北京师范大学珠海分校

本科生毕业论文

论文题目：一个基于BP网络的字符识别器的设计与实现

学　　院	信息技术学院
专　　业	软件工程
学　　号	0901030113
学 生 姓 名	谢思路
指导教师姓名	杨戈
指导教师职称	副教授
指导教师单位	信息技术学院

2013 年 3 月 16 日

北京师范大学珠海分校学位论文写作声明和使用授权说明

学位论文写作声明

本人郑重声明： 所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品或成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律结果由本人承担。

论文作者签名： 日期： 年 月 日

学位论文使用授权说明

本人完全了解北京师范大学珠海分校关于收集、保存、使用学位论文的规定，即：按照学校要求提交学位论文的印刷本和电子版本；学校有权保存学位论文的印刷本和电子版，并提供目录检索与阅览服务；学校可以采用影印、缩印、数字化或其它复制手段保存论文；在不以赢利为目的的前提下，学校可以将学位论文编入有关数据库，提供网上服务。（保密论文在解密后遵守此规定）

论文作者签名： 导师签名：

日期： 年 月 日

一个基于 BP 网络的字符识别器的设计与实现

摘 要

本论文提出了一种利用 BP 网络识别印刷字符的方法，通过实验验证了其有效性和正确性。本论文第一章对人工神经元网络进行了概述，第二章对 BP 网络的理论知识进行了详细的论述。BP 网络是一种神经元网络，本论文以 BP 网络为重点研究对象，在第三章中基于 BP 网络设计了一个字符识别器，并在第四章中用 C 语言实现了字符识别器。第五章分析了传统 BP 算法的缺陷，提出并实现了针对传统 BP 算法的改进方法。最后在第六章中对人工神经元网络和 BP 网络进行了展望和总结。

关键词：人工神经元网络；BP 网；图像识别；C 语言

THE DESIGN AND IMPLEMENTATION OF A CHARACTER RECOGNIZER BASED ON BP NETWORK

ABSTRACT

This paper puts forward a way to use BP network to recognize printed characters, and its validity and the correctness is verified by experiment. In the first chapter of this paper, the theory of Artificial Neural Networks is discussed and in the second chapter, the theory of BP network is introduced in detail. BP network, which is a neural network, is the key of research of this paper in which a character recognizer based on BP network is designed and implemented respectively in the third chapter and the forth chapter with C language. Besides, the defects of traditional BP algorithm are analyzed and the improvement methods are put forward in the fifth chapter. Finally, in the sixth chapter, summarize and prospect of Artificial Neural Networks and BP network is expounded.

Key words: Artificial Neural Networks; Back-Propagation network; image recognition; C language

目　录

1. 绪论

1.1 课题的研究背景及意义

人工神经元网络（ANN Artificial Neural Networks）的理论、实现方式以及算法是人工智能（AI　Artificial Intelligence）研究的重要课题之一，也是智能控制的重要分支领域[1]。自从人们认识到人脑计算与传统的数字计算机相比是完全不同的方式开始，关于人工神经元网络（一般称之为“神经网络”（neural network））的研究工作就开始了[2]。人脑是一个高度复杂的、非线性的和并行的计算机，能够组织它的组成成分，即神经元，以比今天已经存在的最快的计算机还要快许多倍的速度进行特定的计算。而神经元网络是由简单处理单元构成的大规模并行分布式处理器，天然地具有存储经验知识和使之可用的特性。我们以人工的方法模拟人脑的功能，有助于加深对思维及智能的认识。自 40 年代以来，人们在研究人脑机理的基础上，广泛开展模仿脑模型的人工神经网络理论的研究，由于人工神经元网络具有与人脑相似的高度并行性、良好的容错性和联想记忆功能、自适应和自学习能力等特点，尤其是以 BP 网络（Back-Propagation 误差逆传播）为代表的神经网络具有良好的自学习能力、强大的分类能力、容错能力，可以实现输入到输出的非线性映射。

人工神经元网络的崛起，对认知和智力的本质的基础研究乃至计算机产业都产生了空前的刺激和极大的推动作用，它是一门活跃的边缘性交叉学科，研究它的理论与技术，具有重要的现实意义。近年来，人工神经元网络的研究已经取得了很大的进展，被广泛地用于通信、计算机、电力、医学、经济、冶金、测绘等领域中，并且解决了一系列的实际问题，如模式识别、信号处理、专家系统、机器人控制等，给人类各领域的生活带来了极大的便利。

1.2 人工神经元网络概述

1.2.1 神经元网络中的单元

（1）生物神经元。

人工神经网络的研究在一定程度上受到了生物学的启发，因为生物的学习系统是由相互连接的神经元组成的异常复杂的网络，而人工神经元网络与此大体相似[3]。人工神经元网络是对生理学上真实人脑生物神经元网络的结构、功能，以及若干基本特性的某种理论抽象、简化和模拟，它实际上是一种复杂的信息处理系统，是由大量神经元通过极其丰富和完善的联接而构成的自适应非线性动态系统。

人类的智能来源于大脑，而神经元是大脑中的细胞，它的主要功能是收集、处理和分发电信号。图 1 显示了一个典型神经元的示意图。它是由细胞体，树突、轴突和突触四部分构成，其中树突和轴突负责细胞体的兴奋和抑制信息的输入和输出，突触是调节神经元之间相互作用的基本结构和功能单位。它的运行过程如下：前突触过程释放发送器物质，扩散到神经元之间的突触连接，然后作用于后突触过程。这样突触就完成了突触前端的电信号向化学信号的转换，然后转换回突触后端电信号。

---

[1] 孙佰清 张长胜等.提高 BP 神经网络训练速度的研究[J].哈尔滨工业大学学报 2001,5: 32-36.

[2]（加）Simon Haykin. 神经网络与机器学习[M] 北京：机械工业出版社 2011.

[3]（美）Tom M.Mitchell.机器学习[M] 北京：机械工业出版社 2003.

人的神经系统可看作三个阶段系统，如图 2 所示。系统的中央是人脑，由神经网络表示，它持续地接收信息，感知它并作出适当的决定。图中有两组箭头，从左到右的箭头表示携带信息的信号通过系统向前传输，从右到左的箭头表示系统的反馈。感受器把来自人体或外界环境的刺激转换成电冲击，对神经网络（大脑）传送信息。神经网络的效应器会将神经网络产生的电冲击转换为可识别的响应从而作为系统的输出。

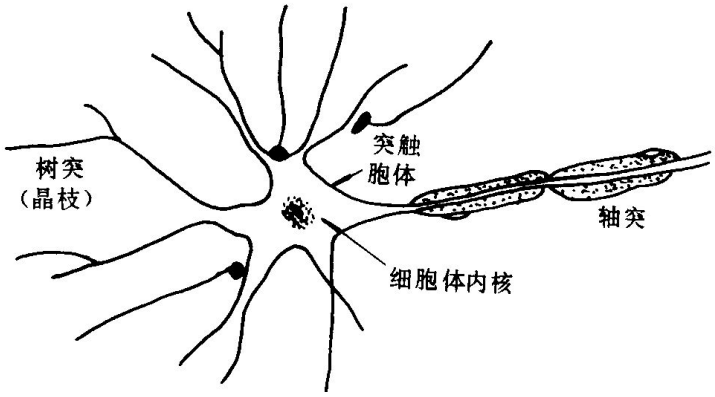

图 1 神经元

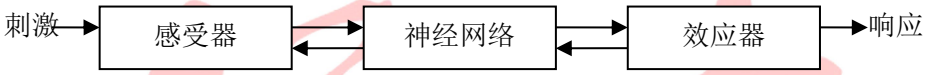

图 2 神经系统的感应过程

（2）人工神经元。

大脑的信息处理能力被认为主要是从这种神经元构成的网络涌现出来的，因此一些早期的人工智能工作致力于创造人工神经元网络，这个领域包括连接主义、并行分布处理，以及神经计算。人工神经元网络在两个方面与大脑相似，第一是神经元网络是通过学习过程从外界环境中获取知识的；第二是互联神经元的连接强度，即突触权值，用于存储获取的知识。神经元网络是神经元构成的，神经元是神经网络操作的基本信息处理单位。它是人工神经元网络的基础。图 3 显示了一个简单的神经元数学模型。当输入的线性组合超过一定的阈值时，它就会激发。

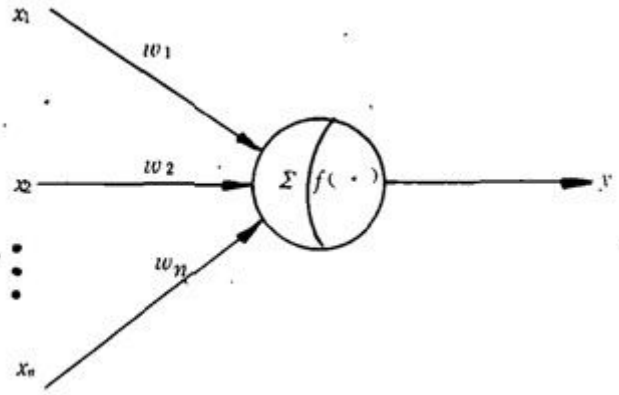

图 3 人工神经元数学模型[1]

[1] 张可,张高燕,吴苏,范海菊.基于 BP 神经网络的字符识别系统.计算机与现代化[J].2009,1:63-72.

