 4 分到类别 0 中

新的聚类中心是:
类别 0 ==>[1.200000,1.000000]
类别成员:
  类别 0 ==> {0 1 2 3 4}

模式 5:
最近的类别是:0
距离 2.800000<3.500000
所以类别 0 通过距离测试
模式 5 分到类别 0 中

新的聚类中心是:
类别 0 ==>[1.666667,1.000000]
类别成员:
  类别 0 ==> {0 1 2 3 4 5}

模式 6:
最近的类别是:0
距离 3.333333<3.500000
所以类别 0 通过距离测试
模式 6 分到类别 0 中

新的聚类中心是:
类别 0 ==>[2.142857,1.000000]
类别成员:
  类别 0 ==> {0 1 2 3 4 5 6}

模式 7:
最近的类别是:0
距离 2.364706<3.500000
所以类别 0 通过距离测试
模式 7 分到类别 0 中

新的聚类中心是:

More documents and datum download website
LuZhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

类别 0 ==>[1.875000,0.875000]
类别成员:
类别 0 ==> {0 1 2 3 4 5 6 7}

模式 8:
最近的类别是:0
距离 2.298097<3.500000
所以类别 0 通过距离测试
模式 8 分到类别 0 中

新的聚类中心是:
类别 0 ==>[1.777778,1.111111]
类别成员:
类别 0 ==> {0 1 2 3 4 5 6 7 8}

模式 9:
最近的类别是:0
距离 3.392075<3.500000
所以类别 0 通过距离测试
模式 9 分到类别 0 中

新的聚类中心是:
类别 0 ==>[1.600000,1.400000]
类别成员:
类别 0 ==> {0 1 2 3 4 5 6 7 8 9}

模式 10:
最近的类别是:0
距离 2.668333<3.500000
所以类别 0 通过距离测试
模式 10 分到类别 0 中

新的聚类中心是:
类别 0 ==>[1.545455,1.636364]
类别成员:
类别 0 ==> {0 1 2 3 4 5 6 7 8 9 10

模式 11:
最近的类别是:0
距离 3.822508>3.500000

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

所以类别 0 未通过距离测试
因此创建新的类别:1
模式 11 分到类别 1 中

新的聚类中心是:
类别 0 ==>[1.545455,1.636364]
类别 1 ==>[5.000000,0.000000]
类别成员:
类别 0 ==> {0 1 2 3 4 5 6 7 8 9 10}

类别 1 ==> {11}

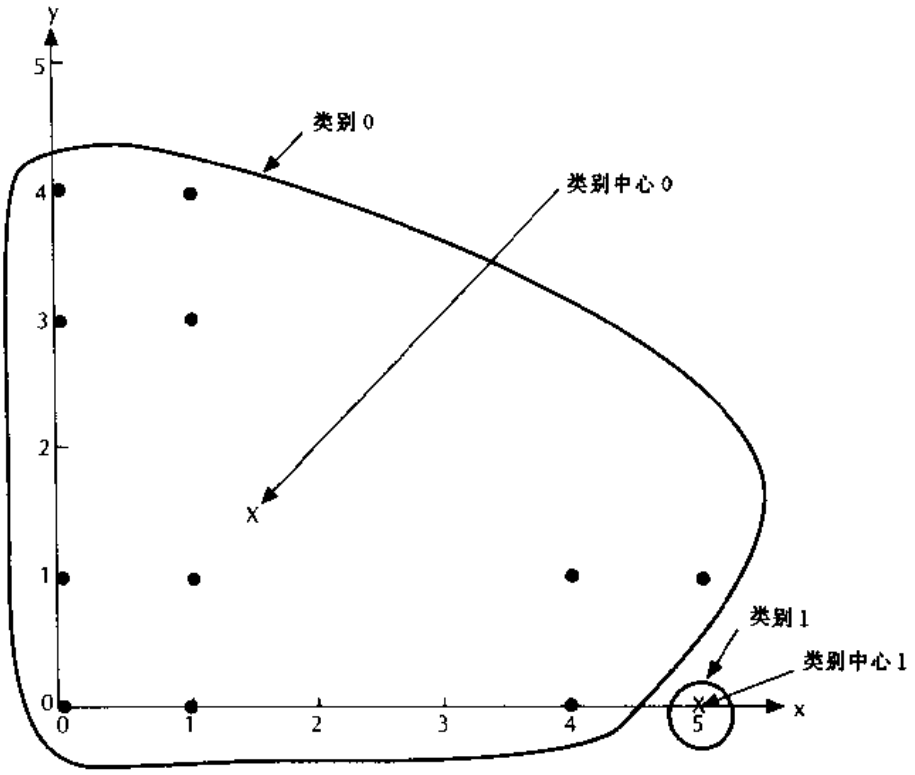


图 10.6 当输入模式出现的顺序改变时,VQ 形成的类别($\rho=3.5$)

10.4 ART 基本原理

自适应共振理论网络是一种无监督的矢量分类器,它能按照已存储的最相似的模式对接收输入矢量进行分类。它也能提供一种机制,允许自适应扩充神经元的输出层直到达到一个足够的大小,这个大小是根据分类的数目自动检查确定的。如果已确定一个输入模式明显地不同于已存在的类,则 ART 网络能创建一个新的相应于这个输入模式的神经元。这种决定称作警戒测试,在自适应反馈网络中使其具体化。从而,ART 结构允许用户控制置于同一类

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

中的模式的相似程度。

ART 的简化结构如图 10.7 所示。

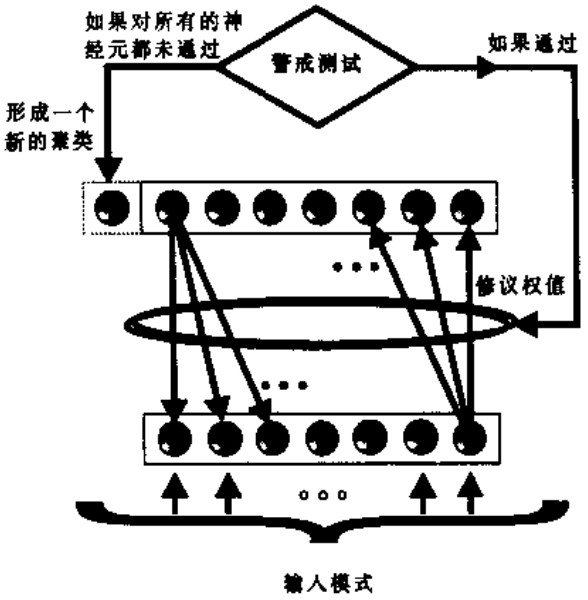


图 10.7 简化的 ART 结构

10.5 稳定性和可塑性两难问题

人脑具有以一种不必使已存信息忘记的形式学习和记忆新事物的能力。为了设计一个和大脑一样的真正智能化模式识别机，将这种能力引入我们的模型是非常必要的。

在前面的章节中讨论过的大多数模式识别范例中，如果我们试图以一种增长形式存贮新的模式，那么它们会忘记旧的信息。在这些范例中，首先需要一个样本集，然后用它去训练系统。在训练过程中，调整系统中的权值来对模式进行编码。通常需重复性地应用样本，直到网络完成对样本集的学习为止。一旦训练足够多，系统即能在可使用的方式下运行[参见图 4.4(a)和 4.4(b)]，并且不允许再修改附加参数(或权值)。

不过，在实际应用中，网络处于不断变化的环境中。如果不继续训练，系统最终仍会变得不精确。在网络运行阶段也会遇到新的样本。考虑一个简单的例子。假设给我们安排了一项训练识别各种人脸的模式分类器的任务，并且需要根据各种人脸在组织中不同的密级把它们放到各种类别中。那么，就应收集合适的图像用于训练网络。

对于前面章节中讨论过的大多数范例，在模型成功地学习了识别某个组织中每位雇员的面孔之后，训练就结束了，并且不允许进一步修改参数(或权值)。如果在未来的某一时间某一雇员离开了这个组织，或又有新雇员加入这个组织，那么就必须相应地在模型的知识库中删除或增加他们的面孔。然而，对于这些范例，如果一个已经完全训练好了的模型必须学习一个新的模式(在这种情况这个新模式是人的面孔)，那么它会破坏原模型的参数，甚至需要完全重新训练该网络。如果仅仅用新面孔来训练，网络将会对这些面孔很好地加以学习，但却会忘记原先学习过的面孔。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xaut.edu.cn

网络的自适应以及在运行的任何阶段能够很好地学习新模式的能力，称为可塑性。

Grossberg[1987]将稳定性－可塑性的两难问题阐述如下：

怎样设计一个学习系统，使其对有效的事件作出响应时保持可塑性，也即自适应性，同时对不相关事件作出响应时保持稳定性？系统如何去了解怎样在稳定方式和可塑方式之间切换，以达到无混沌的稳定性？特别地，它怎样能在继续学习新事情时仍保留它原来已学过的知识？采取什么措施防止新学习冲走对原先学习的记忆？

ART 网络试图解决稳定性－可塑性的两难问题。照此，ART 提供了一种网络能在不忘记(或降低)旧知识的情况下学习新模式的机制。例如，在字符识别问题中，这种方法对诸如在一个联机系统中训练作者特定的笔迹，或在一个已有的脱机系统中增加新的字体，而不必从零开始重新训练网络是非常有用的。

容限措施的采纳(警戒测试)，允许 ART 结构能解决稳定性－可塑性两难问题。来自环境中的新模式能创建附加分类的类别，但它们不能使一个已存在的记忆改变，除非二者特别相近。

在一个物理系统中，当固有频率发生微小振动时会产生大幅度振动，这种现象称为共振。ART 结构就通过这样一个事实而得名的。即一个处理单元输出的信息在各层之间来回反射。当一个固有模式产生了并且接着发生了稳定的振荡时，神经网络的共振平衡就会发生。在共振的状态中产生的活动模式被称为短期记忆(STM)。短期记忆痕迹的存在仅与单个输入矢量的激励相关。

在自适应共振理论中的学习(即修改权值)只有在共振期间才能发生。用于更新处理单元之间权值的时间，要比达到共振的时间长得多。这些在不同层中和处理单元相关的权值，被称为长期记忆(LTM)痕迹。长期记忆痕迹对在网络的一部分长期保持的信息进行编码。

10.6 ART1：基本工作方式

本节讨论 ART1 的网络结构和工作方式。ART1 的输入是二进制值，如一个字符的原始比特图输入到分类器就是这种情况。ART2 扩展了 ART1 的网络结构，允许实数值输入，如表示一幅图像每个象素的灰度值 。

图 10.8 所示为 ART 网络的总体结构图。ART1 的结构由两层神经元组成，这两层神经元分别称为比较层和识别层。类别判决是由在识别层中的一个单一神经元来作出的。类似于大脑皮层感受区中的细胞组，比较层中的神经元对模式的输入特性作出响应。在这两层之间的突触连接(权值)，可以根据两种不同的学习规则进行双向修改。识别层的神经元具有允许竞争的抑制连接。受生物系统的视觉神经生理学影响，这种机制在人工神经网络的网结构中是普遍的。该网络结构还包括三个附加的模块，即：增益 1，增益 2 和复位模块(参见图 10.8)。

注意，子系统包括两层具有前馈和后馈特性的神经元(比较层和识别层)。该系统决定输入模式是否与已存贮的一个原型相匹配。如果匹配，就会产生共振。定位子系统负责检测在识别层自下而上和自上而下模式之间的失配情况。

识别层对输入矢量所作出的反应，可以比作是通过警戒机制的原始输入矢量。警戒提供了一种输入矢量与激励识别层神经元相应的聚类中心之间的距离测度。当警戒低于预先设置的门限值时，必须创建一个新的类别并且将输入矢量存于该类别中。就是说，在识别层中，将先前未分配的神经元分配到一个与新的输入模式相联系的新类别中。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

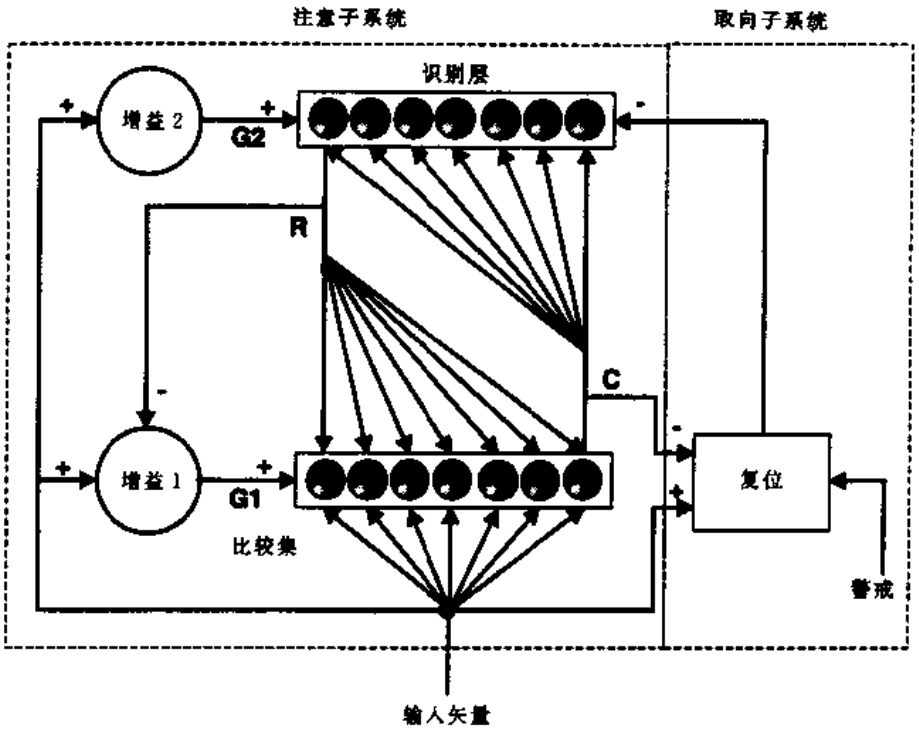


图 10.8 ART 总体结构

识别层遵循胜者取全部的原理(这个过程有时可称为 MAXNET[Kung,1993])。如果输入矢量通过了警戒,获胜的神经元(与输入矢量最像的一个神经元)就会被训练,以使其在特性空间中相应的聚类中心移向输入矢量。比较层也被称为 F1,是一个自下而上层。识别层也被称为 F2,是一个自上而下层。

关于识别层和比较层更详细地描述,请分别参见图 10.9 和图 10.10。

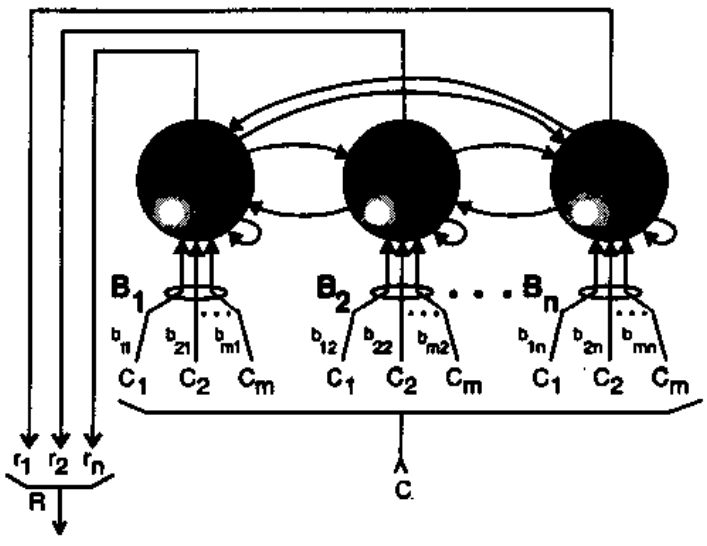


图 10.9 ART 识别层

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@live.com.cn

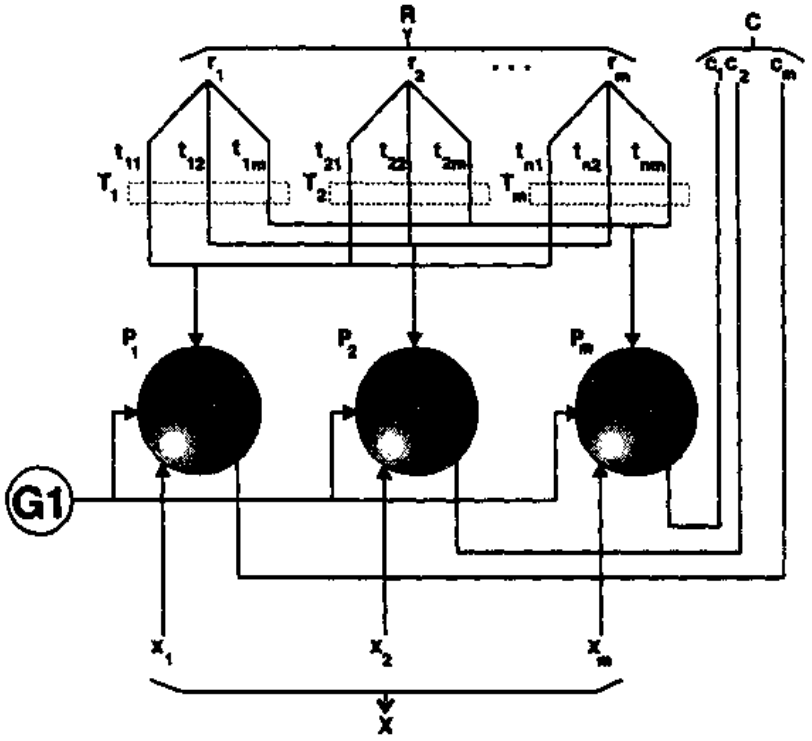


图 10.10 ART 比较层

每个识别层神经元 j 都有一个实权值矢量 $\mathbf{B}_j$ 与其相对应(参见图 10.9)。该矢量对输入模式的一个类别来说,相当于一个已存贮的样本模式。通过它的权值矢量 $\mathbf{B}_j$ 的比较层输出(矢量 C),把每个神经元接收作为输入。

识别层神经元 j 的输出,可由下式表示:

$$\begin{aligned} \text{net}_j &= \sum_{i=1}^{M} b_{ij} c_i \\ r_j &= f(\text{net}_j) = \begin{cases} 1, \text{当 net}_j > \text{net}_i, \text{对于所有的 } i \neq j \\ 0, \text{其它} \end{cases} \end{aligned} \tag{10-4}$$

其中,c_i 是第 i 个比较层神经元的输出;f 是一个阶梯函数,从而 r_j 结果为一个二进制值;M 是比较层中神经元的数目。

如图 10.10 所示,比较层中的每个神经元 i 接收下列三个输入:

1. 输入模式 X 的一个分量,即 x_i。
2. 增益信号 G1 是一个标量(二进制值);所以,相同的值输入到每个神经元。
3. 来自识别层的一个反馈信号,它是识别层输出的加权和。

通过二进制权值 t_{ji} 的反馈 P_i,可由下式得出:

$$P_i = \sum_{j=1}^{N} t_{ji} r_j \qquad i = 1, \cdots, M \tag{10-5}$$

其中,r_j 是第 j 个识别层神经元的输出,N 是识别层神经元的数目。在图 10.10 中,$\mathbf{T}_j$ 是相应于识别层神经元 j 的权值矢量。矢量 C 为比较层的输出,c_i 表示其第 i 个神经元的输出。

当矢量 R 是零并且输入矢量 X 的分量的逻辑"或"结果为 1 时,增益 1 为 1,如等式(10-6)所示。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

$$G_1=(\overline{r_1|r_2|\cdots|r_N})\cdot(x_1|x_2|\cdots|x_M) \tag{10-6}$$

下列 C++ 程序给出了增益 1 的计算：

```
int ARTNET::Gain1(){
int i,G;
G=Gain2();
for (1=0; i<M; i++) {
  if (RVect[i]==1)
   return 0;
  }/* endfor */
return G;
}
```