神经元之间通过有向边连接在一起。从单元 j 到单元 i 的边的作用是把激励 a_j 从 j 传播到 i。每条边还有一个数值的权值 $W_{j,i}$ 与之相关联，它决定了连接的强度和符号。每个单元 i 首先对它的输入计算一个加权和：$net = \sum_{j=0}^{n} Wj,ia_j$ 。然后把激发函数 s 应用于这个和，产生输出：$a_i = s(net) = s(\sum_{j=0}^{n} Wj,ia_j)$。其中 $W_{0,j}$ 是偏离权，是实际阈值。激发函数需要起到两个作用：第一，当提供了正确的输入时，单元就被激活，当提供了错误的输入时，单元是非激活的；第二，激发需要是非线性的，否则整个神经元网络就退化成了一个简单的线性函数。其中激发函数 s 有如图 4 所示的几种类型[1]。当真实输入的加权和超过偏离权 $W_{0,j}$，即实际阈值时，神经元被激发。BP 神经元网络一般选择 S 型函数（见图 4 中的 c，也称逻辑函数）作为激发函数，其公式如下：

$$f(x) = \frac{1}{1+\exp(x)}$$

S 型函数的优点在于它是可微分的[2]。

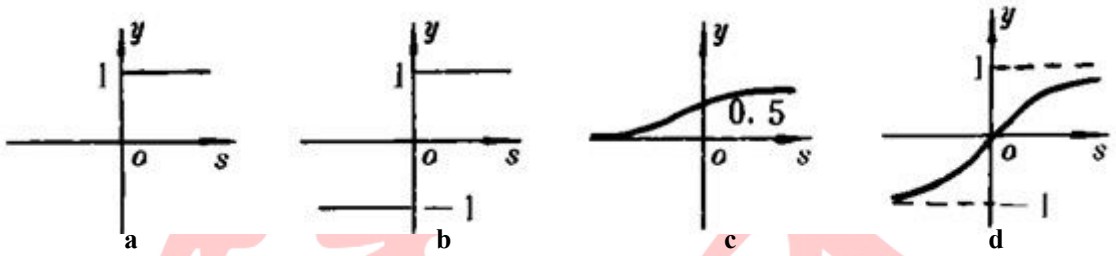

图 4 激发函数[3]

1.2.2 人工神经元网络的分类

人工神经元网络可以按照如下方式分类[4]：

（1）按照网络的结构分类：前向网络和反馈网络。

（2）按照学习方式分类：有教师学习和无教师学习网络。

（3）按照网络性能区分：连续型和离散型网络，随机型和确定型网络。

（4）按照突触性质区分：一阶线性关联网络和高阶非线性关联网络。

（5）按照对生物神经系统的层次模拟分类：神经元层次模型，组合式模型，网络层次模型，神经系统层次模型和智能模型。神经网络有分层网络，层内连接的分层网络，反馈连接的分层网络，互联网络等互连结构。

在几十种神经元网络模型中，人们用的较多的是 Hopfield 网络、BP 网络、Kohonen 网络和 AR 双自适应共振理论网络。Hopfield 网络是最典型的反馈网络模型，它是目前人们研究最多的模型之一。Hopfield 网络是由相同的神经元构成的单层，并且不具学习功能的自联想网络，它需要对称连接。这个网络可以完成制约优化和联想记忆等功能。

---

[1] 胡瑞敏 徐正全 姚天任 李德仁.人工神经元网络的智能神经元模型.电子学报[J].1996,4(24): 86-90.

[2] (美)Stuart Russell,Peter Norvig.人工智能-一种现代方法（第二版）[M].北京：人民邮电出版社 2004.

[3] 胡瑞敏 徐正全 姚天任 李德仁.人工神经元网络的智能神经元模型.电子学报[J].1996,4(24): 86-90.

[4]李佳斌. 人工神经网络系统模型原理及其在计算机中的应用研究与进展[J]. 电脑知识与技术 2011,2(7): 411-414.

BP 网络是误差逆传播(Back Propagation) 网络。 它是一种多层前向网络，采用最小均方差学习方式。 这是一种最广泛应用的网络。 它可用于语言综合、识别和自适应控制等用途，BP 网络需有教师训练。本论文论述的重点是 BP 网络。Kohonen 网络是典型的自组织神经网络，这种网络也称为自组织特征映射网络 SOM。它的输入层是单层单维神经元；而输出层是二维的神经元。在输出层中神经元之间有近扬远抑的反馈特性，从而使 Kohonen 网络可以作为模式特征的检测器。ART 网络也是一种自组织网络模型。这是一种无教师学习网络。 它能够较好地协调适应性，稳定性和复杂性的要求。 在 ART 网络中，通常需要两个功能互补的子系统相互作用。这两个子系统分别为注意子系统和取向子系统。ART 网络主要用于模式识别，它不足之处是在于对转换、失真和规模变化较敏感。

1.2.3 人工神经元网络

由 1.2.2 节可知，按照网络的结构分类，人工神经元网络可分为两类主要结构：前馈网络（见图 5(a)）和循环网络（见图 5(b)）。前馈网络表示了当前输入的一个函数，因此除了权值自身之外，网络没有其它的内部状态。而循环网络将其输出反馈回自己的输入。这意味着该网络的激励层构成一个动力学系统，它可能达到一个稳定状态，也可能发生振荡，甚至进入一个混沌状态。人工神经元网络对于一个给定输入的响应取决于它的初始状态，而初始状态取决于先前的输入。因此循环网络能够支持短时的记忆。

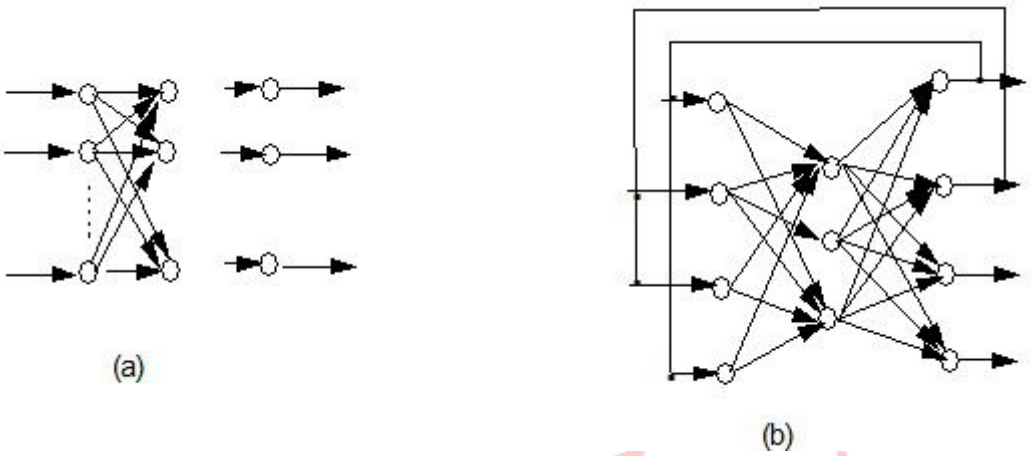

图 5 前馈网络(a)和循环网络(b)

前馈网络分为单层前馈神经元网络（即感知机）和多层前馈神经元网络。在分层网络中，神经元以层的形式组织，如果网络的所有输入直接连接到输出，就称其为单层神经元网络，或称感知机网络。单层指的是计算节点（神经元）输出层。多层前馈神经元网络（见图 6）是指在前馈神经元网络的第二种网络中有一层或多层隐藏层，相应的计算节点称为隐藏神经元或隐藏单元。隐藏是指神经网络的这一部分无论从网络的输入端或者输出端都不能直接看到，隐藏神经元的功能是以某种有用的方式介入外部输入和网络输出之中。循环网络和前馈网络的区别在于它至少有一个反馈环。循环网络可以由单层神经元组成，单层网络的每一个神经元的输出都反馈到所有其他神经元的输入中。

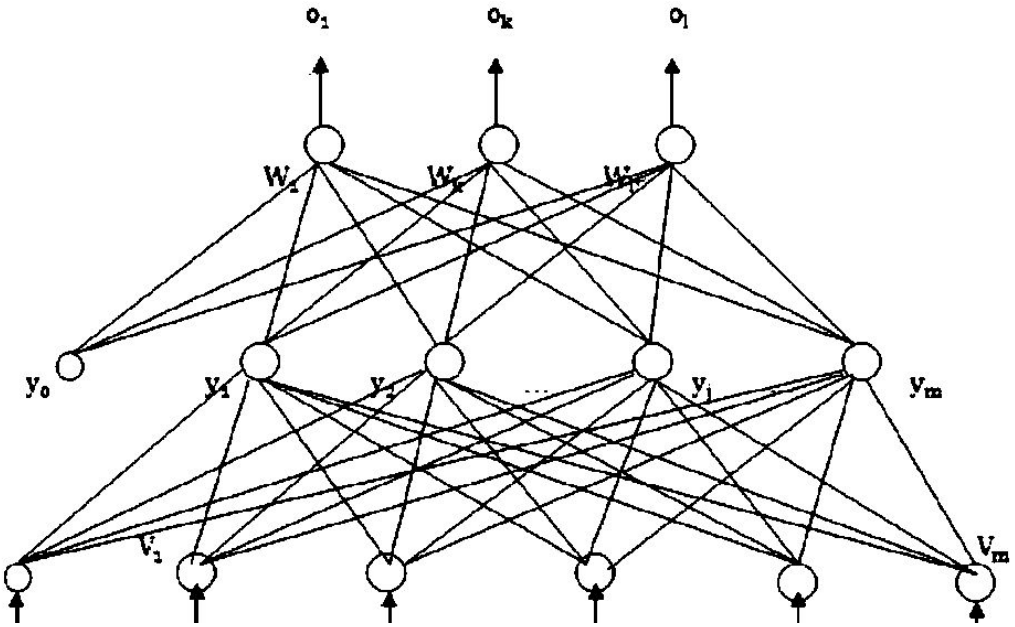

图 6 多层前馈神经元网络[1]

按照神经网络的功能来对其学习过程进行分类可将其分为有教师学习和无教师学习。而有教师学习又可以分为无监督学习和强化学习两个子类。这些应用于神经网络的不同形式是和人类学习的形式相似的。

有教师学习也称为监督学习。图 7 是这种学习方式的框图[2]。教师具有对周围环境的知识，这些知识被表达为一系列的输入-输出样本，而神经网络对环境一无所知。教师可以根据自身掌握的一些知识为神经网络提供对训练向量的期望响应。期望响应一般都代表神经网络完成的最优动作。神经网络的参数可以在训练向量和误差信号的综合影响下进行调整。误差信号可以定义为神经网络的实际响应与预期响应之差。这种调整是逐步而反复的进行着的，其最终目的是让神经网络模拟教师。利用这种方式，教师所掌握的关于环境的知识就可以通过训练过程最大限度地传授给神经网络。当条件成熟的时候，就可以将教师排除在外，让神经网络完全自主地应对环境。这是误差逆传播算法的基础。

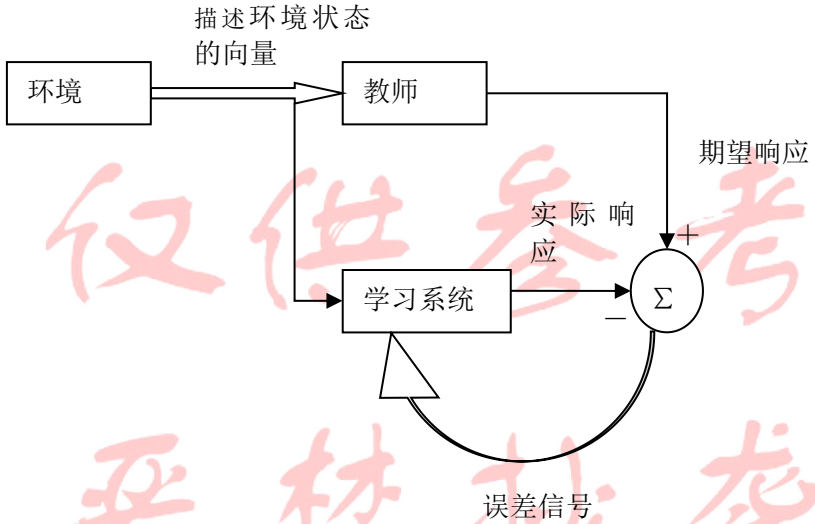

图 7 有教师学习方框图[3]

[1] 李相文，孙文慧，杨波.基于 BP 神经网络的图书编号识别.信息通信[J].2012,4(120):94-95
[2] (加）Simon Haykin. 神经网络与机器学习[M] 北京：机械工业出版社 2011.
[3] (加）Simon Haykin. 神经网络与机器学习[M] 北京：机械工业出版社 2011.