当输入矢量 X 的分量的逻辑“或”为 1 时，增益 2 为 1，如等式(10-7)所示：

$$G_2=(x_1|x_2|\cdots|x_M) \tag{10-7}$$

下列 C++ 程序，给出了对增益 2 的计算：

```
int ARTNET::Gain2(){
  int i;
for (1=0; i<M; i++) {
  if (XVect[i]==1)
   return 1;
  }/* endfor */
}
```

比较层利用了三分之二准则，即如果 3 个输入中有 2 个为 1，则输出为 1，否则输出为 0。三分之二准则，可用等式(10-8)表示：

$$c_j=\begin{cases}0, & \text{当 } G_1+x_j+P_j<2\\ 1, & \text{当 } G_1+x_j+P_j\geqslant 2\end{cases} \tag{10-8}$$

如下的 C++ 程序，给出了利用三分之二准则的神经元输出：

```
void ARTNET::RunCompLayer(){
  int i,x;
for (i=0; i<M; i++) {
  x=XVect[i]+Gain1()+PVect[i];
  if (x>=2) {
   Cvect[i]=1;
   }
  else {
  CVect[i]=0;
  }/* endif */
  }/* endfor */
}
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn
Communication & Cooperation: luzhenbo@yahoo.com.cn

ART 过程是分阶段发生的。最初没有输入，那么从式(10-7)中可以看出 G2 为 0。如图 10.10 和 10.11 所示，当输入矢量 **X** 首先提供给网络时，网络就进入识别阶段。在识别的开始阶段，识别层的反馈矢量 **R** 总是被设置为 0。根据式(10-6)和(10-7)可以看出，在这期间，**X** 的出现使 G1 和 G2 均为 1。

根据在识别阶段中的初始条件可以看出，比较层的输出 C 将是未修改的输入矢量 **X**。从而比较层把 **X** 传导到识别层，如图 10.11 所示。

然后，识别层中的每一个神经元计算它的权值矢量 $\mathbf{B}_j$(实数值)和矢量 **C**(比较层的输出)之间的点积。获胜矢量激活，并抑制所有识别层中其它的神经元(注意图 10.9 中的后抑制)。从而矢量 **R** 的一个分量 r_j 为 1 而矢量 **R** 中的其它分量均为 0。由此比较阶段开始。

换句话说，识别阶段使得每个识别层神经元把它的原型(存贮在自下而上的权值)和输入模式($\mathbf{B}_j$ 和 **C** 的点积)进行比较。相互抑制机制使最匹配的一个激活，见图 10.12 所示。

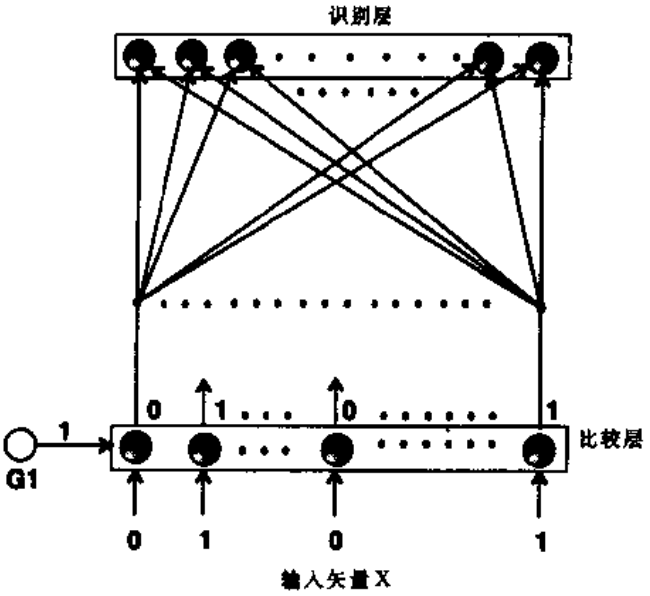


图 10.11 ART 工作方式

步骤 1：G1 = 1，输入矢量通过比较层传到识别层

在比较阶段期间，必须决定一个输入模式是否非常相似于将被吸收掉的已存贮的获胜原型。在这个阶段要执行警戒测试。

在比较阶段中，矢量 **R** 不再是 0，所以增益 1 将是 0。通过三分之二准则，只有在 **X** 和 **P** 矢量中所有分量同时为 1 的神经元才被激活。注意到权值 t_{ji} 是二进制值，然后这个自上而下的反馈通路强迫 **C** 的分量为 0，从而不论什么时候输入矢量 **X** 都不能与已存贮的模式匹配。

令 D 为矢量 x 中 1 的数目，K 为矢量 C 中 1 的数目。那么相似度 **S** 很容易得到，即：$\mathbf{S}=K/D$。

所以，相似矢量 **S** 是比较原型和输入模式之间相似程度的尺度。现在我们必须建立一个标准，通过它并根据相似度来决定是接受还是拒绝类别。警戒测试可以表示如下：

$$\mathbf{S}>\rho \rightarrow \text{警戒测试通过}$$

$$\mathbf{S}\leqslant\rho \rightarrow \text{警戒测试失败}$$

如果警戒性测试通过，那么在输入矢量和获胜原型之间就不存在实质区别。从而所要求

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

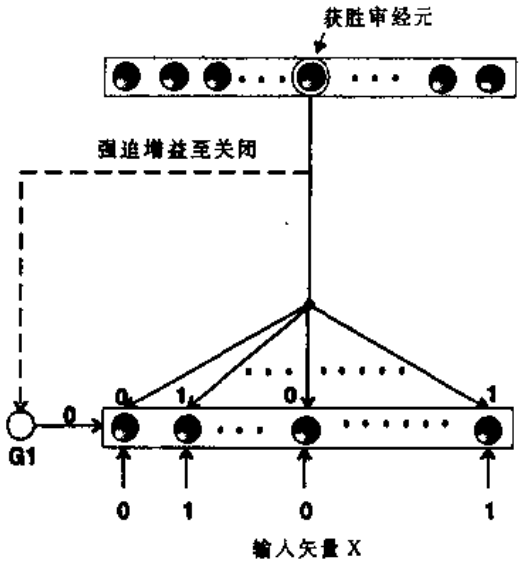


图 10.12 ART 工作方式

步骤 2:选出识别层的获胜神经元

获胜者把它的信号通过它自上而下的权值送回比较层

的动作只是简单地把输入矢量存入获胜神经元的聚类中心。在这种情况下,没有复位信号。因此,当进入搜索阶段时,应调整输入矢量的权值。到此为止,网络的工作就完成了。

如果 S 低于预置的门限值(警戒值),那么模式 p 与获胜神经元聚类中心不十分相似,而且应该抑制激活的神经元。抑制是由复位块执行的,它在整个当前分类的过程中复位目前激活的神经元(参见图 10.13),其中也包括比较阶段。

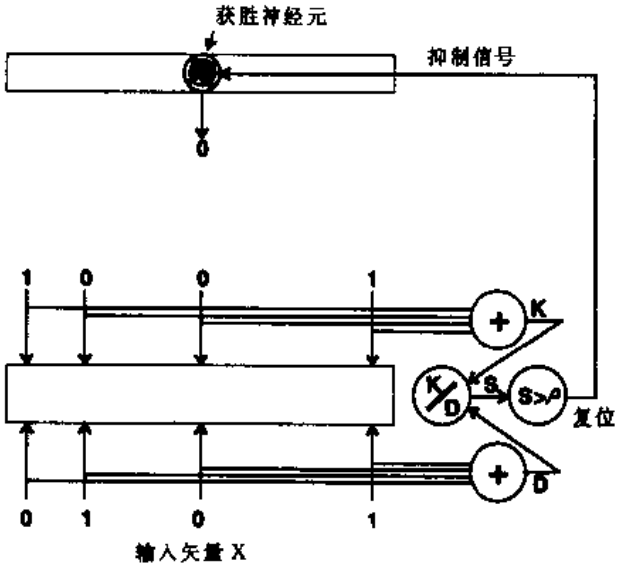


图 10.13 ART 工作方式

步骤 3:输入矢量 x 和来自识别层的矢量 P 进行比较

警戒测试失败。从而复位机制抑制了获胜的神经元

然后,进入搜索阶段。如果没有产生复位信号,那么就认为已经匹配并且分类就完成了。否则,随着激活的识别层神经元被停用,矢量 R 再一次置 0。结果,增益 1(G1)变为 1,以使 X

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

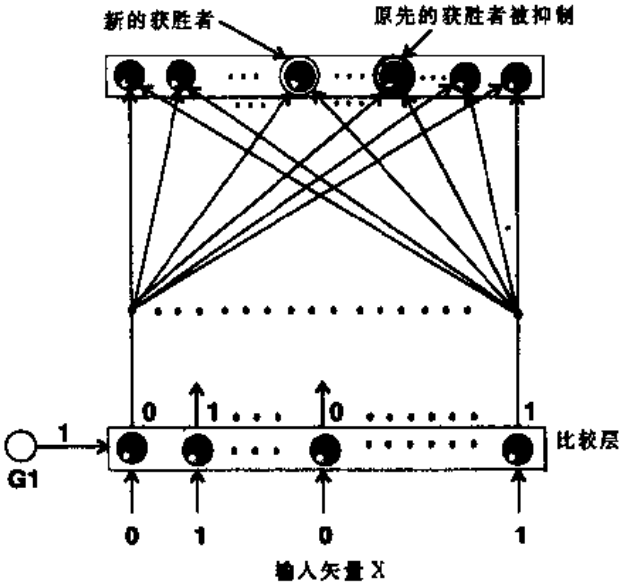


图 10.14 ART 工作方式

步骤 4:原获胜神经元保持禁用状态,输入矢量再次提供给识别层(G1=1),并且选择一个新的获胜者

再一次出现在比较层,并且在识别层中的另一个不同的神经元获胜(见图 10.14)。新的获胜神经元再一次作警戒测试,同前面一样,这一过程不断重复,直到出现下列情况之一:

1. 找到一个在警戒层上($S>\rho$)与 **X** 相似的神经元。激活神经元的权值矢量 $\mathbf{T}_j$ 和 $\mathbf{B}_j$ 被调整;

2. 所有已存入模式均试过。一个原先未分配的神经元与该模式相关,那么令 $\mathbf{T}_j$ 和 $\mathbf{B}_j$ 与该模式匹配。

10.7 ART1:算法

首先,给 b_{ij} 赋相同的小初始权值,并使

$$b_{ij}<\frac{L}{(L-1+m)} \tag{10-9}$$

式中,m 是输入矢量中分量的数目;L 是一个常数,其典型值为 $L=2$。

ART1 结构的算法如下;

1. 将输入模式 **X** 提供给网络,识别层选择网络输出的最大者作为获胜者

$$net_j=\sum_{i=1}^{N} b_{ij}c_i \tag{10-10}$$

式中,N 是比较层中神经元的数目。

2. 执行警戒测试。当且仅当

$$\frac{net_j}{\sum_{i=1}^{N} x_i}>\rho \tag{10-11}$$

则神经元 j 通过警戒测试。式中 ρ 为警戒阈值。

More documents and datum download website
LuZhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

2a. 如果胜者未通过测试,则屏蔽现在的获胜者,并且回到步骤 1 去选择另一个获胜者。

2b. 重复这个循环(步骤 2a 到步骤 1),直到选择一个通过警戒测试的获胜者,然后转到步骤 4。

3. 如果没有神经元通过警戒测试,则产生一个新的神经元来接纳新的模式。

4. 调整获胜神经元的前馈权值。更新从获胜神经元到它的输入的反馈权值。

控制自下而上和自上而下权值的训练公式为:

$$b_{ij} = \frac{Lc_i}{(L-1+\sum c_k)} \tag{10-12}$$

式中,c_i 是比较层矢量的第 i 个分量,j 是获胜识别层神经元的编号。

10.8 增益控制机制

现在,让我们更详细地考察一下增益控制机制和 2/3 规则。

生物系统所表现的强有力的能力之一,就是它们经常能预料到下面该发生的事件。如果在一系列事件中有一件存在,那么它们有能力根据这一事件来估计出事件发生的趋势。例如,在一次谈话中,我们通常能预料到另一个人想说的下一句话。

生物神经系统通常具有层次结构。Grossberg 的大多数工作,集中于研究大脑中实际处理的模型。从而,Grossberg 的 ART 网络(参见图 10.8)只是一个更大系统中网络的层次之一。当识别层被层次结构中的更高层激发时,在 ART 模型中就会产生预期的结果。这会产生一个自上面下的信号,该信号能在一个输入信号从下面到来之前到达比较层。

如果比较层在缺乏任何感知输入模式的情况下产生输出,那么来自于识别层早期激励信号的出现会导致产生错觉。在系统中加入增益控制和 2/3 规则,可使系统避免这种错觉的产生。

如图 10.9 所示,任一来自识别层的信号均会抑制增益。现在回想一下 2/3 规则,只有感知输入模式与来自上面的预知激励一起出现时,来自 G 的抑制才意味着将得出一个来自比较层的输出。如果比较层神经元确实收到了来自这两个源的输入,那么它们会给识别层一个输出使匹配循环开始。

清单 10.2

```
/*************************************************************************
*
* ADAPTIVE RESONANCE THEORY (ART)  NETWORK
*
*************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <math.h>


// DEFINES
#define MAXCNEURONS 75              // MAX COMPARISON LAYER NEURONS
#define MAXRNEURONS 30              // MAX RECOGNITION LAYER NEURONS
#define MAXPATTERNS 30    // MAX NUMBER OF PATTERNS IN A TRAINING SET
#define VERBOSE   1
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xaut.edu.cn


```
class ARTNET
{
private:
 double Wb[MAXCNEURONS][MAXRNEURONS];  // Bottom up weight matrix
 int   Wt[MAXRNEURONS][MAXCNEURONS];    // Top down  weight matrix
 int   InData[MAXPATTERNS][MAXCNEURONS]; // Array of input vectors to be
                                        // presented to the network
 int   NumPatterns;                     // Number of input patterns
 double VigilThresh;                    // Vigilence threshold value
 double L;                              // ART training const (see text)
 int   M;                               // # of neurons in C-layer
 int   N;                               // # of neurons in R-layer
 int   XVect[MAXCNEURONS];              // Current in vect at C-layer.
 int   CVect[MAXCNEURONS];              // Output vector from C-layer
 int   BestNeuron;                      // Current best  R-layer Neuron
 int   Reset;                           // Active when vigilence has
                                        //    disabled someone
 int   RVect[MAXCNEURONS];              // Output vector from R-layer
 int   PVect[MAXCNEURONS];              // WeightedOutput vector from R-layer
 int   Disabled[MAXRNEURONS];           // Resets way of disqualifying neurons
 int   Trained[MAXRNEURONS];            // To identify allocated R-Neurons
 void  ClearPvect();
 void  ClearDisabled();
 void  RecoPhase();                     // Recognition phase
 void  CompPhase();                     // Comparison phase
 void  SearchPhase();                   // Search Phase
 void  RunCompLayer();                  // Calc comparison layer by 2/3 rule
 void  RunRecoLayer();                  // Calc recognition layers R-vect
 void  Rvect2Pvect(int);                // Distribute winners result
 int   Gain1();                         // Comp layer gain
 int   Gain2();                         // Reco layer gain
 double Vigilence();                    // Calc vigilence metric
 void  InitWeights();                   // Initialize weights
 void  Train();                         // Weight adjustment is done here
public:
  ARTNET(void);                         // Constructor/initializations
  int   LoadInVects(char *Fname);       // load all data vectors
  void  Run(int i);                     // Run net w/ ith pattern
  void  ShowWeights();                  // display top down and
                                        //  bottom up weights
  void  ShowInVect();                   // Display current input pattern
  void  ShowOutVect();                  // P-vector from Reco layer(see text)
};

//-------------------------------------------------------------------------


// METHOD DEFINITIONS

ARTNET::ARTNET(){
int i;
L=2.0;
N=MAXRNEURONS;
for (i=0; i<N; i++) {                   //Set all neurons to untrained and enabled
  Trained[i]=0;
  Disabled[i]=0;
  } /* endfor */
}
 int ARTNET::LoadInVects(char *Fname){
 FILE *PFILE;
 int  i,j,k;

 PFILE = fopen(Fname,"r");   // batch
 if (PFILE==NULL){
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
  printf("\nUnable to open file %s\n",Fname);
  exit(0);
  }
fscanf(PFILE,"%d",&NumPatterns);        //How many patterns
fscanf(PFILE,"%d",&M);                  //get width of input vector
fscanf(PFILE,"%lf",&VigilThresh);
for (i=0; i<NumPatterns; i++) {
  for (j=0; j<M; j++) {
    fscanf(PFILE,"%d",&k);              //Read all the pattern data and...

    InData[i][j]=k;                     //  ...save it for later.
    } /* endfor */
  } /* endfor */
InitWeights();
return NumPatterns;
}

int ARTNET::Gain2(){
  int i;
for (i=0; i<M; i++) {
  if (XVect[i]==1)
    return 1;
  } /* endfor */
}


void ARTNET::Rvect2Pvect(int best){
  int i;
for (i=0; i<M; i++) {
  //PVect[i]= Wt[best][i]*RVect[i];
  PVect[i]= Wt[best][i];
  } /* endfor */
}

int ARTNET::Gain1(){
  int i,G;
G=Gain2();
for (i=0; i<M; i++) {
  if (RVect[i]==1)
    return 0;
  } /* endfor */
return G;
}

void ARTNET::RunCompLayer(){
  int i,x;
for (i=0; i<M; i++) {
  x=XVect[i]+Gain1()+PVect[i];
  if (x>=2) {
    CVect[i]=1;
    }
    else {
      CVect[i]=0;
      } /* endif */
    } /* endfor */
  }

  double ARTNET::Vigilence(){
    int i;
    double S,K,D;
  // count # of 1's in p-vect & x-vect
  K=0.0;
  D=0.0;
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: 1021616@qq.com