在监督学习中，学习过程是在教师的监督下进行的，而在无教师学习就是没有教师监视学习的过程，没有任何带标号的样例可以供神经网络学习。无教师学习分为两个子类：强化学习和无监督学习。在强化学习中，输入输出映射的学习是通过与环境的不断交互完成的，目的是使一个标量性能指标达到最小。图 8 是强化学习系统的方框图。强化学习系统建立在一个评价的基础上，评价将从周围环境中接收到的原始信号转换成一种高质量的强化信号，这两种信号都是标量输入。该系统的目的是为了适应延迟强化情况下的学习，即意味着系统观察从环境接收的一个时序刺激，它们最终产生启发式的强化信号。如图 9 所示，在无监督学习系统中，没有外部的教师或评价来监督学习的过程，并且必须提供任务独立度量（task-independent measure）来度量神经元网络的表达质量，让神经元网络学习该度量而且根据这个度量来最优化网络自由参数。对一个特定的任务独立度量而言，一旦神经网络能够和输入数据的统计规律相一致，那么网络将会发展其形成输入数据编码特征的内部表示的能力，从而自动创造新的类别。

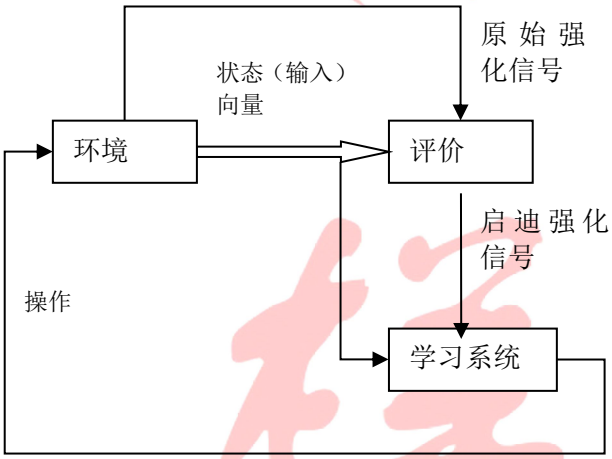

图 8 强化学习方框图

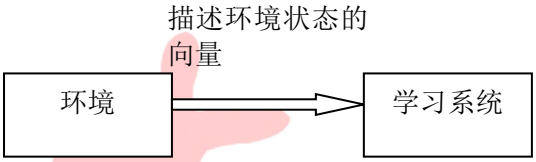

图 9 无监督学习方框图

1.2.3 人工神经元网络的优点与应用

人工神经元网络的计算能力体现在以下两点：第一，神经元网络的大规模并行分布式结构；第二：神经元网络的学习能力以及由此而来的泛化能力。

人工神经元网络具有以下的性质和能力[1]：

（1） 非线性：人工神经元可以是线性或者是非线性的。由非线性神经元互相连接而成的神经网络自身是非线性的，并且非线性是分布于整个网络中的。非线性是一个非常重要的特征，特别是当输入信号的内部物理机制是天生非线性的时候。

（2） 输入输出映射：即有教师学习。它使用带标号的训练样例对神经网络的突触权值进行修改。每个样例由一个唯一的输入信号和相应的期望响应组成。从一个训练集中随机选取一个样例提供给网络，网络就调整它的突触权值，以最小化期望响应和由输入信号以适当的统计准则产生的网络实际响应之间的差别。使用训练集中的很多样例来重复训练神经网络，直到网络达到对突触权值没有显著修正的稳定状态为止。先前已经使用过的训练样例可能还要在训练期间以不同的顺序重复使用。因此神经元网络是通过建立输入输出的映射来从样例中学习的。

（3） 自适应性：神经元网络具有调整自身突触权值以适应外界环境变化固有能力。一个在特定运行环境下接受训练的神经元网络，在环境条件变化不大的时候可以很

[1]（加）Simon Haykin. 神经网络与机器学习[M] 北京：机械工业出版社 2011.

容易地进行重新训练。而且当它在一个不稳定环境中运行时，可以设计神经网络使得其突触权值随时间实时变化。用于模式分类、信号处理和控制的神经网络与它的自适应能力相耦合，就可以变成能进行自适应模式分类、自适应信号处理和自适应控制的有效工具。在保证系统保持稳定时，一个系统的自适应性越好，它就被要求在一个不稳定的环境下运行时其性能越具有鲁棒性（robust）。

（4） 证据响应：在模式分类问题中，神经网络可以设计成不仅提供选择哪一个特定模式的信息，还提供关于决策的置信度信息。后者可以用来拒判那些可能出现的过于模糊的模式，从而进一步改善网络的分类性能。

（5） 上下文信息：神经元网络的特定结构和激发状态代表知识。网络中的每一个神经元都受网络中所有其它神经元全局活动的潜在影响。因此神经元网络能够处理上下文信息。

（6） 容错性：一个以硬件形式实现的神经网络具有天生的容错性。

（7） 分析和设计的一致性：神经元网络作为信息处理器具有通用性，因为涉及神经网络应用的所有领域都使用相同的记号。

（8） 神经生物类比：神经元网络的设计是由与人脑的类比引发的，人脑是一个容错的并行处理的实例，这种处理不仅在物理上是可实现的，而且还是快速的、高效的。神经生物学家将神经元网络看作是一个解释神经生物现象的研究工具。工程师对神经生物学的关注在于将其作为解决复杂问题的新思路，这些问题比基于常规的硬件线路设计技术所能解决的问题更复杂。

（9） 硬件实现：神经网络不仅可以通过软件而且可以通过硬件实现并行处理。近年来，一些超大规模的集成电路实现硬件已经问世，使神经网络快速、大规模处理能力的实现成为可能。

由于人工神经元网络所具有的以上特点，所以它在以下的一些主要领域中得到了应用：

（1） 模式识别与图像处理印刷体和手写体字符识别、语音识别、指纹、人脸识别、RNA 与 DNA 序列分析、癌细胞识别、目标检测与识别、心电图、脑电图分类、油气藏检测、加速器故障检测、电机故障检测、图像压缩复原。

（2） 控制及优化化工过程控制、机械手运动控制、运载体轨迹控制以及电弧炉控制。

（3） 金融预测与管理股票市场预测、有价证券管理、借贷风险分析以及信用卡欺骗检测。

（4） 通信自适应均衡、回声抵消、路由选择、导航以及多媒体处理系统。

（5） 其他如知识发现和数据挖掘、气象与地球科学等。

1.3 本论文的研究任务

本论文主要介绍了人工神经元网络以及 BP 网络，基于 BP 网络设计了一个简单的字符识别器，用 C 语言对其进行了实现。

2. BP 网络

BP 网络（Back Propagation NN）即进行误差逆校正的多层前馈网络，它是利用实际输出与期望输出之差对网络的各层连接权由后向前逐层进行校正的一种网络模型，它适用于任意多层的网络。图 10 所示是一个三层 BP 网络。

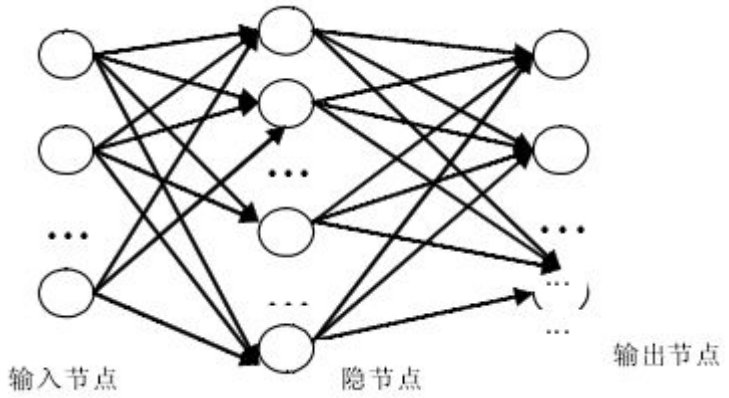

图 10 三层 BP 网络[1]

2.1 BP 网络的拓扑结构

BP 网络是一种大规模并行分布式的单向传播的多层前向网络，其结构如图 10 所示，网络同层节点之间没有任何耦合，输入信号从输入节点依次传过神经网络各隐节点，然后传到输出节点，每一层节点的输出只影响下一层节点的输出。输入层神经元仅仅起传递信息的作用，与隐藏层和输出神经元可以对信息进行加权阈值求和及非线性特征函数变换不同。在多层网络中，一般至少有 3 个层：一个输入层、一个输出层、一个或多个隐层。多层网络可以解决很多单层网络无法解决的问题，比如多层网络可以用来进行非线性分类，可以用来做精度极高的函数逼近，只要有足够多的层和足够多的神经元，这些都可以办到。一个多层网络(图 10)的输入和输出层的神经元个数是由外部描述定义的。

BP 网络可以看作是一个从输入到输出的高度非线性映射，具有的主要特点体现在信息处理的并行性，分布式的信息存储，自组织和自适应性，具有很强的学习和联想功能以及容错性并且由此而来的泛化能力。BP 网络已经成为神经元网络的重要模型之一，在很多领域都得到了应用。

2.2 BP 网络的学习过程

BP 网络的学习过程是有教师学习，训练集包含 M 个样本，对第 P 个训练样本，单元的实际输出为 a_{pj} ， 它的第 i 个输入（也即第 i 个神经元的输出）为 a_{pj}，则：

[1] 唐伟成.手写英文字符识别系统[D].2009.6.

$$net_{pj} = \sum_{j=0}^{n} W_{j,i} a_{pj}$$

BP 算法中大多选用 S 型函数作为输出函数，即：

$$a_{p,j} = f(net_{pj}) = \frac{1}{1+\exp(net_{pj})}$$

定义网络误差函数为：

$$E = \sum_{p} E_P$$

$$E_P = \frac{1}{2}\sum (dpj - apj)^2$$

其中 d_{pj} 对单个训练样本，单元 j 的期望输出。训练网络的目的是找到一组权重，使误差函数极小化。其中权值的校正量为：

$$\Delta W_{ij} = -\frac{\partial E_P}{\partial W ij}$$

令 $-\frac{\partial E_P}{\partial W ij} = C_{pj}$ ，则 $\frac{\partial E_P}{\partial W ij} = \frac{\partial E_P}{\partial netpj}\frac{\partial netpj}{\partial W ij} = \frac{\partial E_P}{\partial W ij} a_{pj} = -C_{pj} a_{pj}$

得到：$\Delta W_{ij} = \alpha C_{pj} a_{pj}$ ，$\alpha > 0$ 表示学习系数。

BP 网络的学习过程的主要步骤如图 11 所示。

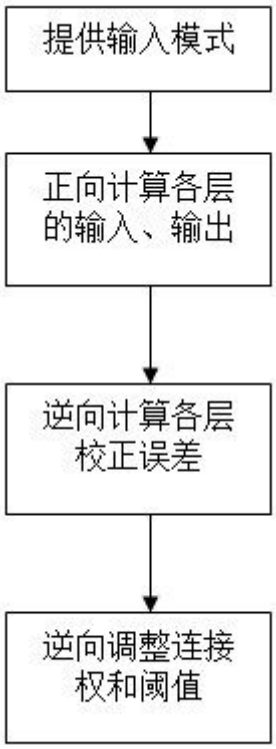

图 11 学习过程主要步骤

在以上的学习步骤中，用激活函数计算中间层各单元和输出层各单元的输入、输出过程是学习模式的“顺传播过程”，而计算输出层各单元和输入层各单元的校正误差为网络误差的“逆传播过程”，最后的更新学习输入模式以及误差判断、学习次数判断则是完成训练和收敛过程。

BP 网络的优点如下：

（1） 对问题的先验知识要求少；

（2） 可实现对特征空间较为复杂的划分；

（3） 适合用高速并行处理系统来实现。

2.3 BP 网络的应用

BP 网络作为一种很重要的神经网络模型在很多领域都得到了应用。

2.3.1 图像和压缩编码

Ackley 和 Hinton 等人提出了利用 BP 网络实现数据编码的基本思想。其原理是把一组输入模式通过少量的隐藏层节点映射到一组输出模式，并使输出模式等同于输入模式。当中间隐藏层的节点数比输入模式维数少时，就意味着隐藏层能更有效地表现输入模式，并把这种表现传给输出层。在这个过程中，输入层和隐藏层的变换可以看成是压缩编码的过程；而隐藏层和输出层的变换可以看成是解码过程[1]。

用 BP 网实现图像压缩时，只需一个隐藏层。输入层和输出层均含有 n*n 个神经元，每个神经元对应于 n*n 个图像分块中的一个像素。隐藏层神经元的数量由图像压缩比决定，如 n=16 时，取隐藏层神经元数为 m=8，则可将 256 个像素的图像块压缩为 8 像素。设用于学习的图像有 N*N 个像素，训练时从中随机抽取 n*n 个图像块作为训练样本，并使教师模式和输入模式相等。通过调整权值使训练集图像的重建误差达到最小。训练后的网络就可以用来执行图像的数据压缩任务了，此时隐藏层输出向量便是数据压缩结果，而输出层的输出向量便是图像重建结果。

在本论文的第 4 章中对此应用进行了实现。

2.3.2 人脸识别

对人脸识别是人类最伟大的视觉功能之一，神经元网络受动物神经系统启发，利用大量简单处理单元互联而构成的复杂系统，以解决复杂模式识别和行为控制问题。将 BP 网络用于人脸识别，建立了人脸识别模型，通过对输入图像实行图像压缩、图像抽样以及输入矢量标准化等图像预处理，将标准化矢量输入 BP 神经网络进行训练。BP 网络用于人脸识别时，网络的每一个输入节点对应样本的一个特征，而输出节点数等于类别数，一个输出节点对应一个类。在训练阶段，如果输入训练样本的类别标点是 i，则训练时的期望输出假设第 i 个节点为 1，而其余输出节点均为 0。在识别阶段，当一个未知类别样本作用到输入端时，考察各输出节点对应的输出，并将这个样本类别判定为具有最大值的输出节点对应的类别。如果有最大值的输出节点与其它节点之间的距离较小（小于某个阈值），则作出拒绝判断。经过竞争选择，获得识别结果。

[1] 周政.BP 神经网络的发展现状综述.山西电子技术[J].2008,2:14-16.

2.3.3 故障诊断

对于故障诊断而言，其核心技术是故障模式识别。而人工神经网络由于其本身信息处理特点，如并行性、自学习、自组织性、联想记忆等，使得能够出色地解决那些传统模式识别难以圆满解决的问题，所以故障诊断是人工神经元网络的重要应用领域之一，已经有不少应用系统的报道。总的来说，神经元网络在诊断领域的应用研究主要集中在两个方面：一是从模式识别的角度应用作为分类器进行故障诊断，其基本思想是：以故障征兆作为人工神经元网络的输入，诊断结果作为输出；二是将神经元网络与其它诊断方法相结合而形成的混合诊断方法。对用解析方法难以建立系统模型的诊断对象，人工神经元网络有着很好的研究和应用前景。

2.3.4 最优预测

目前，前景预测已经成为许多行业不可避免的一个难题。由于预测涉及的因素很多，往往很难建立一个合理的模型。人工神经元网络模拟人的大脑活动，具有极强的非线性逼近、大规模并行处理、自训练学习、容错能力以及外部环境的适应能力。所以利用人工神经网络进行预测已经成为许多项目首选的方法。目前利用 BP 网络进行预测的应用已经很多。例如，可以用来建立公共卫生事件检测与预警系统、旅游业趋势预测系统、物流预测系统、资源调度系统等方面。设计涉及训练样本集设计、网络结构设计和训练与测试三个方面。

BP 网络在应用于预测预报前，需要一个网络学习过程。其学习过程包括信息正向传播和误差反向传播两个反复交替的过程。网络根据输入的训练样本进行自适应、自组织，确定各神经元的连接权 W 和阈值，经过多次训练之后，网络就具有了对学习样本的记忆和联想的能力。

3. 基于 BP 网络的字符识别器的设计

3.1 问题分析

本论文中所设计的字符识别器是可以实现对字符进行简单识别的程序。将待识别的字符作为输入模式提供给网络，并设定字符所对应的希望输出。BP 网络通过反复的顺传播和逆传播的学习过程之后，程序所输出的识别结果与实现结果之间的差值小于允许的最大误差值，则字符就被成功识别，否则识别失败。

例如现在要利用一 BP 网络来识别如图 12 所示的三种图形。识别过程为系统经过基于 BP 算法的反复学习之后正确识别出图形。首先设定实验参数，将图形以数组的方式输入，通过隐藏层节点映射到一组输出模式，当输出模式与希望输出之间的差值小于允许的最大误差值时，待识别图形就被成功识别了。

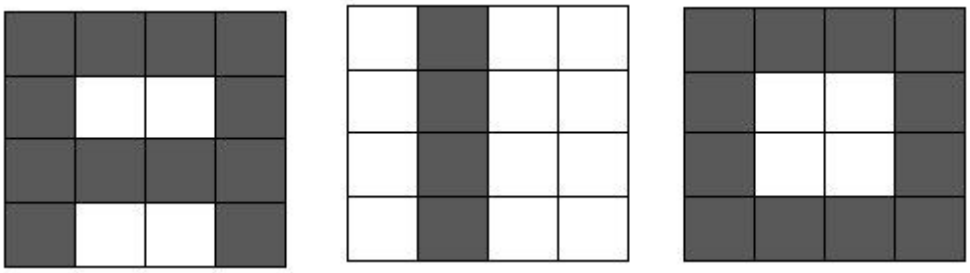

图 12 待识别图形

3.2 BP 网络结构的确定

（1）神经元个数的选取

对于 BP 神经网络，其结构的确定包括了输入层、输出层节点个数的确定和隐藏层层数、隐藏节点个数的确定。其中输入层和输出层节点个数是根据实际问题确定的。在本例中，根据图 12 的待识别图形可以看出输入层的节点个数应设为 16 个（因为 16 个节点可以表示出一个字母），而输出层节点个数设为 3 个，表示期望的三个不同的输出。关于隐藏层数和隐藏节点个数的确定一直是人工神经元网络设计的难点，这是因为隐藏层神经元数与所需解决的问题的特定要求、输入层和输出层神经元的数量以及训练样本的数量等因素有直接关系，故隐藏层神经元网络的数量问题没有精确解只有次优解。隐藏层神经元的数量不能太少，太少会使得网络解决问题的精度不够。而最关键的是隐藏层的神经元数量不能过多，在过多的情况下对于样本学习的精度很容易达到，但是网络很可能出现过拟合，造成泛化能力下降，容错性差。通常是通过经验来确定隐藏层的神经元个数，在本例中将其设定为 8 个。

（2）初始权值的选取

由于系统是非线性的，初始权值对于学习是否达到局部最小、是否能够收敛以及训练时间的长短关系很大。如果初始权值太大，使得加权后的输入落在激活函数的饱和区，从而使得调节过程几乎停顿下来。所以，一般总是希望经过初始加权后的每个神经元的输出值都接近于零，这样可以保证每个神经元的权值都能够在她们的激活函数变化最大之处进行调节。所以，一般取初始权值在（-1，1）之间的随机数。

（3）学习系数的选取

学习系数决定每一次循环训练中所产生的权值变化量。太高的学习系数可能导致系

统的不稳定，但太小的学习系数将导致训练时间较长，收敛速度很慢，不过能保证网络的误差值不超出误差表面的低谷而最终趋于最小误差值。所以在一般情况下，倾向于选取较小的学习系数以保证系统的稳定性。在本例中将学习系数选取为 0.01。

（4） 允许最大学习次数的选取

允许最大学习次数选取的关键是要保证在其范围内能够完成正确识别的功能，即在允许的学习次数内，误差值能够达到期望的精度，小于允许最大误差值。当在允许最大学习次数的范围内，不能够达到期望精度，则学习失败。在本例中，先将其值设定为 3000。