```
 for (i=0; i<M; i++) {
   K+=CVect[i];
   D+=XVect[i];
   } /* endfor */
 S=K/D;
 return S;
 }

 void ARTNET::RunRecoLayer(){
   int i,j,k;
   double Net[MAXRNEURONS];
   int BestNeruon=-1;
   double NetMax=-1;
 for (i=0; i<N; i++) {                    //Traverse all R-layer Neurons
   Net[i]=0;
   for (j=0; j<M; j++) {                  // Do the product
     Net[i] +=Wb[i][j]*CVect[j];
     } /* endfor */
  if ((Net[i]>NetMax) && (Disabled[i]==0)) { //disabled neurons cant win!
     BestNeuron=i;
     NetMax=Net[i];
     }
   } /* endfor */
 for (k=0; k<N; k++) {
   if (k==BestNeuron)
     RVect[k]=1;
    else
     RVect[k]=0;
   } /* endfor */
 }

 void  ARTNET::RecoPhase(){
   int i;
 //First force all R-layer outputs to zero
 for (i=0; i<N; i++) {
   RVect[i]=0;                            // Winner gets 1
   } /* endfor */
 for (i=0; i<M; i++) {
   PVect[i]=0;                            // lateral inhibition kills the rest
   } /* endfor */
 //Now Calculate C-layer outputs
 RunCompLayer();                          //C-vector now has the result
 RunRecoLayer();                          //Calc dot prod w/ bot up weight & C
 Rvect2Pvect(BestNeuron);
 }
 void  ARTNET::CompPhase(){
   double S;
 RunCompLayer();                          //Cvector<-dif between  x & p
 S=Vigilence();
 if (S<VigilThresh){
   Reset=1;
   RVect[BestNeuron]=0;
   Disabled[BestNeuron]=1;
   }
  else
   Reset=0;
 }

 void  ARTNET::SearchPhase(){
   double S;
 while (Reset) {
 //  Rvect2Pvect(0);                      //Rvect 0 turns on gain 1
   ClearPvect();
   RunCompLayer();                        //Xvect -> Cvect
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
  RunRecoLayer();                    //Find a new winner with prev winners disabled
  Rvect2Pvect(BestNeuron);           //new pvect based on new winner
  S=Vigilence();                     //calc vigilence for the new guy
if (S<VigilThresh){                  //check if he did ok
  Reset=1;                           //   if not disable him too
  RVect[BestNeuron]=0;
  Disabled[BestNeuron]=1;
  }
 else
  Reset=0;                           //Current Best neuron is a good
                                     //  winner...Train him
  } /* endwhile */
if (BestNeuron!=-1) {
  Train();
  }
 else {
  //Failed to allocate a neuron for current pattern.
  printf("Out of neurons in F2\n");
  } /* endif */
ClearDisabled();
}

void ARTNET::ClearDisabled() {
  int i;
for (i=0; i<M; i++) {
  Disabled[i]=0;
  } /* endfor */
}

void ARTNET::ClearPvect() {
  int i;
for (i=0; i<M; i++) {
  PVect[i]=0;
  } /* endfor */
}
 void ARTNET::Train(){
   int i,z=0;
 for (i=0; i<M; i++) {
   z+=CVect[i];
   } /* endfor */
 for (i=0; i<M; i++) {
  Wb[BestNeuron][i]=L*CVect[i]/(L-1+z);
  Wt[BestNeuron][i]=CVect[i];
  } /* endfor */
 Trained[BestNeuron]=1;
 }


 void ARTNET::Run(int tp){
   int i,j;

 ClearPvect();
 for (i=0; i<M; i++) {
  XVect[i]=InData[tp][i];
  } /* endfor */

 RecoPhase();
 CompPhase();
 SearchPhase();
 }

 void  ARTNET::InitWeights(){        // Initialize weights
  int i,j;
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: [illegible]


```
  double b;
for (i=0; i<N; i++) {                    // from R-neuron i
  for (j=0; j<M; j++) {                  // to C-neuron j
    Wt[i][j]= 1;                         // All Init'd to 1
    } /* endfor */
  } /* endfor */
b=L/(L-1+M);
for (i=0; i<N; i++) {                    // from C-neuron i
  for (j=0; j<M; j++) {                  // to R-neuron j
    Wb[i][j]= b;                         //All init'd to 1
    } /* endfor */
  } /* endfor */

}
void ARTNET::ShowWeights(){
  int i,j;
printf("\nTop Down weights:\n");
for (i=0; i<N; i++) {
  if(Trained[i]==1){
    for (j=0; j<M; j++) {
      printf("%d ",Wt[i][j]);
      } /* endfor */
    printf("\n");
    } /* endif */
  } /* endfor */
  printf("\nBottom up weights:\n");
  for (i=0; i<N; i++) {
    if(Trained[i]==1){
      for (j=0; j<M; j++) {
        printf("%f ",Wb[i][j]);
        } /* endfor */
      printf("\n");
      } /* endif */
    } /* endfor */
  }

  void ARTNET::ShowInVect(){
    int i;
  printf("BEST NEURON:%d\nIN: ",BestNeuron);
  for (i=0; i<M; i++) {
    printf("%d ",XVect[i]);
    } /* endfor */
  printf("\n");
  }

  void ARTNET::ShowOutVect(){
    int i;
  printf("OUT: ");
  for (i=0; i<M; i++) {
    printf("%d ",CVect[i]);
    } /* endfor */
  printf("\n");
  }

  //-------------------------------------------------------------------
  ARTNET  ART;

  /*************************************************************************
  * MAIN                                                                   *
  *************************************************************************/


  int main(int argc, char *argv[])
  {
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn


```
  int TstSetSize;
  int i;
if (argc>1) {
  TstSetSize=ART.LoadInVects(argv[1]);
  for (i=0; i<TstSetSize; i++) {
    ART.Run(i);                          //Evaluate ith test pattern
    printf("\n");
    ART.ShowInVect();
    ART.ShowOutVect();
    if (VERBOSE==1) ART.ShowWeights();
  } /* endfor */
  }
 else {
  printf("USAGE: ART PATTERN_FILE_NAME\n");
  exit(0);
  }
   return 0;
   }
```

10.8.1 增益控制例 1

下面的例子详细地给出 10.7 节中 ART1 算法的应用,以及上面给出的 ART1 的 C++ 程序执行的计算过程(参见清单 10.2)。

对子这个例子,我们已经对警戒阈值取了一个相对低的值($\rho=3$)。这样会使形成的类别数目比警戒值大的情况要少。警戒值大的情况将在下一个例子中进行讨论。

提供给具有 15 个输入神经元的 ART1 网络的 5 个 3×5 输入模式,如图 10.15 所示。

图 10.15 例 1 中的输入模式

在第一次迭代时,将图 10.15 中的模式(a)提供给已初始化的 ART1 网络。初始化方法在 10.6 节中已讨论过。然后网络进入识别阶段,通过应用 2/3 规则(见公式 10-8)我们可得到矢量 C,这里设 C=X。注意到,利用等式(10-6),在这阶段增益 1 置 1(参见图 10.11)。由于这时不存在已存贮的原型,则在识别层一个未分配的神经元获胜(在这种特殊情况下编号为 0)。结果显示如下:

迭代 1:

获胜神经元:0
输入模式,X: 111100100100111
输出,C: 111100100100111

自上面下权值

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xaut.edu.cn

111100100100111

自下而上权值
0.200000 0.200000 0.200000 0.200000 0.000000
0.000000 0.200000 0.000000 0.000000 0.200000
0.000000 0.000000 0.200000 0.200000 0.200000

在识别层中,第一个神经元被分配到这个类别中;利用等式(10-12),将自上而下的权值设定为输入模式,从而将原型存贮起来。图 10.16 所示为网络状态的示意图。图 10.8 所示为相应形成的类别。

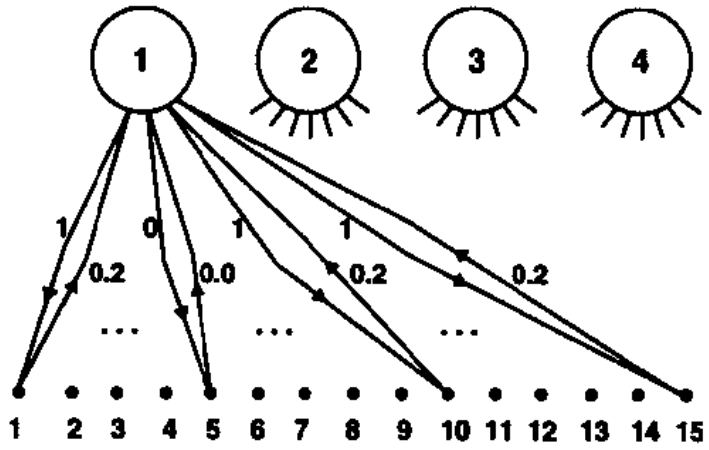


图 10.16 一次迭代后 ART 网络状态举例

在第二次迭代时,将图 10.15 的模式(b)提供给网络。再次将 C 与 X 设定为相同的值。

迭代 2:

获胜神经元:1
输入模式,X: 010010010010010
输出,C: 010010010010010

自上而下权值
111100100100111
010010010010010

自下而上权值
0.200000 0.200000 0.200000 0.200000 0.000000
0.000000 0.200000 0.000000 0.000000 0.200000
0.000000 0.000000 0.200000 0.200000 0.200000
0.000000 0.333333 0.000000 0.000000 0.333333

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and Cooperation: luzhenbo@yahoo.com.cn

0.000000 0.000000 0.333333 0.000000 0.000000
0.333333 0.000000 0.000000 0.333333 0.000000

注意到在迭代 2 中(参见图 10.15(b)),引入网络的模式完全不同于先前存贮的模式(参见图 10.15(a)),从而必须产生一个新的类别。所以,由于识别层神经元已经被分配到一个聚类中心,那么,到现在为止已训练神经元的数目为 2(而不是 1)。

迭代 3:

获胜神经元:0
输入模式,**X**: 101101111101101
输出,**C**: 101100100100101

自上而下权值
101100100100101
010010010010010

自下而上权值
0.250000 0.000000 0.250000 0.250000 0.000000
0.000000 0.250000 0.000000 0.000000 0.250000
0.000000 0.000000 0.250000 0.000000 0.250000
0.000000 0.333333 0.000000 0.000000 0.333333
0.000000 0.000000 0.333333 0.000000 0.000000
0.333333 0.000000 0.000000 0.333333 0.000000

在这个迭代过程中(见图 10.15(c)),引入的模式足以类似于聚类 0 的原型。从而下列情况发生:

1. 识别层中的神经元 0 在识别阶段是获胜者;
2. 神经元 0 的自上而下权值与输入足够类似,可以通过警戒。

结果,神经元 0 的权值(自上而下和自下而上)被修改,并且不再形成另外的聚类中心;即,只有相同的两个识别层神经元被分配。

迭代 4:

获胜神经元:2
输入模式,**X**: 010101101101010
输出,**C**: 010101101101010

自上而下权值
101100100100101

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xidian.edu.cn

010010010010010
010101101101010

自下而上权值
0.250000 0.000000 0.250000 0.250000 0.000000
0.000000 0.250000 0.000000 0.000000 0.250000
0.000000 0.000000 0.250000 0.000000 0.250000
0.000000 0.333333 0.000000 0.000000 0.333333
0.000000 0.000000 0.333333 0.000000 0.000000
0.333333 0.000000 0.000000 0.333333 0.000000
0.000000 0.222222 0.000000 0.222222 0.000000
0.222222 0.222222 0.000000 0.222222 0.222222
0.000000 0.222222 0.000000 0.222222 0.000000

在这个迭代过程中,模式(见图 10.15(d))需要一个新的聚类中心,这个聚类中心导致一个新识别层神经元的分配。连接到这个神经元的权值(在两个权值矩阵中第三行)相应地被更新,以和该模式匹配。

迭代 5:

获胜神经元:0
输入模式,X: 111101100100111
输出,C: 101100100100101

自上而下权值
101100100100101
010010010010010
010101101101010

自下而上权值
0.250000 0.000000 0.250000 0.250000 0.000000
0.000000 0.250000 0.000000 0.000000 0.250000
0.000000 0.000000 0.250000 0.000000 0.250000
0.000000 0.333333 0.000000 0.000000 0.333333
0.000000 0.000000 0.333333 0.000000 0.000000
0.333333 0.000000 0.000000 0.333333 0.000000
0.000000 0.222222 0.000000 0.222222 0.000000
0.222222 0.222222 0.000000 0.222222 0.222222
0.000000 0.222222 0.000000 0.222222 0.000000

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

到此为止,所有五个模式都已被存贮起来。图 10.15 中的模式(a)再一次引入网络,以确保它实属于类别 0。注意到所产生的自上而下和自下而上的权值仍不断被更新,存入的原型与提供给网络的模式则更为相似了。

迭代 6:

获胜神经元:0
输入模式,X: 011100100100011
输出,C: 001100100100001

自上而下权值
001100100100001
010010010010010
010101101101010

b)i

自下而上权值
0.000000 0.000000 0.333333 0.333333 0.000000
0.000000 0.333333 0.000000 0.000000 0.333333
0.000000 0.000000 0.000000 0.000000 0.333333
0.000000 0.333333 0.000000 0.000000 0.333333
0.000000 0.000000 0.333333 0.000000 0.000000
0.333333 0.000000 0.000000 0.333333 0.000000
0.000000 0.222222 0.000000 0.222222 0.000000
0.222222 0.222222 0.000000 0.222222 0.222222
0.000000 0.222222 0.000000 0.222222 0.000000

对应于自上而下和自下而上权值的网络如图 10.17 所示。图 10.18 所示为每个原形模式矢量形成的类别。

10.8.2 增益控制举例 2

在这个例子中,我们说明提高警戒阈值所产生的影响。我们还使用例 1 中的模式(参见图 10.15),但是设警戒阈值 $\rho=0.8$。我们已经观察过,这种情况下形成的类别数目增多了,而且在新定义的类别之中模式的分布会不同。

最优神经元:0
输入:111100100100111
输出:111100100100111

自上而下权值
111100100100111

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

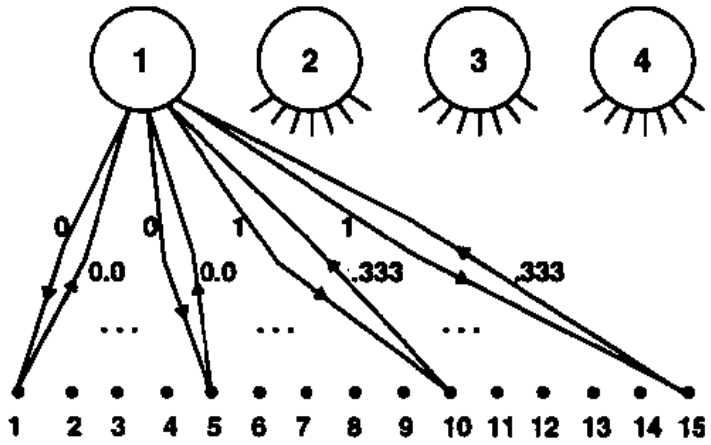


图 10.17 对所有模式训练后的 ART 网络的状态(警戒阈值 $\rho=0.3$)

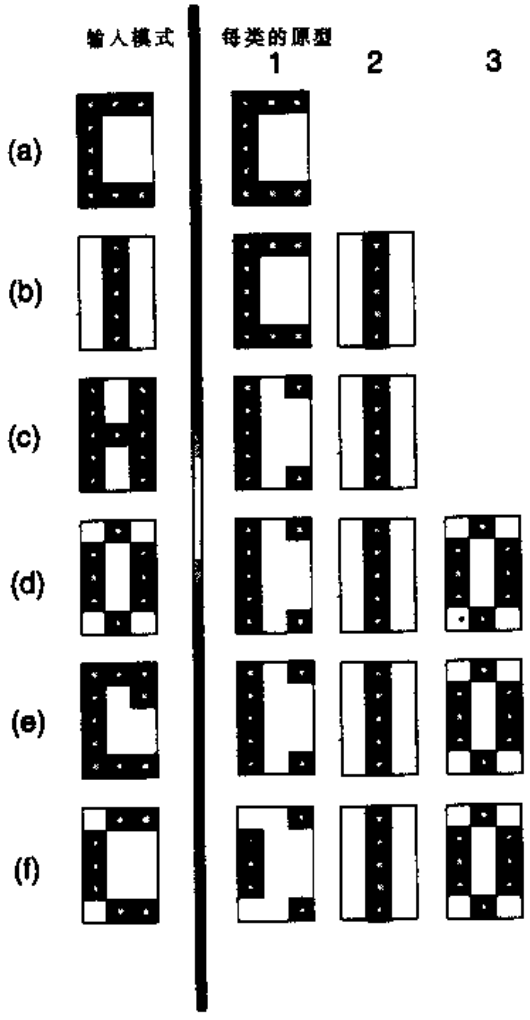


图 10.18 例 1 中的形成的类别

自下而上权值

0.200000 0.200000 0.200000 0.200000 0.000000

0.000000 0.200000 0.000000 0.000000 0.200000

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

0.000000 0.000000 0.200000 0.200000 0.200000

最优神经元:1
输入:0100100100100100
输出:0100100100100100

自上而下权值
111100100100111
0100100100100100

自下而上权值
0.200000 0.200000 0.200000 0.200000 0.000000
0.000000 0.200000 0.000000 0.000000 0.200000
0.000000 0.000000 0.200000 0.200000 0.200000
0.000000 0.333333 0.000000 0.000000 0.333333
0.000000 0.000000 0.333333 0.000000 0.000000
0.333333 0.000000 0.000000 0.333333 0.000000

最优神经元:2
输入:101101111101101
输出:101101111101101

自上而下权值
111100100100111
0100100100100100
101101111101101

自下而上权值
0.200000 0.200000 0.200000 0.200000 0.000000
0.000000 0.200000 0.000000 0.000000 0.200000
0.000000 0.000000 0.200000 0.200000 0.200000
0.000000 0.333333 0.000000 0.000000 0.333333
0.000000 0.000000 0.333333 0.000000 0.000000
0.333333 0.000000 0.000000 0.333333 0.000000
0.166667 0.000000 0.166667 0.166667 0.000000
0.166667 0.166667 0.166667 0.166667 0.166667
0.000000 0.166667 0.166667 0.000000 0.166667

最优神经元:3
输入:0101011011010100

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

输出:0101011011010 10

自上而下权值
111100100100111
010010010010010
101101111101101
010101101101010

自下而上权值
0.200000 0.200000 0.200000 0.200000 0.000000
0.000000 0.200000 0.000000 0.000000 0.200000
0.000000 0.000000 0.200000 0.200000 0.200000
0.000000 0.333333 0.000000 0.000000 0.333333
0.000000 0.000000 0.333333 0.000000 0.000000
0.333333 0.000000 0.000000 0.333333 0.000000
0.166667 0.000000 0.166667 0.166667 0.000000
0.166667 0.166667 0.166667 0.166667 0.166667
0.000000 0.166667 0.166667 0.000000 0.166667
0.000000 0.222222 0.000000 0.222222 0.000000
0.222222 0.222222 0.000000 0.222222 0.222222
0.000000 0.222222 0.000000 0.222222 0.000000

最优神经元:0
输入:111100100100111
输出:111100100100111

自上而下权值
111100100100111
010010010010010
101101111101101
010101101101010

自下而上权值
0.200000 0.200000 0.200000 0.200000 0.000000
0.000000 0.200000 0.000000 0.000000 0.200000
0.000000 0.000000 0.200000 0.200000 0.200000
0.000000 0.333333 0.000000 0.000000 0.333333
0.000000 0.000000 0.333333 0.000000 0.000000
0.333333 0.000000 0.000000 0.333333 0.000000
0.166667 0.000000 0.166667 0.166667 0.000000

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