本例中的具体设定如下：

首先将这三个图形用三个输入模式向量 X_1， X_2， X_3 来表示，其中有颜色的部分为 1，没有颜色的部分为 0。即：

$X_1 = [1,1,1,1,1,1,0,0,1,1,1,1,1,1,0,0,1]^T$

$X_2 = [0,1,1,0,0,0,1,0,0,0,0,1,0,0,0,1,1,0,0,0]^T$

$X_3 = [1,1,1,1,1,1,0,0,1,1,1,0,0,1,1,1,1,1,1,1]^T$

这三个输入模式所对应的希望输出为：

$D_1 = [1,0,0]^T$

$D_2 = [0,1,0]^T$

$D_3 = [0,0,1]^T$

该例对应的 BP 网络的输入层有 16 个神经元，输出层有 3 个神经元，隐含层有 8 个神经元。其网络结构如图 13 所示。

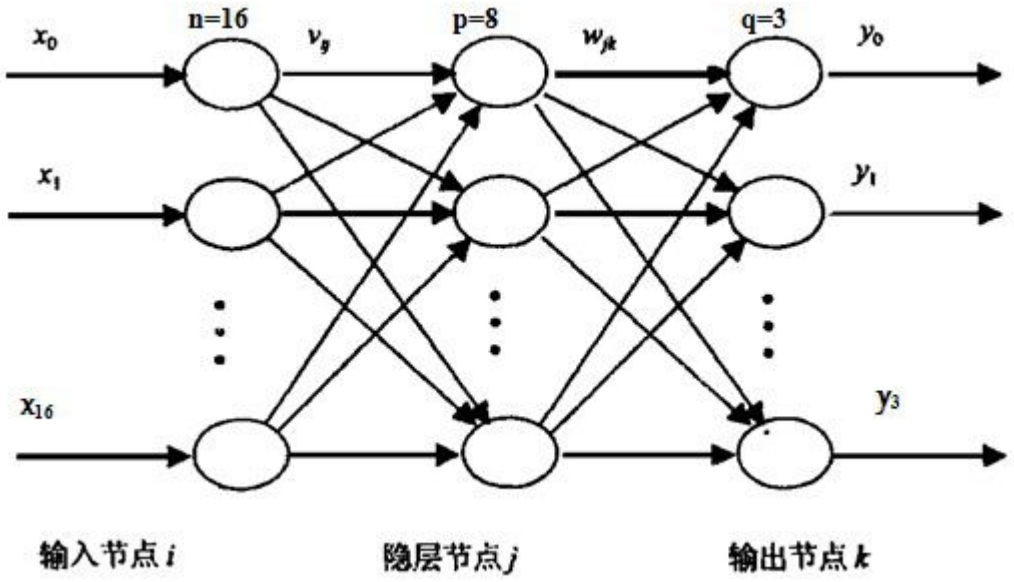

图 13 三层 BP 网络

这里设定允许最大误差值为：

$$E_{max} = 0.01$$

误差的计算方法按照平方和误差计算：

$$E=\frac{1}{2}\sum(d-y)^2$$

学习系数取： $\alpha=\beta=0.1$。

综上所述，实验参数的设定如表格 1 所示。

表格 1 实验参数

输入层神经元个数	16
隐藏层神经元个数	8
输出层神经元个数	3
训练样本个数	3
允许最大学习次数	3000
学习系数	α=β=0.1
允许最大误差值	0.01

3.3 字符识别器的 BP 算法思想的描述

BP 网的学习过程主要由四部分组成[1]：

（1） 输入模式顺传播，即输入模式由输入层经中间层向输出层传播计算。

（2） 输出误差逆传播，即输出的误差由输出层经中间层传向输入层。

（3） 循环记忆训练，即模式顺传播与误差逆传播的计算过程繁复交替循环进行。

（4） 学习结果判别，即判定全局误差是否趋向极小值。

字符识别器的具体学习步骤如下：

（1） 初始化，给各连接权以及阈值赋予（-1，+1）之间的随机值。

（2） 从三种输入模式中选取一个输入模式提供给网络。

（3） 用输入模式的连接权和阈值计算隐含层各神经元的输入（即激活值），然后通过激活函数计算中间层各单元的输出。

（4） 用隐含层的输出、连接权和阈值计算输出层各单元的输入（即激活值），然后用激活函数计算输出层各单元的响应值。

（5） 用希望输出模式和网络实际输出计算输出层各单元的校正误差。

（6） 用隐含层的连接权、输出层各单元的校正误差以及中间层各单元的输出计算中间层的校正误差值。

（7） 计算下一次中间层和输出层之间的新连接权。

（8） 计算下一次的输入层和中间层之间的新连接权。

（9） 选取下一个学习模式对提供给网络，返回第（3）步，直到模式对全部学习完。

（10）重复（3）—（9）步，直到网络全局误差值小于预先设定的限定值（识别成功）或者学习次数超过允许的最大学习次数（识别失败）为止。

（11）学习结束。

其中（3）—（6）步为输入学习模式的顺传播过程，（7）—（8）步为网络误差的逆传播过程，（9）—（10）步为完成训练和收敛的过程。字符识别器的 BP 算法流程图，如图 14 所示。

[1]王旭 王宏 王文辉.人工神经元网络原理与应用[M].沈阳：东北大学出版社.2000.12.

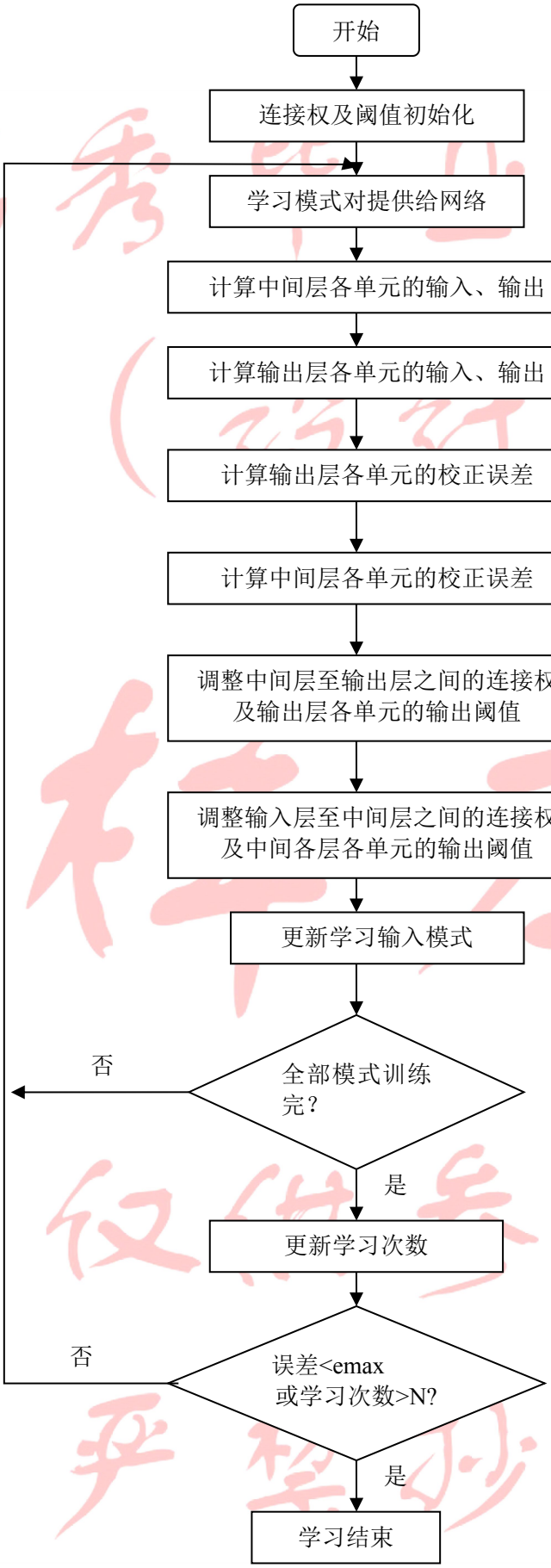
开始
连接权及阈值初始化
学习模式对提供给网络
计算中间层各单元的输入、输出
计算输出层各单元的输入、输出
计算输出层各单元的校正误差
计算中间层各单元的校正误差
调整中间层至输出层之间的连接权
及输出层各单元的输出阈值
调整输入层至中间层之间的连接权
及中间各层各单元的输出阈值
更新学习输入模式
全部模式训练
完？
否
是
更新学习次数
误差<emax
或学习次数>N?
否
是
学习结束

图 14 BP 算法流程图

4. 基于 BP 网络的字符识别器的实现

本章根据第 3 章中的设计进行了识别器的实现。

4.1 实验环境

实验具体的实现环境见表格 2。

表格 2 实验环境

操作系统	Windows 7 旗舰版
内存大小	2.0 GB
硬盘大小	320GB
软件版本	C-Free 5

4.2 字符识别器的算法实现

算法[1]伪代码如下：

```
头文件；
main(){

声明变量；
初始化输入样本 x1 , x2 , x3；
初始化期望输出 y1 , y2, y3；
初始化输入层和中间层的连接权 w；
初始化中间层和输出层的连接权 v；
初始化中间层和输出层的阈值 t1,t2；

do{                                                    //一次循环表示一次学习过程
    while 样本数 <= 3 {                                //每一次学习都要训练三个样本
        选定要学习的样本，将样本赋值给 x0；
        用 S 型激励函数计算中间层的实际输出 bj；
        用 S 型激励函数计算输出层的实际输出 ct；
        计算误差值 er；
        计算输出层各单元的校正误差 dt；
        计算中间层各单元的校正误差 ej；
        计算下一次学习时中间层和输出层之间的新连接权和新阈值；
```

[1]该算法的 C 语言程序见附录 1.

```
            计算下一次学习时输入层和中间层之间的新连接权和新阈值；
            样本号 cp++；
            }

            学习次数 cnt++；

}while 误差值 er < 允许最大误差值 Emax 或者 学习次数 <3000 ；

if er < emax {
            学习成功；
            输出输入层和中间层之间的连接权；
            输出中间层和输出层之间的连接权；
            输出完成学习后中间层的新阈值；
            完成学习后输出层的新阈值；
}

else{
            学习失败；
}
```

程序的运行结果如图 15-图 18 所示。其中图 15 显示了初始化输入层和中间层的连接权 w、初始化中间层和输出层的连接权 v、初始化中间层和输出层的阈值 t1，t2；图 16 和图 17 显示了经过成功学习之后调整的输入层和中间层之间的连接权、中间层和输出层之间的连接权；图 18 显示了完成学习后中间层的新阈值、完成学习后输出层的新阈值以及误差值和完成学习的次数。

其中可以从图 18 看出本次程序的运行结果成功，误差值小于允许最大的误差值，达到了所需精度。

```
Initiative data of w(i,j)
0.87799  -0.19266  -0.63939  0.85150  -0.59752  -0.74798  0.47569  -0.35282
-0.87207  0.13236  0.48125  -0.75829  0.85778  -0.64092  -0.31919  0.61657
0.97308  0.81371  0.35734  0.14237  -0.29777  0.78826  -0.29875  -0.95447
-0.02719  -0.31919  -0.02298  -0.84100  -0.59203  0.25578  0.98975  -0.22868
-0.97034  -0.21586  0.91583  -0.61431  -0.76836  0.37461  0.55876  -0.79833
0.58446  0.80908  -0.90613  0.13181  0.03464  0.19364  0.36344  0.16648
0.06156  0.73998  -0.21543  -0.10184  -0.64684  -0.01352  0.93439  -0.98077
0.56438  -0.22538  -0.12845  -0.39610  -0.28086  0.29075  0.26035  0.98883
0.82067  0.48772  -0.64312  0.06290  0.22996  -0.80303  -0.07859  -0.21928
0.37754  0.24534  0.11905  -0.28971  -0.63109  -0.11228  0.47124  -0.57103
0.04727  -0.13108  -0.07962  0.57933  -0.92419  0.42680  0.07028  0.39756
-0.39787  0.54924  -0.33494  0.52751  0.06699  -0.20212  0.36857  -0.06821
0.72008  -0.98962  -0.09287  -0.28660  0.40562  0.19419  -0.36161  0.66015
-0.82037  0.92425  0.60790  -0.67669  -0.67785  -0.44292  -0.84295  -0.55437
0.30583  0.57408  0.14402  0.54759  0.28471  0.36540  0.84008  0.48766
-0.33518  0.76879  -0.43480  -0.19285  -0.68090  0.67412  0.12345  0.67309

Initiative data of v(i,j)
-0.99030     -0.99304     0.10215
0.23051     0.57945     -0.59612
0.09684     -0.82171     -0.52440
-0.10886     -0.07578     -0.62035
-0.83178     -0.11997     0.88635
-0.67962     0.02512     -0.20859
0.14743     -0.77270     -0.79315
-0.37883     0.28715     0.65917

Init data of t1
-0.85656
-0.24009
0.09189
0.38298
-0.84429
0.37883
-0.35350
-0.65203

Init data of t2
-0.21622
0.69475
-0.20994
0.56963
0.38481
0.42619
0.99658
0.73614
```

图 15 初始化

```
Successful , the results are
1.输入层和中间层之间的连接权为：w[i , j]
w[0 , 0]: 1.2718
w[0 , 1]: -0.7161
w[0 , 2]: -0.3928
w[0 , 3]: 0.9938
w[1 , 0]: -1.2714
w[1 , 1]: 0.4248
w[1 , 2]: 0.6631
w[1 , 3]: 0.0225
w[2 , 0]: 1.3669
w[2 , 1]: 0.2903
w[2 , 2]: 0.6039
w[2 , 3]: 0.2846
w[3 , 0]: 0.3666
w[3 , 1]: -0.8426
w[3 , 2]: 0.2236
w[3 , 3]: -0.6987
w[4 , 0]: -0.5766
w[4 , 1]: -0.7393
w[4 , 2]: 1.1624
w[4 , 3]: -0.4720
w[5 , 0]: -0.2086
w[5 , 1]: 1.6250
w[5 , 2]: -0.9708
w[5 , 3]: 0.7703
w[6 , 0]: 0.0616
w[6 , 1]: 0.7400
w[6 , 2]: -0.2154
w[6 , 3]: -0.1018
w[7 , 0]: 0.9581
w[7 , 1]: -0.7488
w[7 , 2]: 0.1181
w[7 , 3]: -0.2538
w[8 , 0]: 1.2144
w[8 , 1]: -0.0357
w[8 , 2]: -0.3965
w[8 , 3]: 0.2052
```

图 16 运行成功输出连接权

```
w[10 , 2]: 0.7093
w[10 , 3]: 1.4119
w[11 , 0]: -0.0041
w[11 , 1]: 0.0258
w[11 , 2]: -0.0883
w[11 , 3]: 0.6698
w[12 , 0]: 1.1139
w[12 , 1]: -1.5131
w[12 , 2]: 0.1537
w[12 , 3]: -0.1443
w[13 , 0]: -1.1320
w[13 , 1]: 0.8407
w[13 , 2]: 0.0009
w[13 , 3]: -0.7285
w[14 , 0]: 0.7873
w[14 , 1]: -0.3254
w[14 , 2]: -0.3983
w[14 , 3]: -0.1427
w[15 , 0]: 0.0586
w[15 , 1]: 0.2454
w[15 , 2]: -0.1882
w[15 , 3]: -0.0506

2.中间层和输出层之间的连接权为：v[i , j]
v[0 , 0]: 0.3748
v[0 , 1]: -3.7793
v[0 , 2]: 0.8615
v[1 , 0]: -1.4384
v[1 , 1]: 2.8808
v[1 , 2]: -3.7844
v[2 , 0]: 0.4101
v[2 , 1]: -1.5069
v[2 , 2]: -1.5249
v[3 , 0]: 0.5929
v[3 , 1]: -0.0569
v[3 , 2]: -2.1764
v[4 , 0]: -4.1137
v[4 , 1]: 0.1919
v[4 , 2]: 1.7758
v[5 , 0]: -0.9605
v[5 , 1]: 0.6982
v[5 , 2]: -1.5988
v[6 , 0]: 1.9398
v[7 , 0]: -2.1801
v[7 , 1]: -0.2084
v[7 , 2]: 1.5045
```

图 17 运行成功输出连接权（接上图）

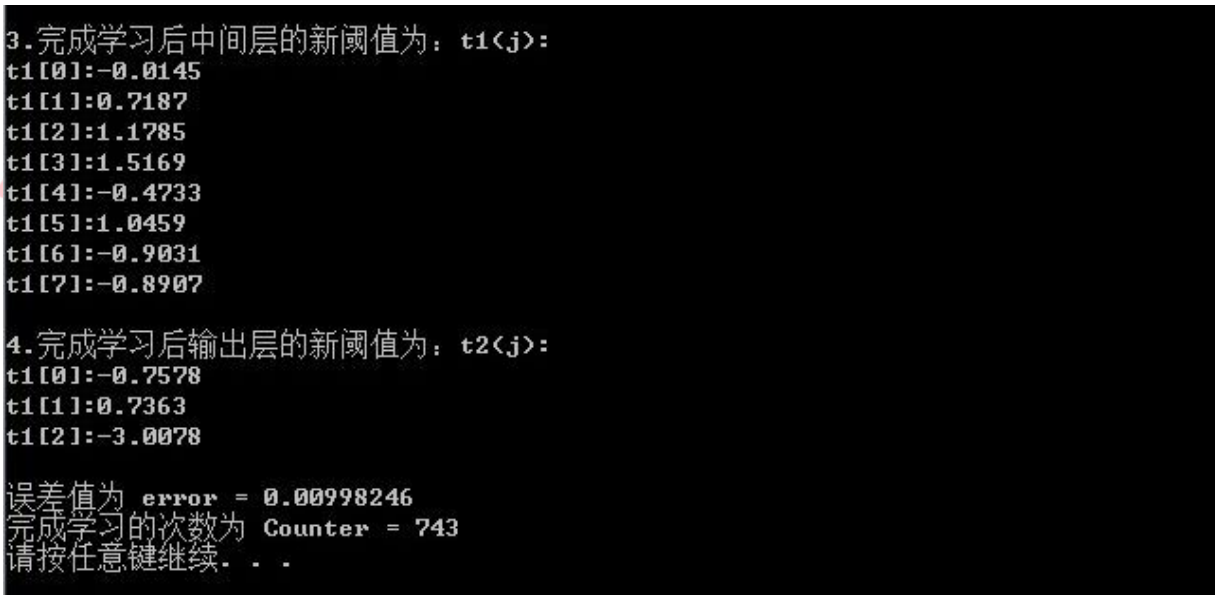

图 18 输出新阈值

4.3 实验结果

表格 3 是对 20 次实验结果的记录。

表格 3 实验结果

序号	学习是否成功	完成学习的次数	误差值	结果分析
1	是	609	0.00999881	成功识别
2	是	664	0.00998786	成功识别
3	是	856	0.00998983	成功识别
4	是	699	0.00999482	成功识别
5	是	842	0.00999512	成功识别
6	是	889	0.00999316	成功识别
7	是	691	0.00998495	成功识别
8	是	662	0.00999111	成功识别
9	是	694	0.00999079	成功识别
10	是	724	0.00998688	成功识别
11	是	757	0.00999421	成功识别
12	是	608	0.00999835	成功识别
13	是	778	0.00998662	成功识别
14	是	709	0.00999037	成功识别
15	是	660	0.00998820	成功识别
16	是	675	0.00999946	成功识别
17	是	562	0.00997647	成功识别
18	是	827	0.00999573	成功识别
19	是	621	0.00999629	成功识别
20	是	849	0.00999169	成功识别

由表格 3 所记录的 20 次实验结果可以看出，该算法的运行结果较为稳定，基本上在允许的范围内都能够实现识别字符的功能，成功完成学习的次数以及误差均存在于合理

的范围之内。其中在 3.2 节中设定误差值应小于 0.01，允许最大学习次数为 3000 次。

在允许最大学习次数为 3000 次的情况下，让程序自动进行 1000 次反复试验，试验结果如图 19 所示。此时学习成功率为 100%。

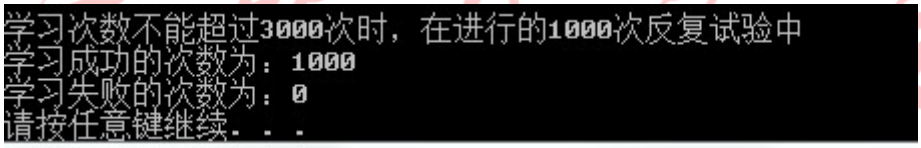

图 19 1000 次实验结果 1

当将允许最大学习次数缩小为 1000 次时，让程序自动进行 1000 次反复试验，试验结果如图 19 所示。此时学习成功率为 91.8%。

图 20 1000 次实验结果 2

总结实验结果 1 和实验结果 2，可以看出，当学习次数可以多达 3000 次时基本能够保证正确识别字符；当将允许最大学习次数缩小为 1000 次时，并不能保证完成识别字符的功能，但是能够缩短程序的运行时间，当程序需要识别的输入模式对的数量庞大时，这种时间的节省有着十分重要的意义。

5 对传统 BP 算法的改进

5.1 BP 算法存在缺陷的原因

BP 神经网络的具有如下缺点[1][2]：

（1） 网络学习收敛速度太慢。

由于学习速率是固定的，因此网络的收敛速度慢，需要较长的训练时间，而对于一些复杂问题，BP 算法需要的训练时间可能非常长，这主要是由于学习速率太小造成的，可以采用变化的学习速率或自适应的学习速率加以改进。

（2） 不能保证收敛到全局最小点。

BP 算法可以使权值收敛到某个值，但并不保证所求为误差超平面的全局最小值，而有可能是一个局部极小值。这是因为 BP 算法采用的是梯度下降法，训练是从某一起点沿误差函数的斜面逐渐达到误差的极小值，对于复杂的网络，其误差函数为多维空间的曲面，就像一个碗，其碗底是最小值点，但是这个碗的表面是凹凸不平的，因而在对其进行训练的过程中，可能陷入某一小谷区，而这一小谷区产生一个局部最小值，由此点向各个方向变化均使误差增加，以至于使训练无法逃出这一局部最小值。

（3） 网络的隐含层的层数以及它的单元数的选取没有理论上的指导，而是根据经验确定的，因此网络的设计不是最佳的。

（4） 网络的学习、记忆具有不稳定性。

5.2 改进的 BP 算法

5.2.1 累积误差校正算法

在 4.2 节中提出的 BP 算法对每一次输入都校正一次权值，浪费了学习时间。这种算法不是全局误差意义上的梯度下降计算，为了克服这个缺点，本节对该字符识别器的实现提出了改进方法，即累积误差校正算法。

累积误差校正算法是真正的全局误差意义上的梯度下降算法，计算方法流程图如图 21[3]所示。

此算法的校正次数减少，仅当每学习完一次之后才校正，而上文中的 BP 算法是每计算完一个模式都要进行校正。因此本算法的学习时间可以缩短，但不足之处在于会使得学习模式的误差平均化，在某些情况下容易引起震荡。

[1]王旭 王宏 王文辉.人工神经元网络原理与应用[M].沈阳：东北大学出版社.2000.12.
[2]储琳琳 郭纯生.浅析 BP 神经网络算法的改进和优化.科技创新导报[J]. 2009,(12):4-5
[3]王旭 王宏 王文辉.人工神经元网络原理与应用[M].沈阳：东北大学出版社.2000.12.