0.166667 0.166667 0.166667 0.166667 0.166667
0.000000 0.166667 0.166667 0.000000 0.166667
0.000000 0.222222 0.000000 0.222222 0.000000
0.222222 0.222222 0.000000 0.222222 0.222222
0.000000 0.222222 0.000000 0.222222 0.000000

最优神经元:0
输入:011100100100011
输出:011100100100011

自上而下权值
011100100100011
010010010010010
101101111101101
010101101101010

自下而上权值
0.250000 0.000000 0.250000 0.250000 0.000000
0.000000 0.250000 0.000000 0.000000 0.250000
0.000000 0.000000 0.250000 0.000000 0.250000
0.000000 0.333333 0.000000 0.000000 0.333333
0.000000 0.000000 0.333333 0.000000 0.000000
0.333333 0.000000 0.000000 0.333333 0.000000
0.166667 0.000000 0.166667 0.166667 0.000000
0.166667 0.166667 0.166667 0.166667 0.166667
0.000000 0.166667 0.166667 0.000000 0.166667
0.000000 0.222222 0.000000 0.222222 0.000000
0.222222 0.222222 0.000000 0.222222 0.222222
0.000000 0.222222 0.000000 0.222222 0.000000

可以观察到例 2 中形成了另外一个类别(类别 3)。此外,我们还可以观察到,类别 3(在前而的例子中属于类别 0)现在也包含模式 5(在例 1 中也分配到类别 0 中)。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

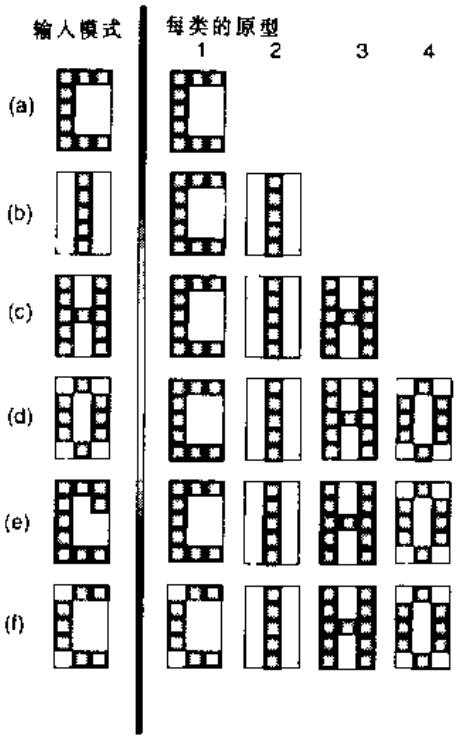


图 10.19 例 2 中形成的类别

10.9 ART2 模型

ART2 的结构是在 ART1 思想(具有两层的结构,和在两个方向可修改的权值,即前馈和反馈权值)的基础上建立的。ART2 和 ART1 均包含一个注意子系统和一个取向子系统。同 ART1 类似,ART2 的取向子系统由比较层 F1 和识别层 F2 构成,并且执行和 ART1 相同的功能。

为了处理具有连续值分量的模式,对 ART1 做了一些修改。图 10.20 所示为 ART2 的结构,其中 ART1 的比较层被具有多个神经元集的多层结构取代。注意到比较层(或 F1 层)已经分成了几层。另外,取向子系统已经被修改以便处理实数值数据。如图 10.20 所示,ART2 设计增加了节点,它们的功能包括:

1. 允许噪音抑制;
2. 标准化,即通过对比,增强模式的重要部分;
3. 自上而下和自下而上信号的比较,以用于复位机制;
4. 处理实数值数据,这些实值数据彼此可以任意地靠近。

因此,ART2 中的预处理比 ART1 网络中的预处理复杂得多。

虽然 ART2 网络和 ART1 网络相比看起来更复杂,但 ART2 的学习规则实际上比 ART1 更简单。对 ART2 来说,由于 **X** 和 **C** 不是二进制的,那么用于 ART1 的匹配标准 S(即相似度)不再有效。ART2 则是利用距离测度,这种测度方法通常用于统计模式识别技术(也称 MINNET 技术。前面我们已经在矢量量化、K-均值聚类算法和 Kohonen 网络中看到了这种距离测度)。输入矢量和原型矢量之间夹角的余弦值,用于和警戒阈值进行比较。

关于 ART2 及其算法的更详细地讨论,请参阅 Carpenter 和 Grssberg[1987b]的文献。实

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

现 ART2 网络的 C 语言程序,可以参见 Freeman 和 Skpura[1992]的著作。

10.10 讨 论

ART 与竞争模式识别技术相比,具有以下一些优点:

1. 许多定理已经证明[Carpenter 和 Grossberg, 1987a],ART 具有相当好的稳定性,而且它不会被大量任意的输入所干扰。

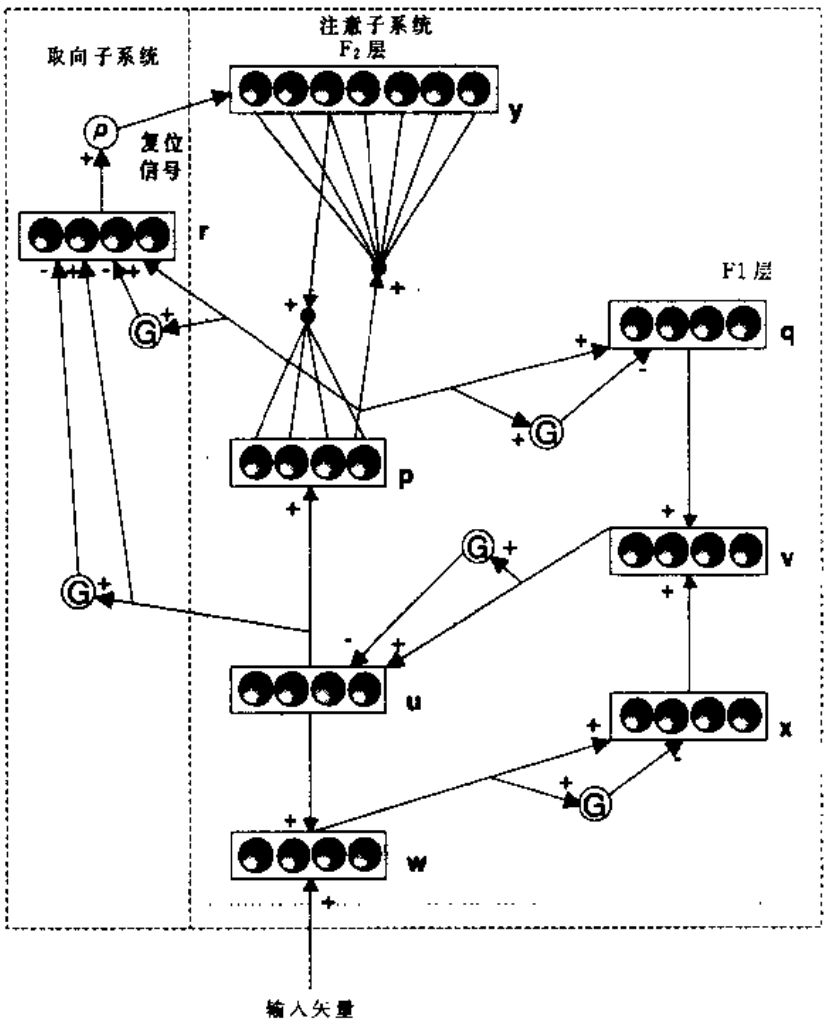


图 10.20 ART2 结构

2. 网络可以通过适当地更新类别原型,以自适应地反映环境中最常见的模式类型。

3. ART 结构很容易就能与其它分层认知理论结合起来。它考虑到了来自 ART 网络外围的其它子系统(诸如注意子系统、预测子系统或取向子系统)对特性和分类层的影响。

ART 结构对模式进入网络的先后顺序很敏感。Kung[1993]已经通过实验证明,当模式进入网络的顺序相反时,ART2 对同一个输入模式集产生不同的聚类。

表 10.1 所示为 ART2 对一个由十个输入组成的模式集合进行分类的情况[Kung, 1993]。其中,对于第三个和第五个模式,警戒测试失败并且形成了新的类别。从而,对警戒阈值 1.5,形成了三个类别。图 10.21 所示为实际模式的模式空间。这些模式按相反的顺序进入网络时

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

表 10.1 ART2 的执行顺序

顺序	模式	胜者	测试值	判决	类别 1 中心	类别 2 中心	类别 3 中心
1	(1.0,0.1)	–	–	新的聚类	(1.0,0.1)		
2	(1.3,0.8)	1	1.0	通过警戒测试	(1.15,0.45)		
3	(1.4,1.8)	1	1.6	失败→新的聚类		(1.4,1.8)	
4	(1.5,0.5)	1	0.4	通过警戒测试	(1.27,0.47)		
5	(0.0,1.4)	2	1.8	失败→新的聚类			(0.0,1.4)
6	(0.6,1.2)	3	0.8	通过警戒测试			(0.3,1.3)
7	(1.5,1.9)	2	0.2	通过警戒测试		(1.45,0.85)	
8	(0.7,0.4)	1	0.63	通过警戒测试	(1.30,0.45)		
9	(1.9,1.4)	2	0.9	通过警戒测试		(1.6,1.7)	
10	(1.5,1.3)	2	0.5	通过警戒测试		(1.58,1.6)	

(见表 10.2),采用了不同的存贮原型值,警戒测试只失败了一次(对第三个模式),并且只形成两个类别。图 10.22 所示为相应于这个顺序所形成的实际聚类情况。注意到在这两种情况下,警戒阈值相同。

表 10.2 模式以相反的顺序进入 ART2

顺序	模式	胜者	测试值	判决	类别 1 中心	类别 2 中心
1	(1.5,0.3)	–	–	新的聚类	(1.5,0.3)	
2	(1.9,1.4)	1	1.5	通过警戒测试	(1.7,1.35)	
3	(0.7,0.4)	1	1.95	失败→新的聚类		(0.7,0.48)
4	(1.5,1.9)	1	0.75	通过警戒测试	(1.63,1.53)	
5	(0.6,1.2)	2	0.9	通过警戒测试		(0.65,0.8)
6	(0.0,1.4)	2	1.25	通过警戒测试		(0.43,1.0)
7	(1.5,0.5)	1	1.17	通过警戒测试	(1.6,1.28)	
8	(1.4,1.8)	1	0.72	通过警戒测试	(1.56,1.38)	
9	(1.3,0.8)	1	0.84	通过警戒测试	(1.52,1.28)	
10	(1.0,0.1)	2	1.47	通过警戒测试		(0.58,0.78)

ART 结构应该支持可塑性,即要求能够学习新模式,而阻止对先前已学习过的模式的修改。然而,对 ART1 来说,它的编码稳定性问题还没有完全解决。如果网络接受到许多已存输入模式的变形,那么它能在模式空间中逐渐移向给定的方向。每一个变形都能与先前已存贮的类别原型紧密匹配,而被放入同一类别中。这会导致足够大的偏移,甚至使网络可能识别不出原来的模式。图 10.23 示出了对于一组五个字符的带有噪声的输入模式,这种情况是如何发生的。Ryan 和 Winter[1987]已经提出了一种 ART 的变形,用于克服这一局限性。

警戒值在处理类别和特征的过程中一直保持不变,这可能会导致产生另一个问题。例如,为把"O"和"Q"区分开来所设的警戒值,可能会不能容忍其它字符的噪声,从而会终止区分"a"和"a"。Weingard[1990]提出了一种 ART 的变形,它允许动态设置警戒参数。

ART 的另一种特征,是它总能对某一输入模式属于哪个类别作出判决。与许多其它的模式识别技术不同,它不记录输入中存在的不明确性(即输入与类别之间的边界有多近)。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

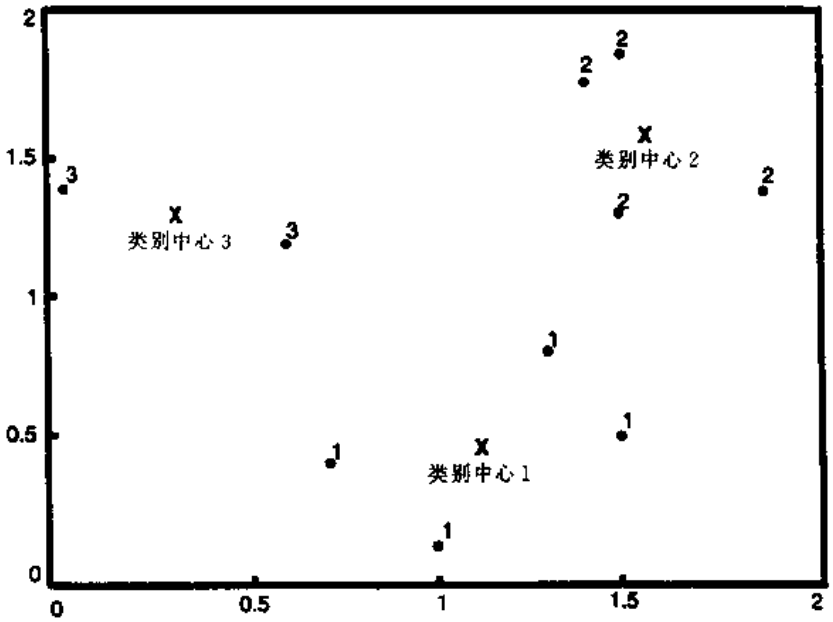


图 10.21 模式以原顺序进入网络的 ART 结果

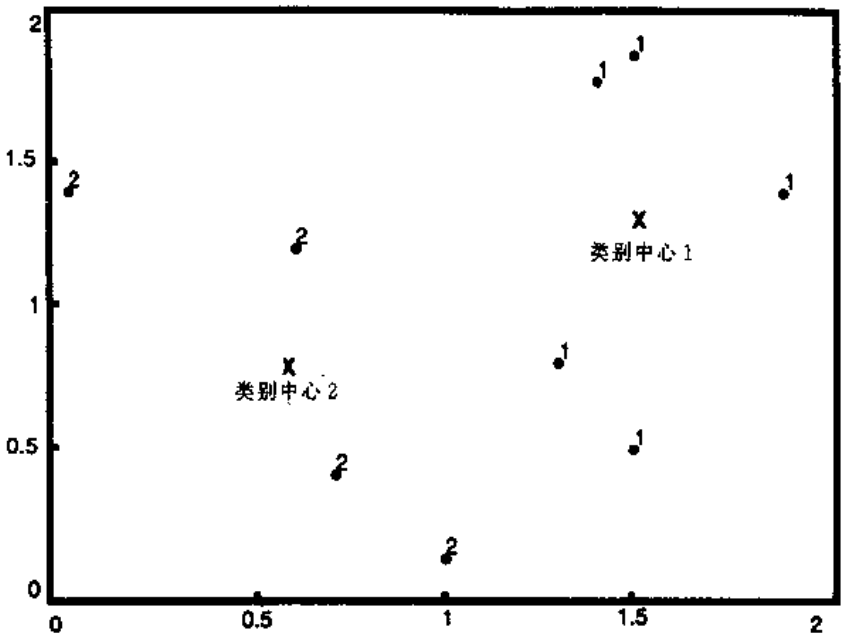


图 10.22 模式以相反顺序进入网络的 ART 结果

Levine[1989]和 Penz[1990]研究了一种解决这一问题的改进模型,用于记录这种不明确性。

通过适当选择警戒参数值,可以改变 ART 系统区分不同输入模式类别的能力。因此,输入模式属于哪一类可以由警戒因子来决定。对给定的一组输入模式,警戒阈值大,则使类别之间的区分更细。即,警戒阈值越大,产生的类别越多。另一方面,警戒阈值小些,会把更多带噪声的模式分到相同的类别中。图 10.24 和 10.25 显示出了对于不同的警戒阈值 ART2 的结果[Kung,1993]。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

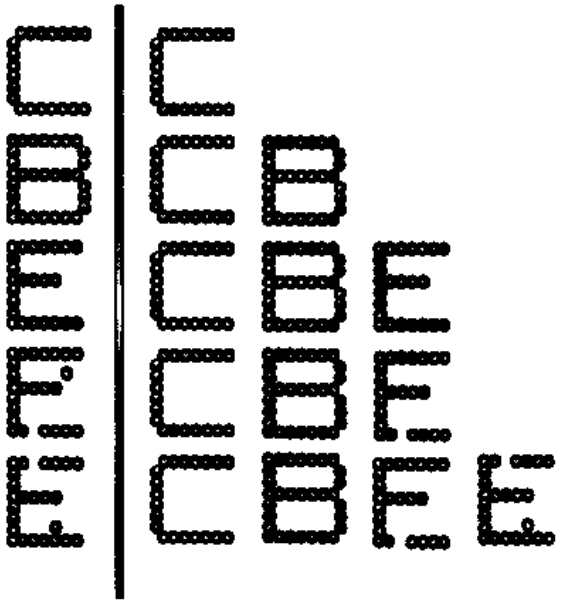


图 10.23 由五个带噪音模式训练的 ART 网络的变化情况

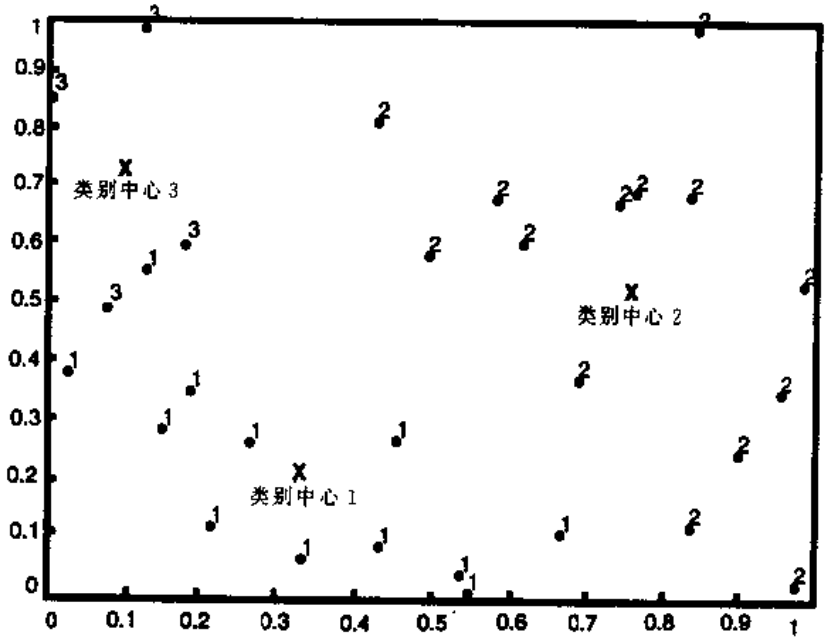


图 10.24 警戒值 $\rho=0.6$ 时 ART 的结果(摘自 Kung[1993])

10.11 应 用

Carpenter 和 Grossberg[1987b]已经将 ART2 结构应用于模拟模式的分类问题。图 10.26 显示出了 50 个输入模式,ART2 把它们分成 34 个类别。

Gan 和 Lua[1992]将 ART2 结构应用于中文字符识别问题。选择 ART2 是因为中文字符识别使用的是一个实值特征集。该特征集由 12 个几何特征组成,包括交叉点、转折点和横竖笔划等。在这个应用中,ART2 网络没有用来做最后的分类器,而是用来把 3755 个汉字分到 7 个组或类别中,以便为最后的识别阶段做准备。实现 ART 的一些问题,包括快速与慢速训

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication and cooperation: luzhenbo@yahoo.com.cn

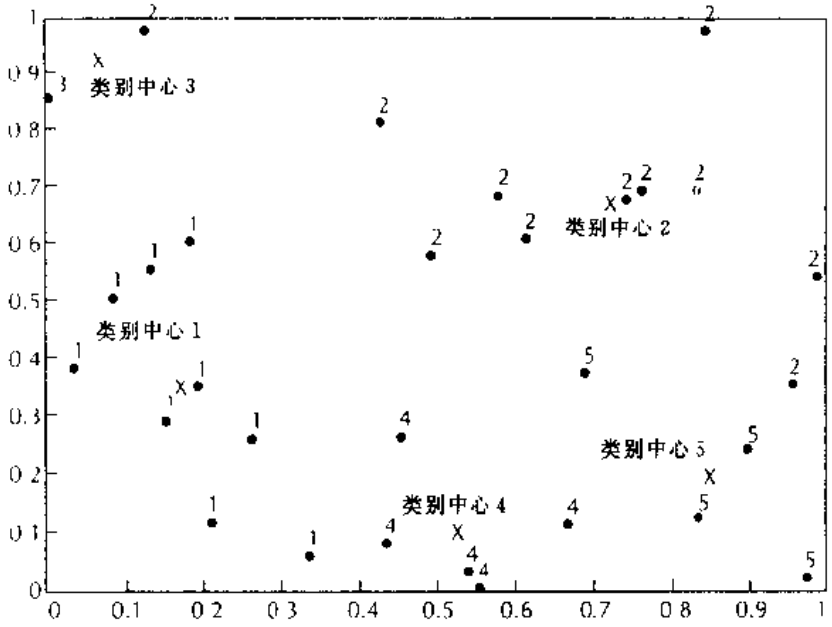


图 10.25 警戒值 $\rho=0.4$ 时 ART 的结果(摘自 Kung[1993])

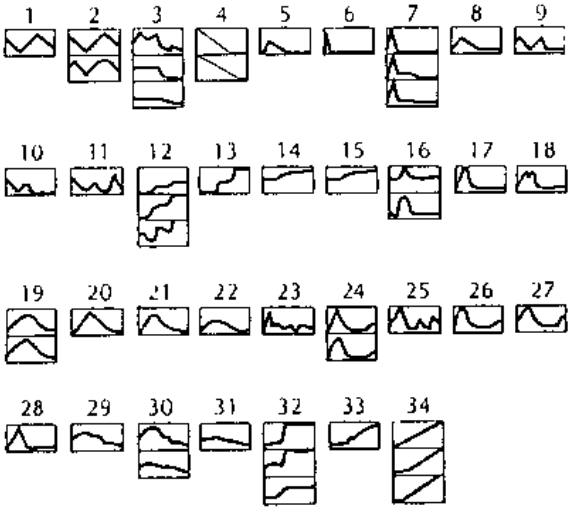


图 10.26 ART2 将 50 个模拟输入模式分到 34 个类别中的例子

练、阈值选择和为自下而上层提供初始值。据报道,在最好的情况下,训练集分类精度可达 97.23%,测试集精度可达 90.20%。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

参考书与文献

Carpenter, G. A. and Grossberg, S., "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Comput. Vision, Graphics Image Process.*, Vol. 37, pp. 54–115, 1987a.

Carpenter, G. A. and Grossberg, S., "ART 2: self-organization of stable category recognition codes for analog input patterns," *App. Optics*, vol. 26, pp. 4919–4930, 1987b.

Carpenter, G. A. and Grossberg, S., "ART 3: hierarchical search using chemical transmitters in self-organizing pattern recognition architecture," *Neural Networks*, Vol. 3, pp. 129–152, 1990.

Freeman, J. A. and Skapura, D. M., *Neural Networks: Algorithms, Applications and Programming Techniques*, Addison-Wesley, Reading, MA, 1992.

Gan, K. W. and Lua, K. T., "Chinese character classification using adaptive resonance network," *Pattern Recognition*, vol. 25, no. 8, 1992.

Grossberg, S., "Adaptive pattern classification and universal recoding:parallel development and coding of neural feature detectors," *Biol. Cybern.*, vol. 23, pp. 187–202, 1976.

Grossberg, S., "Competitive learning: from interactive activation to adaptive resonance," *Cognitive Sci.*, vol. 11, pp. 23–63, 1987.

Kung, S. Y., *Digital Neural Networks*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

Levine, D. S., "Selective vigilance and ambiguity detection in adaptive resonance networks," In W. Webster (Ed.), *Simulation and AI*, Society for Computer Simulation, San Diego, CA, pp. 1–7, 1989.

Levine, D. S. and Penz, P. A., "ART 1.5 — a simplified adaptive resonance network for classifying low dimensional analog data," *Proc. of the Int. Joint Conf. Neural Networks*, vol. 2, pp. 639–642, 1990.

Pao, Y. H., *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley, Reading, MA, 1989.

Ryan, T. W. and Winter, C. L., "Variations on adaptive resonance," *Proc. IEEE First Int. Conf. Neural Networks*, vol. 2, pp. 767–776, 1987.

Tou, J. T. and Gonzalez, R. C., *Pattern Recognition Principles*, Addison-Wesley, Reading, MA, 1974.