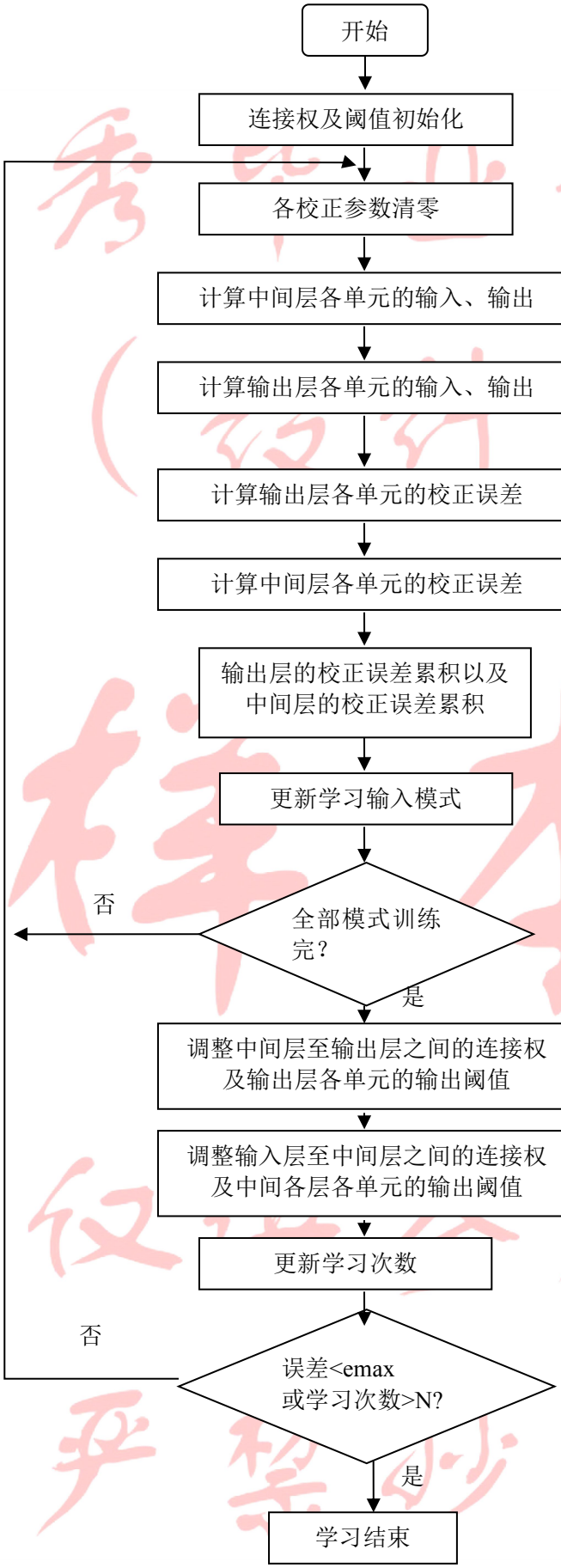

图 21 累积误差校正算法流程图

用累积误差校正算法[1]对字符识别器实现的伪代码如下：

```
头文件；
main(){
声明变量；
初始化输入样本 x1 , x2 , x3；
初始化期望输出 y1 , y2, y3；
初始化输入层和中间层的连接权 w；
初始化中间层和输出层的连接权 v；
初始化中间层和输出层的阈值 t1,t2；
do{                                                    //一次循环表示一次学习过程
    while 样本数 <= 3 {                                //每一次学习都要训练三个样本
        选定要学习的样本，将样本赋值给 x0；
        用 S 型激励函数计算中间层的实际输出 bj；
        用 S 型激励函数计算输出层的实际输出 ct；
        累积误差值 er；
        计算输出层各单元的校正误差 dt；
        计算中间层各单元的校正误差 ej；
        输出层误差累积；
        中间层误差累积；

        样本号 cp++；
        }
        计算平均误差值 er;
        计算下一次学习时中间层和输出层之间的新连接权和新阈值；
        计算下一次学习时输入层和中间层之间的新连接权和新阈值；

        学习次数 cnt++；

}while 误差值 er < 允许最大误差值 Emax 或者 学习次数 < 3000 ；

if er < emax {
        学习成功；
        输出输入层和中间层之间的连接权；
        输出中间层和输出层之间的连接权；
        输出完成学习后中间层的新阈值；
        完成学习后输出层的新阈值；
}
else{
        学习失败；
}
```

[1]该算法的 C 语言程序见附录 2.

用累积误差校正算法实现字符识别器的实验运行结果如图 22 所示。

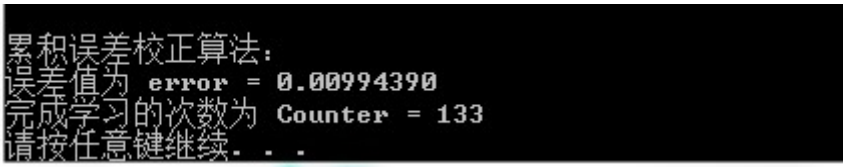

图 22 实验运行结果

表格 4 是次程序运行的 20 次实验结果。将表格 4 与表格 3 进行比较（见表格 5），可以看出累积误差校正算法使得完成学习的次数明显减少，学习时间明显缩短。当程序需要识别的输入模式对的数量庞大时，这种时间的节省有着十分重要的意义。

表格 4 20 次实验结果记录

序号	学习是否成功	完成学习的次数	误差值	结果分析
1	是	133	0.00994390	成功识别
2	是	181	0.00999313	成功识别
3	是	110	0.00993318	成功识别
4	是	144	0.00999476	成功识别
5	是	137	0.00993990	成功识别
6	是	122	0.00991837	成功识别
7	是	173	0.00994986	成功识别
8	是	203	0.00994708	成功识别
9	是	148	0.00993679	成功识别
10	是	154	0.00996291	成功识别
11	是	183	0.00999198	成功识别
12	是	170	0.00995069	成功识别
13	是	152	0.00991020	成功识别
14	是	92	0.00990484	成功识别
15	是	137	0.00992072	成功识别
16	是	174	0.00999370	成功识别
17	是	166	0.00993298	成功识别
18	是	103	0.00990787	成功识别
19	是	183	0.00995476	成功识别
20	是	226	0.00996255	成功识别

表格 5 两种算法的比较

	传统 BP 算法	累积误差校正算法
完成识别功能的平均学习次数	718 次	154 次

5.2.2 其它的改进方法

（1）有条件地改变参数的值

传统的 BP 算法中，学习率和学习因子是不变的，为了改善网络的收敛能力，可以使用类似退火算法的方案，其核心思想是：初始时，将学习率和学习因子都设置为较高的值，一般定为 0.7-0.9，随着学习次数增加，学习率和学习因子递减，递减的规律视问题而定，当其递减到一定程度时，如果网络仍然没有收敛或者误差仍然没有改善时，重新设置学习率和学习因子的值，一般定为 0.5-0.7，再进入学习过程，直到运行结束。这种改进是有条件的改进，在误差长时间没有得到改善或网络达不到收敛的目的的时候，通过直接设定这两个参数的值，适当增大网络权值的修改量，经过再学习的过程，使网络能达到收敛，且避免进入局部极小。这种方案虽然改善了局部极小值问题，但是网络的学习速度却因此而变得较慢[1]。

（2）改变激活函数

可以使用三角函数作为激活函数，但并不是所有的三角函数都是可以用来作为 BP 网络的激活函数的，也不是仅仅依靠一个三角函数可以成为有效的激活函数的。

也可以使用双极性压缩函数，其表达式为：$y = -1/2 + 1/[1+\exp(-x)]$。在 BP 算法中，它的一阶导数是 $y = 1/4 - y^2$。使用这个函数可以大大加快收敛速度，通常可以加快 30%-50%，这显然是一个十分有效的改进。

[1] 陆琼瑜 童学峰.BP 算法改进的研究.计算机工程与设计[J].2007, 28(3):648-650.

6. 展望与总结

人工神经元网络已经取得了巨大的进步，它拓展了人们对外部环境的认识和控制能力，它特有的信息处理能力使之在智能控制，预测等领域取得了广泛的应用，它是一门独特的信息处理学科。人工神经元网络还有很大的发展空间，在将来它将往模拟人类认知的道路上深入发展，与模糊系统、进化机制等结合，形成计算智能，成为人工智能的一个重要方面。

基于BP神经网络系统的字母识别技术虽然在各个环节中都已经取得了辉煌的成绩，但是在多元化的今天，神经网络技术将和遗传算法，小波变换，智能技术相结合完善识别这一科学领域。

6.1 对人工神经元网络的展望

6.1.1 神经生理学、神经解剖学的研究与发展

通过神经网络研究的发展，人们对人脑一些局部功能的认识已经有所提高，如对感知器的研究，对视觉处理网络的研究，对存储与记忆问题的研究等都取得一定的成功。但是这些成功一方面还远不够完善，另一方面，在对人脑作为一个整体的功能的解释上几乎起不到任何作用。科学家已经累积了大量关于大脑组成、大脑外形、大脑运转基本要素等知识，但仍无法解答有关大脑信息处理的一些实质问题[1]。

整体功能绝不是局部功能的简单组合，而是一个巨大的质的飞跃。人脑的知觉和认知等过程是包含着一个复杂动态系统中对大量神经元活动进行整合的统一性行动。由于人们对人脑完整工作过程几乎没有什么认识，这使得人工神经元网络研究始终缺乏一个明确的大方向。只有在这个方向上有所突破，神经元网络的研究才能达到质的飞跃。

而BP网络的自学习很适用于进行这一方面的研究。

6.1.2 与之相关的数学领域的研究与发展

神经元以电为主的生物过程在认识上一般采用非线性动力学模型，其动力学演变过程是非常复杂的，神经元网络这种很强的生物学特征和数学特征性质要求有更好的数学手段[2]。而对于解决非线性微分方程这样的问题，稍微复杂一些的则无法用数学方法求的完整的解。这使得很难去分析诸如一般神经网络的震荡性、稳定性、混沌等问题。

因此，当今神经网络理论的发展已经在客观上要求有关数学领域必须有所发展，并且这一领域的数学发展的重要目标之一则是发现一种更简洁、更完善和更有效的非线性系统的表达与分析方法。

6.1.3 神经元网络应用的研究与发展

从神经网络发展的过程来看，理论的研究有时会超出实际使用阶段，但实际需求是科技发展的主要动力。目前，在神经网络的实际应用上虽然有不少，如智能控制、模式识别及机器人控制等，但真正成熟的应用还比较少见。

[1] 胡守仁 余少波 戴葵.神经网络导论[M].长沙 国防科技大学出版社 1992.
[2] 韩力群.人工神经网络理论、设计及应用[M].北京:化学工业出版社 2002.