Weingard, F. S., "Self-organizing analog fields (SOAF)," *Proc. Int. Joint Conf. Neural Networks*, vol. 2, pp. 34–38, 1990.

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

第十一章 神经认知机

11.1 引 言

到目前为止,我们所讨论的神经网络,都是适合于进行模式识别问题的普通网络。与此相对应,神经认知机是为了处理手写体字符识别问题而特意提出来的。此网络是多层的分级神经网络。每层的神经元都是相同类型的,它们或是简单的,或是复杂的,或是超复杂的神经元,在每层之间都有非常稀少并且固定模式的联接。这种模型是受到由 Hubel 和 Wiesel[1962]提出的生物视觉神经系统模型启发而提出的。我们对神经认知机的讨论,大部分是基于 Fukushima 和 Wake [1991]所提出的模型。与大多数神经网络不同,神经认知机对字符畸变、字符位移具有较高的容错性,并且对字符的大小具有至少是中等的免疫性。

11.2 网络的结构

神经认知机是分级组织的,每一级包括两层。如图 11.1,每一级包括两层不同的神经元,一层为简单的神经元或称作 S 神经元;另一层为复杂的神经元或称作 C 神经元。

我们应注意到,神经元层又可分成若干子神经元平面。每一个 S 神经元子平面,负责识别一个特定的特征。因此同一个子平面内的 S 神经元都具有相同的权值。在同一个子平面内的 S 神经元之间的差别是:每一个 S 神经元的输入是来自前一层的不同位置,这和第六章所讨论的特征图中的感受野相类似(见 6.3 节)。第一个 S 神经元层的输入是光感受器的象素位图,这一层只是提取相对简单的一些特征。在接下来的 S 神经元层提取更为复杂一些的特征。就这样随着一层一层的增加,提取的特征也随着增加。

图 11.2 是神经认知机各层之间相连的草图。图 11.2(a)示出了向下一层提供兴奋输入的 S 和 C 神经元。注意到,每一 S 层都包括 S 神经元以及向同一层的 S 神经元提供抑制输入的 V 神经元。图 11.2(b)示出了由 Fukushima 和他的同事提出的一个实际系统的结构,这个系统是用于进行识别字母数字式的字符(大字和数字)。注意到每一个 S1 层神经元都有三个输入连接,所以感受野为 3×3;对于 S2—S4 层的神经元,有五个输入连接故感受野为 5×5;C1 层的每个神经元有三个连接;C2 层的神经元有七个连接等等。

C 神经元的输入,来自前一 S 神经元层相对应的子平面中的一组 S 神经元。因为在同一子平面中的不同 S 神经元检测不同位置的相同特征,所以 C 神经元就对输入层的特征的精确位置不太敏感。这样,神经认知机系统就可不考虑光学感受器的位移而识别出字符来。此系统对畸变字符的识别也具有一定的灵活性。

第一特征图层检测的只是非常简单的特征。譬如一些具有不同方向的直线,而且这些特

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

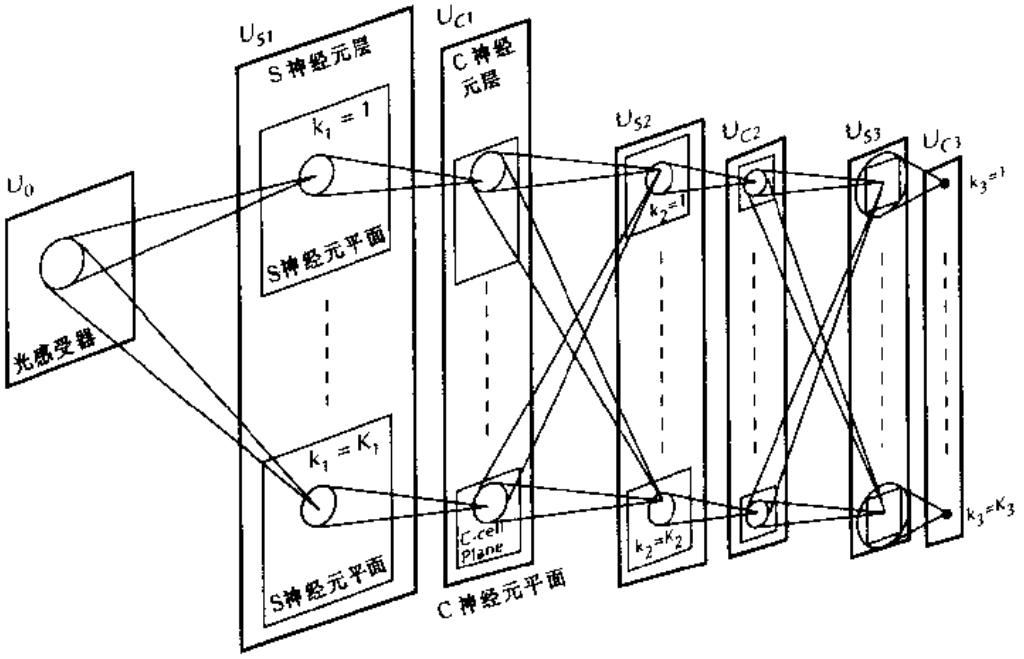


图 11.1 神经认知机的分级结构，每层 S 神经元和 C 神经元的数量标于底部(摘自 Fukushima，K．和 Wake，N．，IEEE Trans. Neural Network ，vol 2，no.3，pp.355－365，1991．)

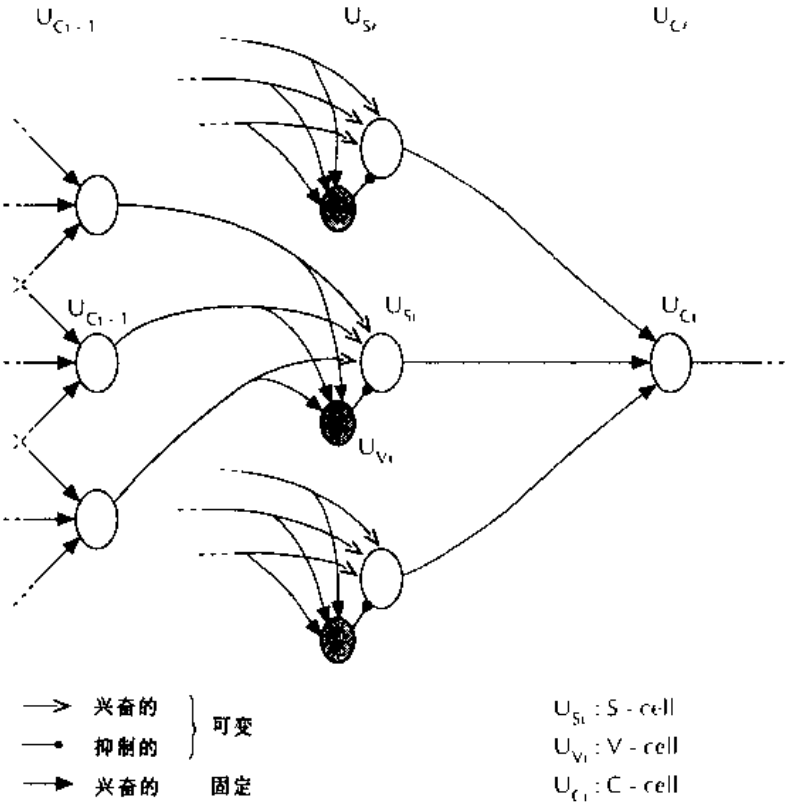


图 11.2(a) 神经认知机中的不同类型的神经元之间的连接(摘自 Fukushima，K．和 Wake，N．，IEEE Trans. Neural Networks，vol.2，no.3，pp.355－365，1991．)

征只能在 S 神经元感受野的有限边界内才可观察到。接下来的层就能将这些简单的特征转换

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

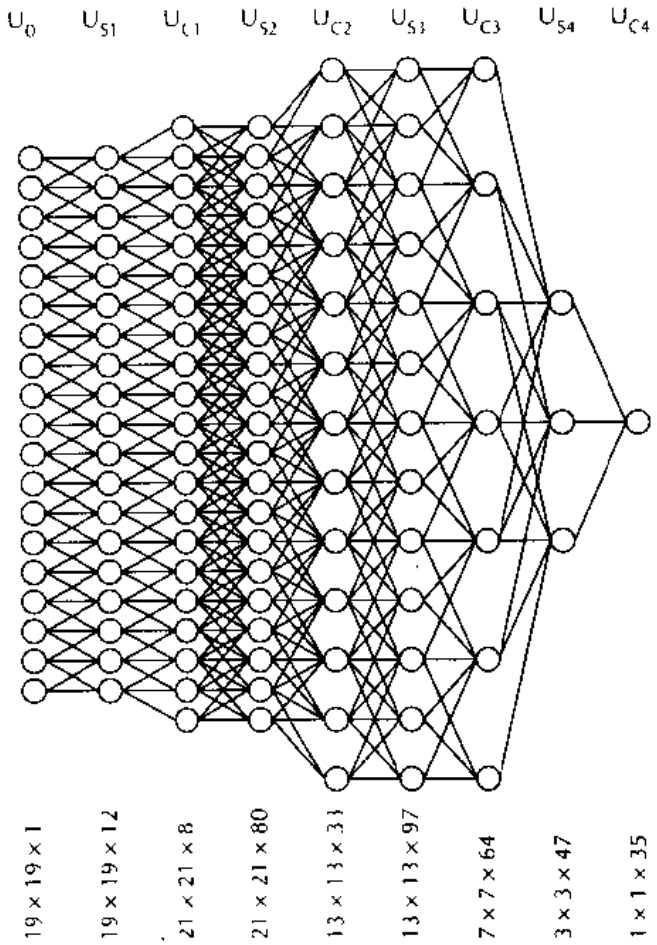


图 11.2(b) 每层神经元之间连接的剖面图,注意到每一个子平面中只画出了一个神经元(摘自 Fukushima,K . 和 Wake,N .,IEEE Trans. Neural Networks,vol.2,no.3,pp.355－365,1991.)

成逐渐复杂的特征,示于图 11.3。

一 S 神经元的输出为:

$$u_{Sl}(k_l,\mathbf{n})=l\Phi\left[\frac{1+\sum_{k_{l-1}=1}^{K_{l-1}}\sum_{\upsilon\in A_l}a_l(k_{l-1},\upsilon,k_l)U_{C_{l-1}}(k_{l-1},\mathbf{n}+\upsilon)}{1+\frac{r_l}{1+r_l}b_l(k_l)V_{C_l}(\mathbf{n})}-1\right] \tag{11-1}$$

上式中的 Φ 是线性阈值函数。

$$\Phi(x)=\begin{cases}x & x\geq 0\\ 0 & x<0\end{cases} \tag{11-2}$$

我们使用下列符号表示法:

$U_{Sl}(k_l,\mathbf{n})$是 S_l 层的 S 神经元的输出。

k_{l-1}是如图 11.2(b)底部示出的每个子神经元平面的大小。

$a_l(k_{l-1},\upsilon,k_l)$是 C 神经元的可调兴奋权值。

$U_{C_{l-1}}(k_{l-1},\mathbf{n}+\upsilon)$是来自前层的 C 神经元的输入。

$b_l(k_l)$是附属的 V 神经元的可调抑制权值。

$V_{C_l}(\mathbf{n})$是来自附属的 V 神经元的输入。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

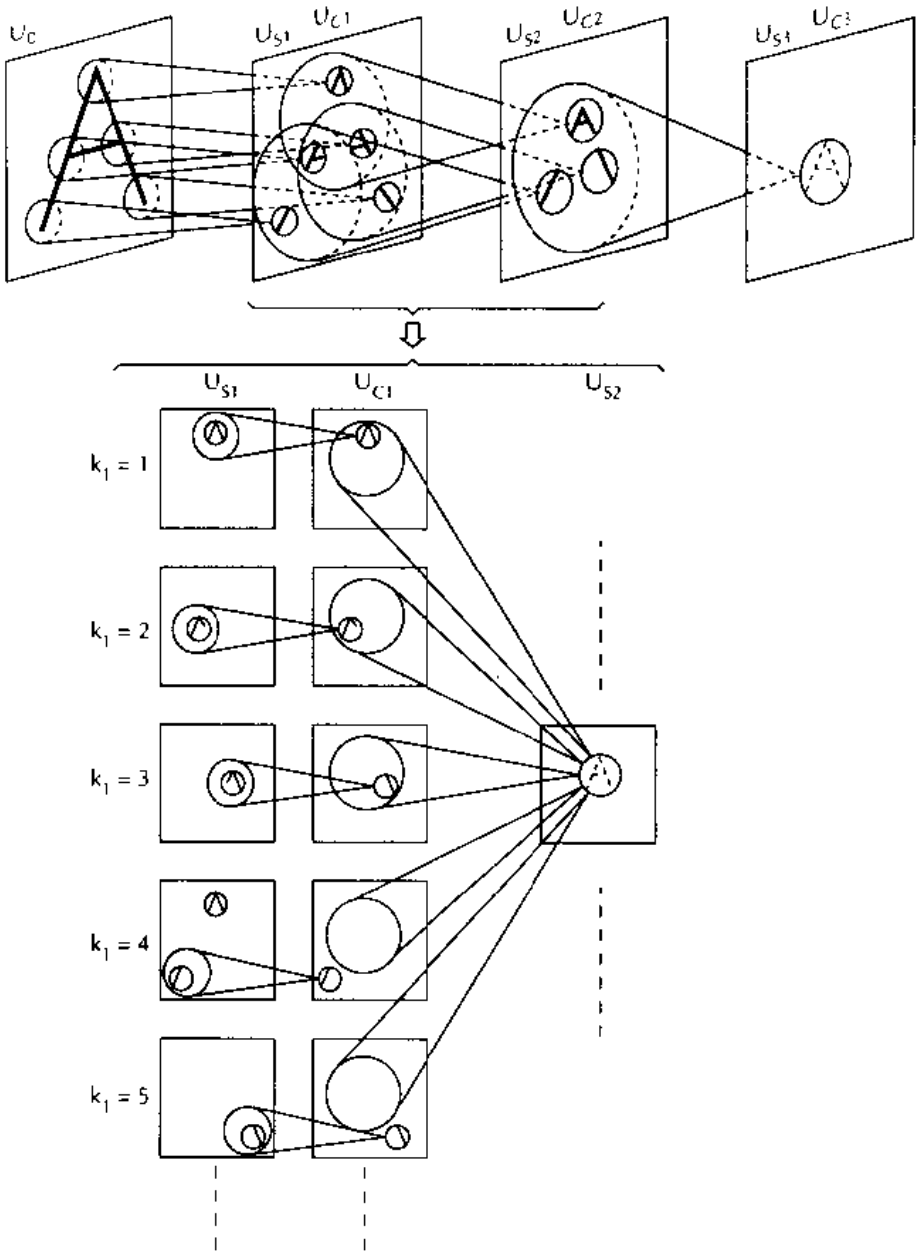


图 11.3 训练完的神经认知机的每一层对样本模式的反应。不同层的神经元之间的互连(摘自 Fukushima, K ., Biol .Cybern ., vol.36, pp.193－202, 1980.)

A_l 确定 S 神经元要提取的特征大小。图 11.2(b)中 $A_1 = 3 \times 3$ 以及 $A_2 - A_4 = 5 \times 5$。

r_l 控制特征提取的选择性。r_l 越大就意味着对噪音和特征畸变有越差的容错性。

式(11-1)k_l 中指的是第 l 级的第 k 个子平面，n 是在子平面内确立一个神经元的二维矢量，v 也是一个矢量并且它是位于 n 感受野中的前层的神经元 n 的相对位置，A_l 代表 n 的感受野。因此，式中里面对 v 的求和？也就包含了指定区域当中的所有神经元；外面对于 k_{l-1} 的求和，也就包含了前一级的所有子平面。因此在分子中的求和项有时也称作兴奋项或 e，实际上只是乘积的和。输出到 n 的那些神经元的输出都乘上它们相应的权值然后输出到 n_c

r_l 项是一常量，它能控制位于每一 S 层处的单个抑制子平面中的每个神经元的输入。每个 V_C 神经元的输出，都送到与 V_C 子平面内相对应位置处的所有 S 子平面内的所有神经元

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

(图 11.4 示出了和第一个 S 层相关的子平面以及它的前一层,在图中也就是光学感受器)。r_l 的值越大,与抑制性成比例的兴奋性就得越大,以便能产生一个非零输出。也就是说,相当好的匹配才能使神经元兴奋。然面因为 r_l 还乘以 $\phi()$,所以 r_l 值越大就能产生比较大的输出。相反,小的 r_l 值就能允许神经元对不太好的匹配也能兴奋,但它产生一个比较小的输出。

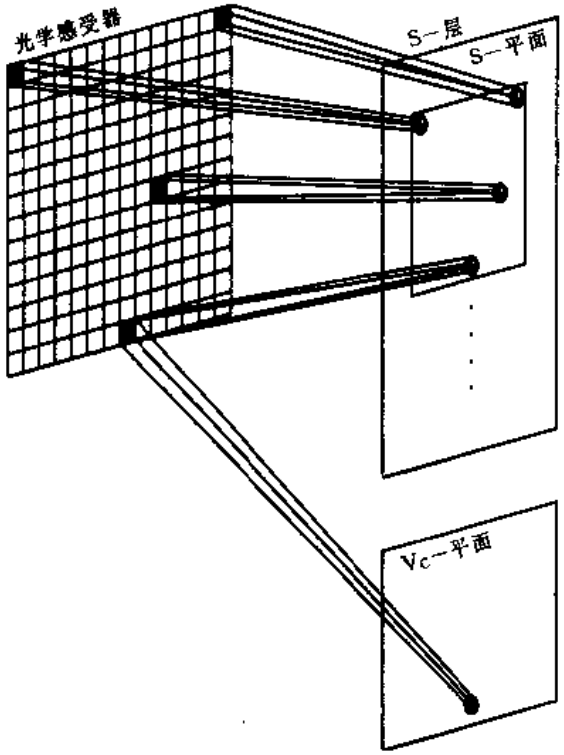


图 11.4 光学感受器和第一 S 神经元层的连接图

位于 n 处的抑制性 V_c 神经元的输出为:

$$V_{c_l}=\sqrt{\sum_{k_{l-1}=1}^{k_{l-1}}\sum_{v\in A_l}c_l(v)U_{c_{l-1}}^2(k_{l-1},\mathbf{n}+\mathbf{v})} \tag{11-3}$$

权 $c_l(\mathbf{v})$是与位于 V_C 神经元感受野中的 v 处的神经元相连的权值。我们不训练这些权值,但它们应随着 v 的幅度增加而单调减小。由此,我们选择下面这种权值,其中 $r'(\mathbf{v})$是从 v 处到感受器中心的规一化距离。

$$c_l=\frac{1}{c_l}(l)\alpha_l^{r'(v)} \tag{11-4}$$

上式中的规一化常量 $C(l)$由下式给出:

$$C(l)=\sum_{k_{l-1}=1}^{K_{l-1}}\sum_{v\in A_1}\alpha_l^{r'(v)} \tag{11-5}$$

一个 C 神经元的输出由下式给出:

$$U_{c_l}(k_l,\mathbf{n})=\Psi\left[\frac{1+\sum_{k_{l-1}=1}^{K_1}j_l(k_l,k_{l-1})\sum_{v\in D_1}d_l(\mathbf{v})U_{s_l}(k_l,\mathbf{n}+\mathbf{v})}{1+V_{s_l}(\mathbf{n})}-1\right] \tag{11-6}$$

上式中(x)由(11-7)式给出:

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

$$\Psi(x)=\begin{cases}\dfrac{x}{\beta+x} & x\geq 0\\ 0 & x<0\end{cases} \tag{11-7}$$

这里 β 是一常量。

K_l 是第 l 级中的 S 子平面的数量。D_l 是 C 神经元的感受野。因此，它和特征的大小相对应。在图 11.1 中，$D_1=3\times3$，$D_2=5\times5$，$D_3=7\times7$ 以及 $D_4=3\times3$。$d_l(\mathbf{v})$是固定兴奋连接权的权值，它是$|\mathbf{v}|$的单调递减函数。如果第 k_l 个 S 神经元子平面从第 k_{l-1}子平面处收到信号那么 $j_l(k_{l-1},k_l)$的值为 1，否则为 0。

最后，S 层的 V_S 神经元的输出为

$$V_{S_l}=\frac{1}{K_l}\sum_{k_{l-1}=1}^{K_{l-1}}\sum_{v\in D_l}d_l(\mathbf{v})U_{s_l}(k_l,\mathbf{n}+\mathbf{v}) \tag{11-8}$$

要训练的权值只有 a_l 和 b_l。如果所需要的特征已预先确定，那么这些权值的训练就可以是有监督的。这种情况下，每一个 S 层都将单独学习（详细情况请见 Fukushima 和 Wake [1991]）。

另一种情况，就是可以像下述那样使用无监督学习（详细情况请见 Fakushima[1980]）那些兴奋子平面的权值 a_l 初始化为一小的正的随机值。权值 b_l 初始化成 0。在学习样本输入后，选出不同子平面输出最大的子神经元作为多个子平面的“代表”。抑制性子平面的代表也以类似方式选出。然后使用下式来更新各个权值：

$$\Delta a_l(k_{l-1},\mathbf{v},\dot{k}_l)=q_lC_{l-1}(\mathbf{v})U_{c_{l-1}}(k_{l-1},\dot{\mathbf{n}}+\mathbf{v})$$

$$\Delta b_l(\dot{k}_l)=q_lC_{l-1}(\dot{\mathbf{n}}) \tag{11-9}$$

q_l 为正常数，其大小决定了学习的速度。

上式中 $U_{C_{l-1}}(k_{l-1},\mathbf{n}+\mathbf{v})C$ 是神经元的响应，$V_{c_{l-1}}(\mathbf{n})$是 V 神经元的响应。它们都由输入到输入层的学习样本决定。在有监督学习的情况下，网络是一层一层地学习。因此从输入单元到 S_1 单元的权值首先被训练，然后权值不变。其后训练从 C_1 单元到 S_2 单元的权值，再固定权值以此类推，最后到达输出层。

据 Fukushima 和 Wake(1991)的报道，当有监督训练完后，神经认知系统能产生一个较好的效果。然而，报道也指出在有监督训练的情况下，相应的参数选取需要艰苦的工作，并且当学习的样本数量增加时参数选取的困难性大幅增加。

11.3 神经认知机的一个例子

Fukushima 和 Wake 在 1991 年提出了一种网络，用来识别大写的字母数字式字符。在这种级连网络的中间过程，使用有监督学习来控制要提取的的选择。局部特征可以像下述方式那样，通过学习输入样本来确定。

图 11.5 示出 S_1 层的 12 个学习样本，每个样本对应着一个子平面。从图 11.3 中可看出，此层每个神经元都有一个 3×3 的感受野。我们训练它们来提取一定方向的线段特征。S_2 层有 80 个神经元子平面，每个子平面都具有如图 11.3 所示的 9×9 感受野。图 11.6 示出了由典型畸变样本所组成的 80 个学习样本。这些畸变样本都是数字字母字符的一部分。S_3 层通过将前几层所提取的局部特征进行组合，来提取全局特征。图 11.8 画出了 47 个训练样本，这

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

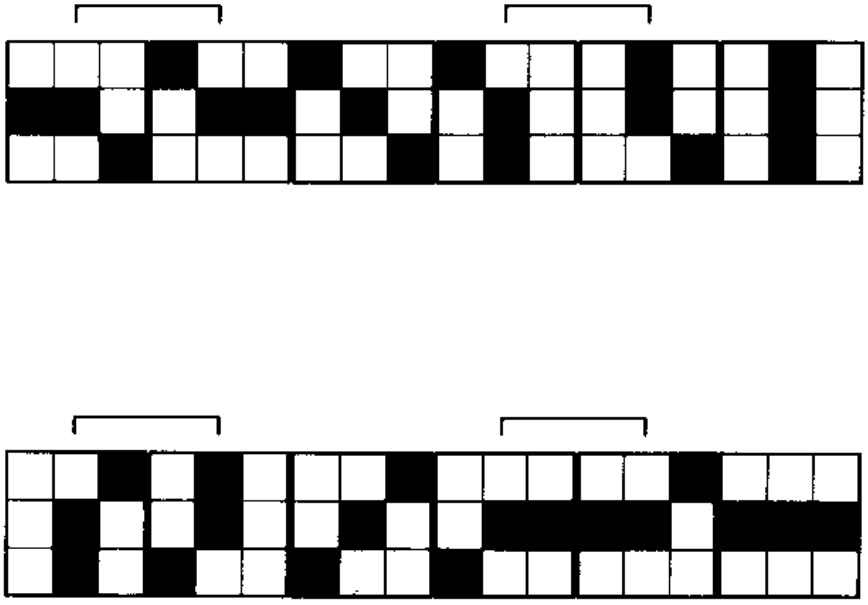


图 11.5 S1 层子平面的训练模板(摘自 Fukushima,K. 和 Wake,N.,
IEEE Trans.Neural Network,vol.2,no.3,pp.355－365,1991)

些样本用来训练 S_4 层的 47 个子平面。图 11.9 示出了一个网络响应的例子,这个网络是用图 11.5～11.8 中的学习样本训练完的网络。

神经认知机系统的计算机模拟已经有报道(Fukushima 和 Imagmwm 1993,Fukushima,1992)。但是到目前为止,还没有见到能够定量衡量识别精度的报道。而且现有的实验,大多数都是基于不太大的数据集合。

本章中所描述的是基本的神经认知机模型。当识别样本由两个或多个学习样本构成时,网络模型就不能很好地工作。为了能够处理这类组合样本以及能够识别草写体字符,在 1993 年,Fukushima 和 Imagawa 基于基本模型提出了具有选择注意力的神经认知机模型。在识别样本由两个或多个学习样本构成时,网络能分时将它们认出。网络能够有选择地先识别出样本中的一个,然后转换注意力识别余下的样本。只要这些样本数量比较小,这种改进的模型就是非常成功的。但是潦草的手写体中样本的数量是很大的,为了处理这种情况,这种模型必须还得进一步改进。于是加上了一个查找控制器,来限制样本的数量。

Lee 和 Choi 于 1992 年提出一个不是严格基于神经认知机模型的、不太复杂的神经认知机系统。特征图的尺寸大大增加了,于是使得特征更加复杂并且更有意义。他们删掉了神经元层。不变的特征在特征图阶段就已经处理完毕。为了处理不变特征,就必须进行前端的特征提取。在这个实验里,同样也没有定量结果。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

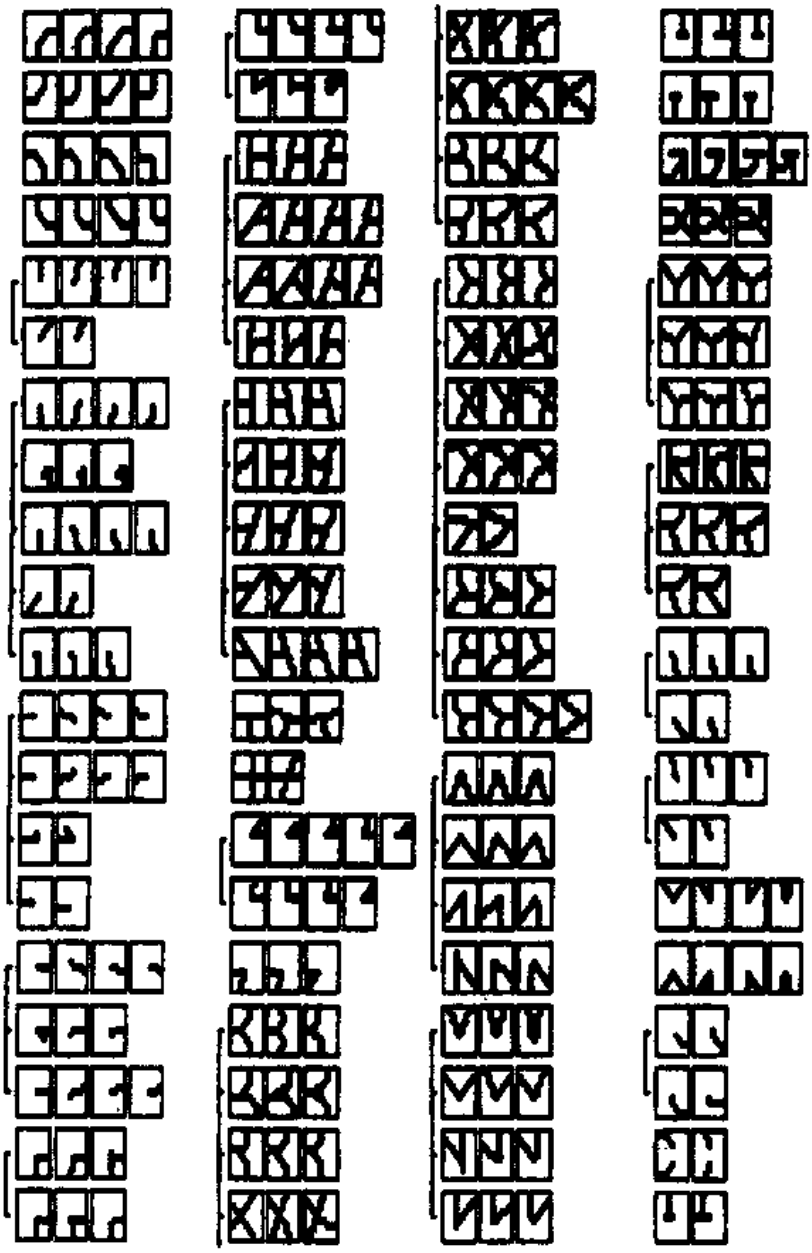


图 11.6 用来训练层的每个子平面所需的 80 个训练样本(摘自 Fukushima, K. 和 Wake, N., IEEE Trans. Neural Network, vol. 2, no. 3, pp. 355 - 365, 1991.)

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

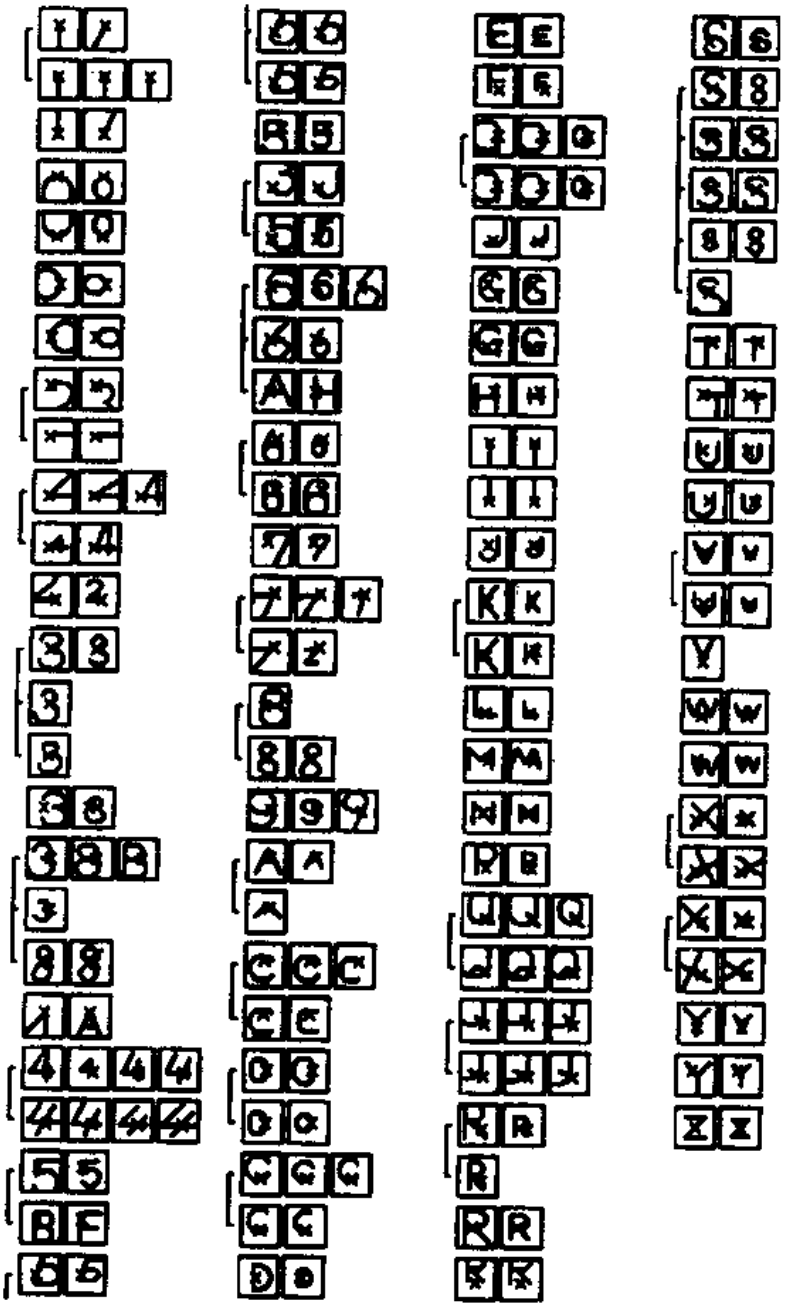


图 11.7 Fukushima 和 Wake 用于训练层每个于平面的 97 个训练样本。图中的是正交线标出了种子神经元的感受野的中心(摘自 Fukushima, K. 和 Wake, N., IEEE Trans. Neural Network, vol.2, no.3, pp.355－365, 1991.)

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

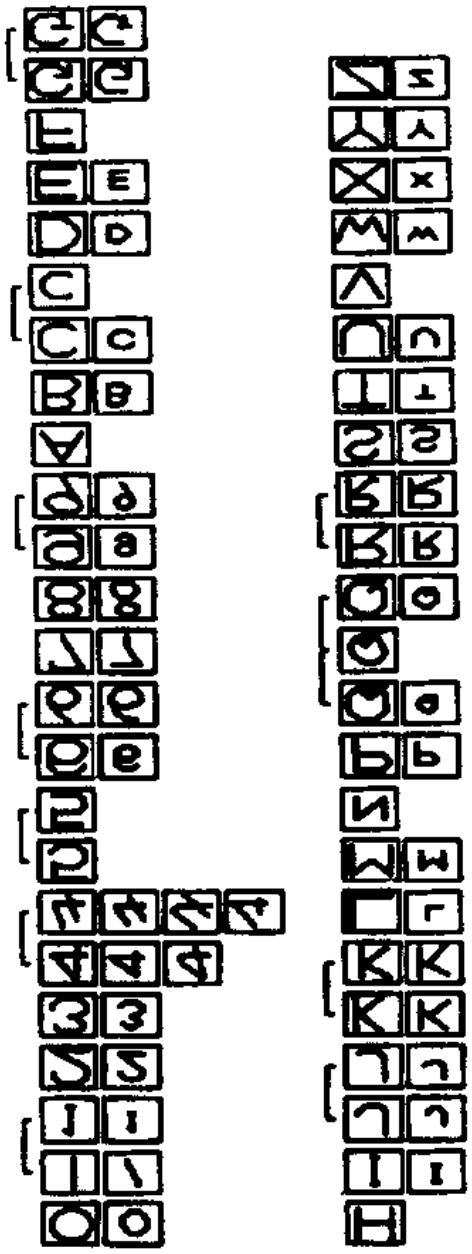


图 11.8 用于训练层的每个子平面的 47 个训练样本(摘自 Fukushima,K. 和 Wake,N., IEEE Trans. Neural Network, vol.2, no.3, no.3, pp.355－365, 1991.)

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

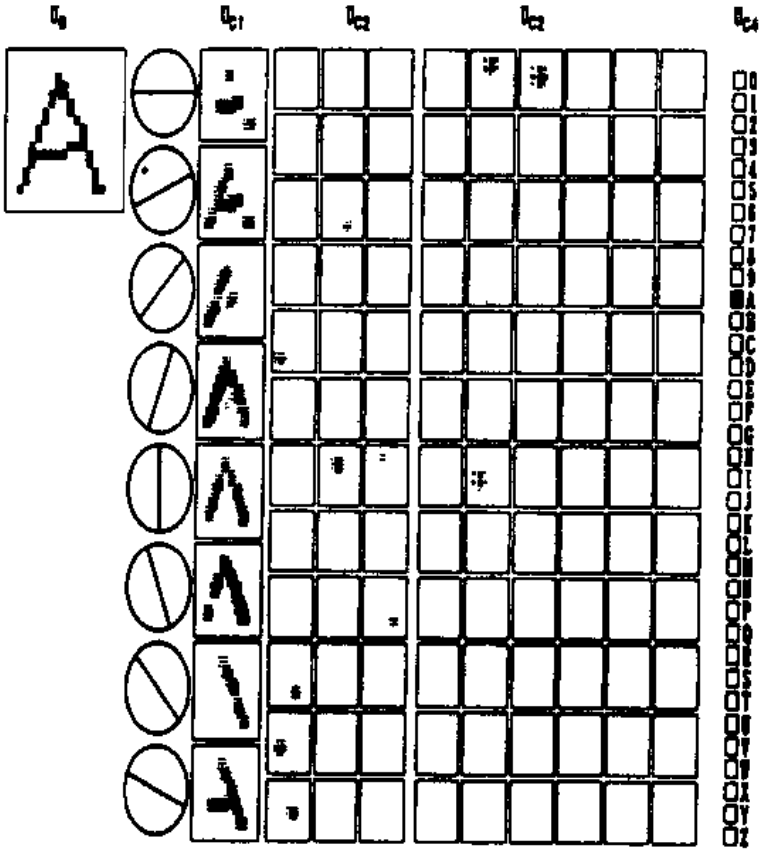


图 11.9 对于已训练完的网络，输入字母"A"时，不同层的 C 神经元的响应
（摘自 Fukushima，K. 和 Wake，N.，IEEE Trans. Neural Network，vol.2，no.3，no.3，pp.355－365，1991.）

参考书与文献

Freeman, J. A. and Skapura, D. M., *Neural Networks: Algorithms, Applications and Programming Techniques*, Addison-Wesley, Reading, MA, 1992.

Fukushima, K., "Neocognition: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.*, vol. 36, pp. 193–202, 1980.

Fukushima, K. and Wake, N., "Handwritten alphanumeric character recognition by neocognition," *IEEE Trans. Neural Networks*, vol. 2, no. 3, pp. 355–365, 1991.

Fukushima, K., "Character recognition with neural networks," *Neurocomputing*, vol. 4, pp. 221–233, 1992.

Fukushima, K. and Imagawa, T., "Recognition and segmentation of connected characters by selective attention," *Neural Networks*, vol. 6, pp. 33–41, 1993.

Hubel, D. H. and Wiesel, T. N., "Receptive fields, binocular interaction and functional architecture in cat's visual cortex," *J. Physiol.*, vol. 160, pp. 106–154, 1962.

Lee, S. and Choi, Y., "Robust recognition of handwritten numerals based on dual cooperative networks," *IEEE IJCNN*, vol. 3, pp. 760–768, 1992.

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@xidian.edu.cn

第十二章　多分类器系统

12.1　综　　述

我们现在已看到有许多不同种类的模式识别分类器，每种都有其自己的许多优点和缺点。因此有理由想到，是否能将许多不同的分类器以某种方式进行组合，以便在总体上能取得比单一分类器更好的效果。图 12.1 给出了应用于模式识别问题的多识别器系统的概念。注意从图中我们可看到：系统中的每个分类器不必是同一类型的。

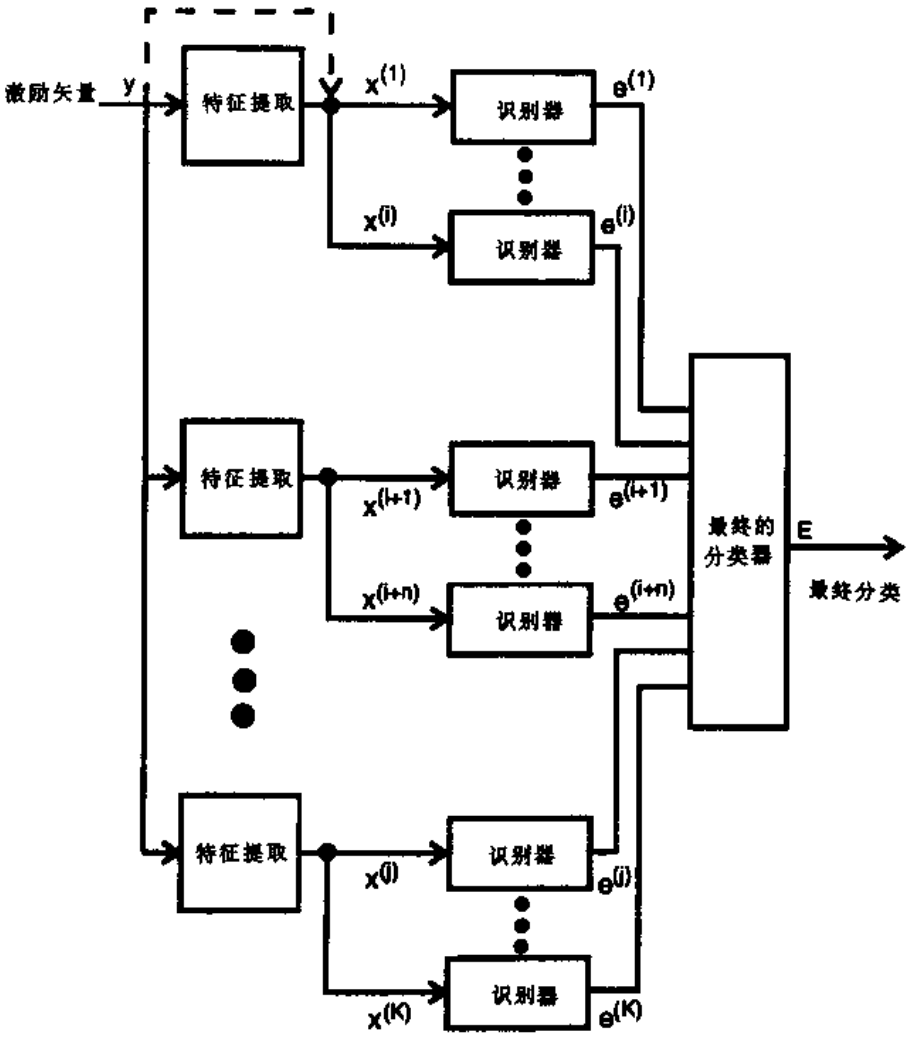


图 12.1　用于模式识别问题的多分类器的应用

尽管我们讨论的主要是以神经网络为基础的系统，但并不是说我们要组合的所有分类器，原则上都必须是基于神经网络的。基于不同方式组合起来的混合系统，具有相当多的优点，并

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Welcome to E-cooperation: luzhenbo@yahoo.com.cn

已应用于商业产品中。在模式识别可能应用的许多不同问题中,统计方法、句法方法和神经网络方法都有自己相应的作用。而且在 1980 年由 Duerr 等发表的文章中我们可看到,基于不同分类器以及不同特征的分类系统经常互有补充。

很明显,为了达到更好的整体识别精度,我们所采用的各个识别器之间存在着很大的差异。一给定的识别器的识别性能,不仅随着识别器的类型和结构而有所变化,也随着下面的许多选择而有所变化:

1. 用以得到原始输入矢量的测量方法的数量与种类;它可能由于许多原因而有所变化,这些原因从识别器自身的性能到已有的系统(由于使用一些特定的测量方法而恰好有许多有用的特点的现有系统)。

2. 用于对输入矢量进行变换(如骨架处理、混淆化以及边缘检测)的预处理的数量与类型。

3. 输入矢量变换的选择,以及用来将输入映射到识别器输入空间的特征的选取(Fouriers 算于、Gabor 变换和 Karhunen－Loeve 变换等)。

4. 训练数据集的大小以及选取。

5. 训练方法和训练参数的选取。

很明显,在许多不同的识别器类型中还会有更多的突出差异。一个识别系统的识别精度,除了依赖于系统中单个识别器的性能,还依赖于多个这样的识别器的组合方式。

尽管存在很多困难,为了进一步说明,将多个分类器组合成一个识别系统的优点,我们考虑两种识别器:识别器 1 和识别器 2。我们将这网种分类器都应用于字符识别问题。我们知道:网种识别器都不是完美的。识别器 1 可能对 a 和 o 会发生混淆,而另一种具有不同特点的识别器 2 可能识别 a 和 o 的效果非常好,但是在识别 8 和 B 问题上效果并不好。当然,识别器 1 对 8 和 B 的识别效果非常好。如果我们将这两种识别器组合起来,以达到只吸收它们对相应问题识别率高的优点,那么整个系统的识别精度就会提高。

正如上面的例子,各种识别器在各自相应的特定问题中,会取得不同程度的成功。因此就提出了这样一个问题:我们在将多个不同系统组合成一个系统时,怎样才能只吸收各分系统的优点而摒弃它们的缺点?在将神经网络识别器和其它的识别器——神经网络型和非神经网络型——组合的过程中,必须注意神经元的兴奋级别和网络选择网络可靠性之间的关系。因而,我们必须考虑设计训练数据集的可靠性。本章中还要讨论摒弃类,识别器可靠性的度量方法以及混淆矩阵。

除了将各个识别器的优点进行组合之外,还有其它的原因也促使我们将几种分类器进行组合。只要是要处理的特征在形式上不同(例如,连续的或离散的),那么就有相应不同的分类器更适合处理这些具有不同特征的数据集。此外,即使特征相对来说是同类的,可能整个特征矢量维数非常高。在这种情况下,如能将这种高维矢量分成几个低维矢量是非常有益的。那么可以用几个分类器来分别处理这些低维矢量,然后用某种方式将几个分类器的结果组合起来,以构成整个识别系统。这种组合是我们非常需要的,因为我们知道高维矢量不仅导致计算复杂,面且会引起网络的运行问题以及精度问题。这时我们可想起第四章所讨论的 OCON 网络,它是用一系列的专家网络来取代单个的 ACON 网络。

把不同的识别系统进行组合,现在还远远不能处理通常的实际问题,并且现有的方法还不能处理给定的问题。因此,我们将讨论多种识别器进行组合的可行方法。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

12.2 多种识别器组合成的系统结构

想要进一步讨论多识别器系统，我们必须如 Xu 等人在 1992 年讨论的那样，将问题以更数学化的方式进行讨论。因此考虑一个具有 M 个互不相交集合的判定空间，$C_i, \forall i \in \Lambda = \{1,2,\cdots,M\}$。每个集合 C_i 代表模式所需分组或分类成的一类或一组。判定空间可写为：

$$P = C_1 \cup C_2 \cup \cdots \cup C_M \tag{12-1}$$

因此，判定空间是所有类中所有可能模式的集合。现在定义一系列相应的整数标记 $\Lambda = \{1,2,\cdots,M\}$。因此，集合 Λ 对于已定义的类提供了所有可能的整数标记。判决空间集合 P 扩展为包括 $\Lambda \cup \{M+1\}$，$M+1$ 表示一个摒弃类。此类是根据已有的标准识别器识别不出的模式。下面用 K 表示在组合系统中识别器的总共数量。系统中的每个识别器都用 e 表示，因此系统中的识别器为 $e_1, e_2, \cdots, e_k$。将模式 x 输入到识别器 e 中，识别器 e 的输出是一整数标号 $j \in \Lambda \cup \{M+1\}$，此标号指明 $\mathbf{x}$ 属于类 C_j（或者可能为摒弃类）。这可表示成 $e(\mathbf{x}) = j$。注意到不管识别器是什么类型，识别器都被看成是一函数黑盒。此函数黑盒输入一模式样本 x，然后输出一分类标号 j。现在我们已建立了大部分所需的符号表示。

尽管由标号 j 指定的分类是任一给定的识别器 e 的最后输出，但是许多现有的分类系统还能在中间阶段提供一其它的有用信息。譬如，贝叶斯分类器能提供 M 个后验概率 $P(i/\mathbf{x})$，$i = 1,\cdots,M$。也就是说，矢量 $\mathbf{x}$ 分类为 C_i 的概率。在这种分类器中，标号就被简单当作这些概率中的最大的一个概率。一旦分类完成，可能就不被保留这些概率；然而，这些摒弃的信息在多分类系统中可能非常有用。不同的识别器所能提供的信息种类是不同的。下面是一种通常用在识别输入矢量 $\mathbf{x} \in P$ 时的信息级别分类法。

1. 抽象级别——识别器对于输入 $\mathbf{x}$ 只提供标号的第一或最好的选择。

2. 排序级别——识别器对于输入 $\mathbf{x}$ 提供标号选择的已排序清单（例如，第一、第二、第三以及第 n 选择）。然而，在排序级别中，没有指明清单中每种选择的优先程度。

3. 测量级别——识别器对代表“输入 $\mathbf{x}$ 是某种标号的概率”的每一标号提供一量度。

我们可能会认为，某些神经网络识别器（譬如，多层感知器网络）工作于测量级别。不幸的是，在网络的识别过程中即 $e(\mathbf{x}) = j$ 时，这种网络输出神经元兴奋级别通常不能代表任何像网络确定性那样可靠的度量标准。判定边界、超空间中的曲面，是用来保证（通过训练）类分离的。也就是说，创建判定边界是为了使误差——采用一个非常有代表性的训练集合——低于预先设定的阈值。将神经元的兴奋级别当作测量级别的困难在于：画出分离超平面的方法不只一种。图 12.2 对于非常简单的分类问题给出几种画法。

图 12.2 中的区域 A 和 B 表示训练集合。线 1、2 和 3 表示满足训练规则的三个分离超平面。这三个超平面都能完美地分离训练数据。但我们也观察到：随着误差表面形成的方式不同，在这些区域外可能存在截然不同的兴奋级别。选择一个大小合适、具有代表性的训练集合并且使用现代的训练方法，那么结果就没有例子那样极端。我们是特意选取这一极端例子，使得网络兴奋级别本身并不表示置信级别。尽管可通过一些实验观察，将神经网络分类器的分类结果进行整理，但现在我们仍有必要将神经网络识别器看作抽象级别。

相反，贝叶斯分类器能提供一衡量输入矢量属于每一类程度的量度。对于单个贝叶斯分类器 e，都可得到概率集合。这就是 $\mathbf{x}$ 属于类 C_i 时，使 $\mathbf{x}$ 发生的概率。

$$\mathrm{P}(\mathbf{x} \in C_i | \mathbf{x}), i = 1,\cdots,M \tag{12-2}$$

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

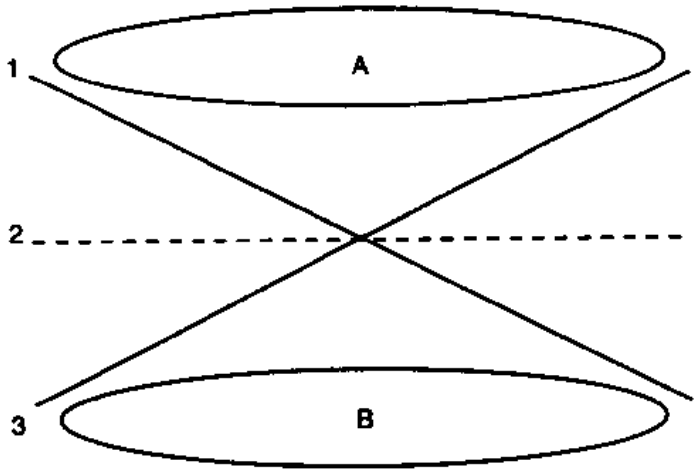


图 12.2 三个超平面，每个都能分离指定的类

现在我们考虑怎样使用这种多分类器系统(注意到这通常并不是问题的解决方法，因为我们仍在讨论一种类型或一个识别器)。系统内的每个识别器 e_k，给出一后验概率如下：

$$P_k(\mathbf{x}\in C_i|\mathbf{x}), i=1,\cdots,M, k=1,\cdots,K \tag{12-3}$$

对应于各个识别器 e_k，相应的判定函数表示如下：

$$e_k(\mathbf{x})=j \text{ 当 } P_k(\mathbf{x}\in C_j)=\max_{i\in\Lambda}P_k(\mathbf{x}\in C_i|\mathbf{x}) \text{成立时} \tag{12-4}$$

各个识别器的判决函数对于我们相对来说并不太重要，因为我们是想建立整个系统的判决函数 E。因此想做的是将各个识别器的概率(见式(12-3))以某种方式组合起来，使得对于整个系统来说有意义。也就是说，对于每一类 C_i，我们想把组合后的识别器后验概率作为一个独立量进行估值。一个最简单的方法是将所有识别器取平均值，见式(12-5)。

$$P_E(\mathbf{x}\in C_i|\mathbf{x})=\frac{1}{K}\sum_{k=1}^{K}P_k(\mathbf{x}\in C_i|\mathbf{x}), i=1,\cdots,M \tag{12-5}$$

P_E 得到后，整个分类器的判决函数 $E(\mathbf{x})$就很容易得到：

$$E(\mathbf{x})=j$$
$$\text{当}$$
$$P_E(\mathbf{x}\in C_j|\mathbf{x})=\max_{i\in\Lambda}P_E(\mathbf{x}\in C_i|\mathbf{x}) \tag{12-6}$$

上面的判决函数，总是选择整个组合系统的后验概率为最大值的那个分类。然而我们也可考虑将错误识别率用摒弃率来代替。当错误分类时，就出现识别误差。摒弃率指的是识别器自身不能处理的模式数量。很明显，使用式(12-6)就不会出现摒弃。在认为宁愿摒弃也不愿错误分类时，将式(12-6)扩展，得新的判决函数如下：

$$E(\mathbf{x})=\begin{cases} j & P_E(\mathbf{x}\in C_j|\mathbf{x})=\max_{i\in\Lambda}P_E(\mathbf{x}\in C_i|\mathbf{x})\geq\alpha \\ M+1 & \text{其它值} \end{cases} \tag{12-7}$$

上式中的 $M+1$ 表示摒弃类，并且在如下范围内选择：

$$0\leq\alpha\leq 1 \tag{12-8}$$

α 值越小就摒弃越少，α 值越大对属于摒弃类的检验就越严格。上面的讨论，描述了怎样组合测量级别(即类3)的分类器。但也应注意到，我们所组合的识别器都是同一类型的。我们可以扩展上面讨论的技术，以便使用其他形式的识别器，只要这种识别器的测量级信息可以得到。如果能找到一种合适的方法对测量级别的量度进行规一化，我们就可以对不同形式的

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

识别器进行组合。下节中将讨论在抽象级别上不同种类的识别器的组合。

12.3 投 票 方 案

在抽象级别,从每个识别器 e_k 只能得到第一或最好选择。因此,自然我们就可使用投票方法组合抽象级别(类型 1)的分类器。假定各个识别器的判决函数 $e_k(\mathbf{x})$有可能不同意,对于组合识别器我们找一个总体判决函数 $E(\mathbf{x})$。为便于讨论,将 $e_k(\mathbf{x})$用另一种形式表示。定义一二值特征函数 $T_k(\mathbf{x})$如下:

$$T_k(\mathbf{x}\in C_i)=\begin{cases}1 & e_k(\mathbf{x})=i,i\in\Lambda\\ 0 & \text{其它值}\end{cases} \tag{12-9}$$

下面的一段 C++程序代码,给出了按照式(12-9)计算特征函数的过程。$e(K)$是 K 个识别器的选择数组。我们也将 $T[M+1]$定义成包含特征函数的全局数组。

```
void characterisfic() {
  int i,j,c;
for(j=0; j<M; j++)
  T[j]=0;
for (i=0; i<K;i++) {
  c=e[j]
  if()
    T[c]=1;
  |
}
```

投票的最保守形式是所有的识别器都必须同意;否则此模式就被摒弃。这可表示为:

$$E(\mathbf{x})=\begin{cases}j & \text{如果} \quad \exists j\in\Lambda,\bigcap_{k=1}^{K}T_k(\mathbf{x}\in C_j)>0\\ M+1 & \text{其它值}\end{cases} \tag{12-10}$$

上式中的符号,表示逻辑"与"运算符。本节中,我们也用它来表示逻辑"或"运算符。

下面的一段 C++程序代码,给出了式(12-10)的投票方案。$e(k)$是 K 个识别器的选择数组。

```
int e[K];
int T[M+1];

int E(){
int c;
c=e[0];
for(i=0;i<K;i++){
  if(e[i]! =c)
    return (M+1);
```

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication E-mail: luzhenbo@yahoo.com.cn


```
  }
return c;
}
```

很明显,所有的识别器都选择同一类是非常严格的条件,并可能导致将能正确分类的模式矢量摒弃掉。然而在错误分类的结果可能是非常严重的情况下,这种方法可能是比较适合的。

如果我们允许系统中的某些识别器可以弃权,那么就可得到上面方案的一种改进方案。在这种方案中,只需那些表决的识别器同意即可。也就是说,那些选择摒弃类的识别器不表决,但所有表决的识别器必须同意。这可由式(12-11)表示:

$$E(\mathbf{x})=\begin{cases} j & \text{如果 } \exists j\in\Lambda, \bigcap_{k=1}^{K}\left\{ T_k(\mathbf{x}\in C_j)\cup\left(1-\bigcup_{q=1}^{M}T_k(\mathbf{x}\in C_q)\right)\right\}>0 \\ M+1 & \text{其它值} \end{cases} \tag{12-11}$$

即仅当一些分类器认为 $\mathbf{x}\in C_j$ 而没有分类器认为 $\mathbf{x}\in C_{ij}$ 时,$E(\mathbf{x})$将 x 分类为 C_j。

下面的一段 C++程序代码,给出了式(12-11)的投票方案。

```
int e[K];
int T[M+1];

int E(){
int c;
int i,j
i=0;flag=0;
while(j<K)&&(! flag)){
  c=e[j];
  if(c! =M+1)
    flag=1;
   else
    j++;
   }
for(i=j+1;i<K;i++){
  if ((e[i]! =c)&&(e[i]! =M+1))
    return (M+1);
  }
return C;
}
```

一更常见并且不太严格的投票方案,可能更有用。在此方案中,并不是完全不允许识别错误。这即是大多数投票规则,表示如下:

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

$$E(\mathbf{x})=\begin{cases} j & \text{如果} \quad T_E(\mathbf{x}\in C_i)=\max_{i\in\Lambda}T_E(\mathbf{x}\in C_i)>\dfrac{K}{2} \\ M+1 & \text{其它值} \end{cases} \tag{12-12}$$

这里：

$$T_E(\mathbf{x}\in C_i)=\sum_{k=1}^{K}T_k(\mathbf{x}\in C_i) \qquad ,i=1,\cdots,M \tag{12-13}$$

把式(12-12)和式(12-13)综合起来，就是说：如果超过一半的识别器认为 $\mathbf{x}\in C_j$，那么模式即被分类为 C_j。大多数投票归纳为：由 $0\leq\alpha\leq1$ 所指定的一部分识别器，必须同意下式：

$$E(\mathbf{x})=\begin{cases} j & \text{如果} \quad T_E(\mathbf{x}\in C_i)=\max_{i\in\Lambda}T_E(\mathbf{x}\in C_i)\geq\alpha\cdot K \\ M+1 & \text{其它值} \end{cases} \tag{12-14}$$

注意到由式(12-14)定义的投票规则，只需与最后被选标号 j 对应的 T_E 超过阈值即可。不考虑其它的标号。在 $\alpha<0.5$ 时，可能有几个标号都是最大值(即对于第一出现同票)。当然也有可能不管怎样选取 α，第二好的选择与收到最多票的标号的值相差不多。这种情况下确定的选择可靠性就值得怀疑，因为有可能存在由第二好选择的这个识别器得到另外的分类结果。解决这类问题的方法，就是需要有一些阈值使得最好的和第二好的选择不同。取 γ_1 和 γ_2 表示附加的特征函数，如下：

$$\gamma_1=\max_{i\in\Lambda}T_E(\mathbf{x}\in C_i) \tag{12-15}$$

$$\gamma_2=\max_{i\in\Lambda-\{j\}}T_E(\mathbf{x}\in C_i) \tag{12-16}$$

上式的 j 表示对应于 γ_1 的标号。

式(12-17)给出了考虑识别器之间竞争情况的投票方法：

$$E(\mathbf{x})=\begin{cases} j & \text{如果} \quad T_E(\mathbf{x}\in C_j)=\gamma_1 \quad \text{并且} \quad (\gamma_1-\gamma_2)>\beta\cdot K \\ M+1 & \text{其它值} \end{cases} \tag{12-17}$$

12.4 混 淆 矩 阵

很明显，12.3 节所讨论的投票方案，并没有考虑每个识别器的特点。每个识别器都以非常民主的方式得到一票。能容纳各识别器差异的一种方式，就是采用混淆矩阵。每个分类器 e_k 的误差都可由混淆矩阵等式描述。

$$C(e_k)=\begin{bmatrix} \eta_{11}^{(k)} & \eta_{12}^{(k)} & \cdots & \eta_{1(M)}^{(k)} & \eta_{1(M+1)}^{(k)} \\ \eta_{21}^{(k)} & \eta_{22}^{(k)} & \cdots & \eta_{2(M)}^{(k)} & \eta_{2(M+1)}^{(k)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \eta_{M1}^{(k)} & \eta_{M2}^{(k)} & \cdots & \eta_{M(M)}^{(k)} & \eta_{M(M+1)}^{(k)} \end{bmatrix} \tag{12-18}$$

上式中，η_{ij} 表示本应属于类 C_i 的样本被识别为 C_j 的数量。从上式可看到，混淆矩阵中总样本数为：

$$N^{(k)}=\sum_{i=1}^{M}\sum_{j=1}^{M+1}\eta_{ij}^{(k)} \tag{12-19}$$

每类 C_i 中的样本总数为：

$$\eta_i^{(k)}=\sum_{j=1}^{M+1}\eta_{ij}^{(k)},i=1,\cdots M \tag{12-20}$$

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

由 e_k 给出的识别器将样本分类为 C_j 的总数量为：

$$\eta_j^{(k)} = \sum_{i=1}^{M} \eta_{ij}^{(k)}, \quad j = 1, \cdots, M+1 \tag{12-21}$$

理想的识别器的混淆矩阵应为对角矩阵，并且所有下标为 $M+1$ 项的值为零。式(12-22)给出此理想识别器的混淆矩阵：

$$C(e_k^{\text{理想}}) = \begin{bmatrix} \eta_{11}^{(k)} & 0 & \cdots & 0 & 0 \\ 0 & \eta_{22}^{(k)} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \eta_{M(M)}^{(k)} & 0 \end{bmatrix} \tag{12-22}$$

建立一个实验集合序列，此序列是模式空间中有代表性的模式，我们从此可得到混淆矩阵。混淆矩阵常应用于单识别器系统，用以指出在哪对网络的拓扑结构和(或)训练集合进行调整。对于组合系统的识别器总体来说，各个构成部分(识别器)的混淆矩阵，能提供此识别器将模式分类于特定类中的置信度。混淆矩阵能使我们了解识别器的许多性能和特点。我们举一个简单例子。假定有五类模式，并且由一给定识别器对于 500 个模式矢量(每类 100 个)实验，序列的行为得到如图 12.3 所示的混淆矩阵。

现在考虑从图 12.3 所示的混淆矩阵中能得到什么信息。得到的信息总结于表 12.1，在表中我们对混淆矩阵分行进行讨论。

注意到，从混淆矩阵中我们可看到：识别器不仅必须要将本应属于类 C_j 的模式分类为 C_j，而且也应避免出现识别错误。基于此点，在设计训练集合时，把正面例子和反面例子都包含进来是非常关键的。

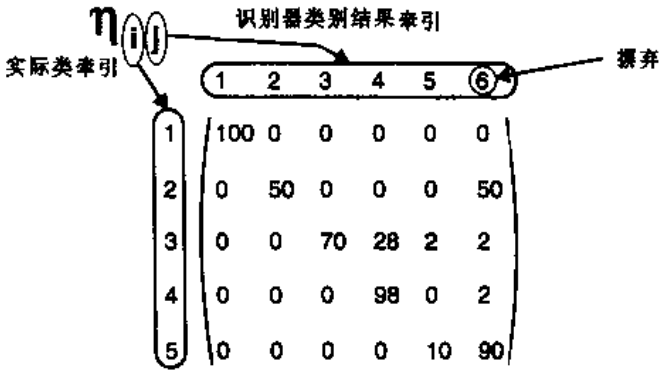


图 12.3 混淆矩阵的例子

对于整个系统，我们可建立一混淆矩阵，以对组合系统的效果进行定量测量。此混淆矩阵表示如下：

$$C(E) = \begin{bmatrix} \eta_{11} & \eta_{12} & \cdots & \eta_{1(M)} & \eta_{1(M+1)} \\ \eta_{21} & \eta_{22} & \cdots & \eta_{2(M)} & \eta_{2(M+1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \eta_{M1} & \eta_{M2} & \cdots & \eta_{M(M)} & \eta_{M(M)} \end{bmatrix} \tag{12-23}$$

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

表 12.1

$x\in C_j$	分　析
1)	识别器总是能识别一矢量 $x\in C_j$；没有识别错误以及摒弃；因此此情况是理想的；在 $x\in C_j$ 以及 $e_k(x)=1$（即列 $\eta_{j1}=0, \forall j\neq 1$）时，$C_1$ 的分类总是可靠的（$e_k(x)=1\rightarrow x\in C_1$）。
2)	在 $x\in C_j, j\neq 2$ 以及 $e_k(x)=6$ 时，只有 50% 的模式矢量 $x\in C_2$ 被正确识别；也就是说，类 C_2 中有一半模式被识别器识别成摒弃类；所以此识别器常常丢失类 C_2 当中的模式；注意到一旦选择 C_2，那么结果就完全可靠，因为 $\eta_{2j}=0, \forall j\neq 2$；我们还应注意到：因为 $e_k(x\in C_2)$ 永远不会和其它类混淆，当 $x\notin C_2$ 时 $x\in C_2$ 的模式不降低准确度成可靠性。
3)	对于属于类 3 的模式，$x\in C_3$；此时识别器能正确地对 70% 的模式分类；尽管看起来似乎比上面的 50% 效果要好，实际上它也有困难。在这种情况下，如果 $e_k(x\in C_3)\neq C_3$，那么并不是将其归为摒弃类；相反，它实际上是识别错误（常常分类为 C_4）；识别为类 3 的结果是可靠的（也就说如果 $e_k(x)=3$ 时，我们应该相信 $x\in C_3$）；然而我们看到这将影响 $x\in C_4$ 的分类效果。
4)	此时像 $x\in C_4$ 这样的模式矢量的正确分类率为 98%；如未正确分类，那么就如我们所希望的那样将此模式摒弃；这表示已非常接近理想；然而，当此识别器对一模式分类结果为 $e_k(x)=4$ 时，我们一定对此结果表示相当的怀疑；当 $x\in C_3$ 时由于错误分类（识别错误误差），导致有相当大的可能性当 $e_k(x)=4$ 时，实际上 $x\in C_3$。
5)	识别器对属于 C 类的模式矢量的识别效果非常差；对于此识别器的性能所能说的就是：对矢量 $x\in C_5$ 识别时，它总是能将不能正确分类的矢量摒弃并且不影响对其它类的分类结果；即使这样，如此结果明显表明此识别器对于 C 类中矢量的分类性能需要引起注意。

式中：

$$\eta_{ij}=\sum_{k=1}^{K}\eta_{ij}^{(k)} \tag{12-24}$$

另外，从 M 个识别器所提供的候选类集合中，选出类 j 的这个问题本身，可看作一分类问题。由于每个识别器的内在特点，对于某些模式，这些识别器之间的关系可能不是很明显。所以我们设想有可能将混淆矩阵的信息通过简单方法提供给神经网络（FFMLP），也就是将与建立混淆矩阵所用的相同实验序列和含有正确分类的训练样本集合一同输入到网络。输入含有从每个识别器得到的分类结果（如果可以的话也许加上额外的测量级别信息）。我们有理由设想，这样的网络可以找出各识别器特性之间的不明显联系。所以，可用此网络代替先前讨论的投票方案。实际上，它就像 OCON 上下文中的末端一样，就是一个后处理的最后分类器。另外，我们可使网络将摒弃类和由输入训练样本建立的敏感性合并。

12.5 可　靠　性

在衡量不管是单个识别器还是组合识别系统时，分类器的另外一有用的量度是可靠性。成功识别（模式被正确分类）的百分比、错误识别（识别器错误分类）的百分比和摒弃（识别器不管怎样都不能正确对模式进行分类）的百分比之和，一定为 100% 。当识别器对一模式进行分类（向对于摒弃来说）时，分类实际上是正确的可靠程度，我们用可靠性来表示。可靠性由下式定义：

$$\text{可靠性}=\frac{\text{识别率}}{100\%-\text{摒弃率}} \tag{12-25}$$

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

从式中可看出:可靠性这个量度是奖励正确分类,而同时又惩罚错误分类。摒弃是有益的,因为它不降低可靠性。注意到:即使识别器对于输入模式只能正确地识别出很少的模式,只要没有错误分类,那么可靠性就为 1.0(即 100% 的可靠)。我们基于测试集来计算可靠性。

可靠性可应用在单个识别器中,错误分类是非常不利的情况下。它也可用来计算每个候选识别器能被整个识别系统所采用的可能性。我们在先前讨论混淆矩阵时以看到:具有较低可靠性的识别器,对整个识别系统的正确分类有很大影响。在设置一定摒弃标准以达到 100% 可靠性的场合下,经常有其它测量方法的报道。

12.6 一些经验方法

先前在 12.2 节中我们已知道,FFMLP 网络的兴奋级别不适合在测量级别上进行分析。尽管如此,在将投票方案应用于这些网络时,一些关于兴奋级别的实验发现,对于我们的选择大有启发。令具有最高兴奋级别的神经元,即最好神经元为 γ_1。令具有第二兴奋级别的神经元,即第二好神经元为 γ_2。从实验中发现,网络的确定性是 $\gamma_1-\gamma_2$ 的差值的函数,同时也是 γ_1 的函数。Yizhak 和 Chevalier 于 1991 年也发现了此结果,他们使用三条规则来建立一摒弃类:

1. 最好的神经元输出超过一阈值。
2. 次好的神经元输出低于一阈值。
3. 最好的神经元与次好的神经元之差低于一阈值。

可以证明,含有上述思想的连续函数,对于在一投票方案中衡量网络投票是有益的。需要强调的是,这种函数仍不能像前面类型 1 那样代表所处理的概率。但是,这种函数可用于加权的投票方案,在此方案中一个选票的权值通过 $f(\gamma_1,\gamma_2)$ 来加权。一更为保守的方法是将其应用于阈值以在识别器根本不投票时建立摒弃类。下面是在考虑了最大兴奋级别值以及其与第二好神经元差值的情况下的一种 $f(\gamma_1,\gamma_2)$ 函数:

$$f(\gamma_1,\gamma_2)=\gamma_1 e^{\frac{\gamma_2^2}{2\gamma_1^2(\gamma_1-\gamma_2)}} \tag{12-26}$$

由式(12-26)产生的函数族画于图 12.4。将式(12-13)中的 $T_k(\mathbf{x})$ 用 $f(\gamma_1,\gamma_2)$ 来代替,得到改进的投票方案。

神经网络识别器的组合不只局限于投票方案。早在第四章,就至少讨论了两种这样的方法。我们看到,MAXNET 网络可用来对各识别结果进行选择。与此类似,前端网络可用来从一整体系统的一些专家网络中选出一个网络。不容置疑,我们只要对组合识别系统进一步研究,那么随着此领域的发展将会有更多的发现。

More documents and datum download website
Lu Zhenbo's Blog: blog.sina.com.cn/luzhenbo2
Communication & Cooperation: luzhenbo@yahoo.com.cn

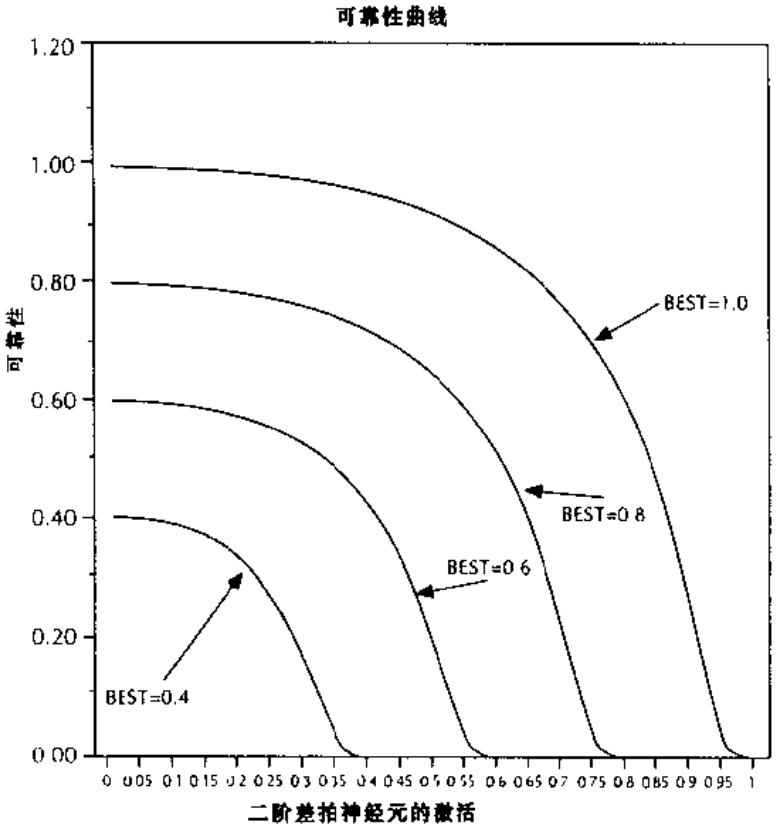


图 12.4 可靠性曲线

参考书与文献

Bebbis, G. N., Georgiopoulis, M., Papadourakis, G. M., and Heilman, G. L., "Increasing classification accuracy using multiple neural schemes," *Proc. SPIE Appl. Neural Networks III*, vol. 1709, pp. 221–231, 1992.

Duerr, B., Haettich, W., Tropf, H., and Winkler, G., "A combination of statistical and syntactical pattern recognition applied to classification of unconstrained handwritten numerals," *Pattern Recognition*, vol. 12, no. 3, pp. 189–199, 1980.

Perone, M. P. and Cooper, L. N., "When Networks Disagree: Ensemble Methods for Hybrid Neural Networks," U.S. Dept. of Commerce Report AF-S260 045, 1992.

Yizhak, I. and Chevalier, R. C., "Handwritten digit recognition by supervised Kohonen-like learning algorithm," *1991 IEEE Joint Conf. Neural Networks IJCNN*, pp. 2576–2581, 1991.

Xu, I., Krzyzak, A., and Suen, C. Y., "Methods of combining multiple classifiers and their applications to handwriting recognition," *IEEE Trans. Syst. Man, Cybern.*, vol. 22, no. 3, 1992.