6.2 对 BP 网络的展望

在人工神经网络的实际应用中，绝大部分的神经网络模型都采用 BP 网络及其变化形式。它也是前向网络的核心部分，体现了人工神经网络的精华。

BP 网络的主要应用将集中在以下几个方面。

（1）函数逼近：用输入向量和相应的输出向量训练一个网络以逼近一个函数。

（2）模式识别：用一个待定的输出向量将它与输入向量联系起来。

（3）分类：把输入向量所定义的合适模式进行分类。

（4）数据压缩：减少输出向量维数以便传输或存储。

6.3 总结

人工神经元网络实质就是从输入空间到输出空间的一个非线性映射，BP 网络训练是通过不断地调整权值和阈值来学习发现变量之间的关系，即 BP 网络训练的目的就是找到合适的权值和阈值，从而达到对目标的分类和识别[1]。本文论述了人工神经元网络的理论知识，并且用 BP 算法实现了一个字符识别器，但这种传统的 BP 算法具有一定的缺陷，因此本文还提出了几种改进方法，并以其中一种改进算法，即累积误差校正算法，实现了字符识别器，明显缩短了学习所需时间。

[1]曾志军 孙国强.基于改进的 BP 网络数字字符识别.上海理工大学学报[J].2008,30(2):201-204.

参考文献

[1]孙佰清 张长胜等.提高 BP 神经网络训练速度的研究[J].哈尔滨工业大学学报 2001,(5):32-36.

[2]（加）Simon Haykin. 神经网络与机器学习[M] 北京：机械工业出版社 2011.

[3]李佳斌. 人工神经网络系统模型原理及其在计算机中的应用研究与进展[J].电脑知识与技术 2011，7(2)：411-414.

[4]张可,张高燕,吴苏,范海菊.基于 BP 神经网络的字符识别系统.计算机与现代化[J].2009,1:63-72.

[5]（美）Tom M.Mitchell.机器学习[M] 北京：机械工业出版社 2003.

[6]胡瑞敏 徐正全 姚天任 李德仁.人工神经元网络的智能神经元模型.电子学报[J].1996, 24(4)：86-90.

[7](美)Stuart Russell,Peter Norvig.人工智能-一种现代方法（第二版）[M].北京：人民邮电出版社 2004.

[8]唐伟成.手写英文字符识别系统[D].2009.6.

[9]周政.BP 神经网络的发展现状综述.山西电子技术[J].2008,2(5):14-16.

[10]王旭 王宏 王文辉.人工神经元网络原理与应用[M].沈阳：东北大学出版社 2000.12.

[11]储琳琳 郭纯生.浅析 BP 神经网络算法的改进和优化.科技创新导报[J]. 2009,12：4-5.

[12]陆琼瑜 童学峰.BP 算法改进的研究.计算机工程与设计[J].2007,28(4):648-650.

[13]胡守仁 余少波 戴葵.神经网络导论[M].长沙：国防科技大学出版社 1992.

[14]韩力群.人工神经网络理论、设计及应用[M].北京:化学工业出版社 2002.

[15]曾志军 孙国强.基于改进的BP网络数字字符识别.上海理工大学学报[J].2008,30(2):201-204.

致谢

在本毕业设计论文即将完成之际，我要对所有曾经给过我帮助和支持的老师和朋友表示衷心的感谢。尤其要感谢这几个月以来杨戈老师耐心的指导和帮助，让我对很多领域的认识都有了提升，也锻炼了各方面的能力。杨戈老师对工作认真负责的态度和对我们的严格要求使我受益匪浅。感谢杨戈老师！

附录 1

字符识别器的 C 语言程序如下：

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
int main()
{
    float a1 = 0.1 , b1 = 0.1;//a1 和 b1 是学习系数
    int n = 16 , p = 8 , q = 3;//n 是输入层的神经元个数，p 是中间层神经元的个数，q 是输出层神经元的个数
    int cnt = 0;//cnt 是学习的次数
    float emax = 0.01;//允许的最小误差
    float w[n][p];//输入层和中间层之间的连接值
    float v[p][q];//中间层和输出层之间的连接值
    float t1[8] , t2[3] ;//t1 是中间层的阈值， t2 是输出层的阈值
    int x0[4][4] ;
    int y0[4];
    double bj[p];//存放中间层的激活函数值
    double ct[p];//存放输出层的激活函数值
    float er;//er 表示误差值
    int cp ; //输入样本的个数

    //初始化输入
    int   x1[4][4] = {1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1};
    int x2[4][4] = {0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0};
    int x3[4][4] = {1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1};


    //初始化期望输出
    int y1[3] = {1, 0, 0};
    int y2[3] = {0, 1, 0};
    int y3[3] = {0, 0, 1};

    //初始化输入层和中间层的连接权
    int i,j;
    srand(time(NULL));
    printf("Initiative data of w(i,j)\n");//初始化输入层的连接权
    for(i = 0;i < n;i++)
    {
        for(j = 0 ; j < p ; j++)
        {

            float temp_r1 ;
            temp_r1 = rand()*1.0/RAND_MAX ;//生成一个 0—1 的随机数
            w[i][j] = 2 * temp_r1 - 1;//生成从-1 到 1 的随机数
            printf("%1.5f    ", w[i][j]);
        }
        printf("\n");
    }
```

```
//初始化中间层和输出层之间的连接权
printf("\nInitiative data of v(i,j)\n");
for(i = 0;i < p;i++)
{
    for(j = 0 ; j < q ; j++)
    {
        float temp_r2 ;
        temp_r2 = rand()*1.0/RAND_MAX ;//生成一个 0—1 的随机数
        v[i][j] = 2 * temp_r2 - 1;//生成从-1 到 1 的随机数
        printf("%1.5f          ", v[i][j]);
    }
    printf("\n");
}

//初始化中间层和输出层的阈值
 printf("\nInit data of t1\n"); //中间层的阈值
 int k , l;
 for(k = 0; k < p ; k++)
 {
        float temp_r1 ;
        temp_r1 = rand()*1.0/RAND_MAX ;//生成一个 0—1 的随机数
        t1[k] = 2 * temp_r1 - 1;//生成从-1 到 1 的随机数
        printf("%1.5f\n",t1[k]);
}

 printf("\nInit data of t2\n"); //输出层的阈值
 for(l = 0; l < p ; l++)
 {
        float temp_r2 ;
        temp_r2 = rand()*1.0/RAND_MAX ;//生成一个 0—1 的随机数
        t2[l] = 2 * temp_r2 - 1;//生成从-1 到 1 的随机数
        printf("%1.5f\n",t2[l]);
}

do
{
    cp = 1;
    int a, b, c;
    while(cp <= 3)
    {
        switch(cp)//选择学习样本
        {
        case 1:
            for(a = 0; a < 4; a++)
                for(b = 0; b < 4; b++)
                {
                    x0[a][b] = x1[a][b];
                }
            for(c = 0; c < q; c++)
            {
                y0[c] = y1[c];
            }
            break;
```

```
        case 2:
            for(a = 0; a < 4; a++)
                for(b = 0; b < 4; b++)
                {
                    x0[a][b] = x2[a][b];
                }
            for(c = 0; c < q; c++)
            {
                y0[c] = y2[c];
            }
            break;

        case 3:
            for(a = 0; a < 4; a++)
                for(b = 0; b < 4; b++)
                {
                    x0[a][b] = x3[a][b];
                 }
            for(c = 0; c < q; c++)
            {
                y0[c] = y3[c];
            }
            break;
    }


  //计算中间层的实际输出
  int u ,h , d , e , loc ;
  float temp_xy;
  float temp_y[p];
   for(u = 0; u < p; u++)//初始化
        temp_y[u] = 0;

   for(h = 0 ; h < p; h++)
   {
        for(d = 0; d < 4; d++)
            for(e = 0; e < 4; e++)
            {
                loc = 4 * d    + e;//计算数的位置
                temp_xy = w[loc][h] * x0[d][e];
                temp_y[h] = temp_y[h] + temp_xy;

            }

   }

   //计算中间层的激活函数值
     for(i = 0; i < p; i++)
     {
        temp_y[i] = temp_y[i] - t1[i];
        bj[i] = 1/(1+exp(-temp_y[i]));
  }

float temp_outy[q] ;

//计算输出层的实际输出
```

```
for(u = 0; u < q; u++)//初始化
      temp_outy[u] = 0;

 for(h = 0 ; h < q; h++)
 {
      for(d = 0; d < p; d++)
           {
                temp_xy = v[d][h] * bj[d];
                temp_outy[h] = temp_outy[h] + temp_xy;
           }

 }

 //计算输出层的激活函数值
  for(i = 0; i < q; i++)
  {
     temp_y[i] = temp_outy[i] - t2[i];

     ct[i] = 1/(1+exp(-temp_outy[i]));

 }

 er = 0;

 //计算误差值
   for(i = 0; i < q; i++)
   {
     er = er + (y0[i] - ct[i]) * (y0[i] - ct[i]);
     }

 //计算输出层各单元的校正误差
 float dt[q];
 for(i = 0; i < q; i++)
   {
     dt[i] = (y0[i] - ct[i]) * (1 - ct[i]);
 }

//计算中间层各单元的校正误差
float ej[p];
for(i = 0; i < p; i++)

{
      temp_xy = 0;
      for(j = 0 ; j < q; j++)
      {
          temp_xy = temp_xy + dt[j] * v[i][j];
      }
      ej[i] = temp_xy * bj[i] * (1 - bj[i]);
}

//计算下一次学习时中间层和输出层之间的新连接权和新阈值
   for(i = 0; i < q; i++)
   {
      for(j = 0; j < p; j++)
      {
           v[j][i] = v[j][i] + a1 * dt[i] * bj[j];//中间层与输出层之间的新连接权
```

```
            }
            t2[i] = t2[i] + a1 * dt[i];//输出层的新阈值
        }

        //计算下一次学习时中间层和输出层之间的新连接权和新阈值
        for(i = 0; i < p; i++)
        {
            for(j = 0; j < 4; j++)
            {
                for(k = 0; k < 4; k++)
                {
                    loc = 4 * j + k;
                    w[loc][i] = w[loc][i] + b1 * ej[i] * x0[j][k];//输入层与中间层的新连接权
                }
            }
            t1[i] = t1[i] + b1 * ej[i];//中间层的新阈值
         }
        cp++;

        }
        cnt++;
    }while(er > emax && cnt<30000);

    if(er < emax)
    {
        printf("\nSuccessful , the results are\n");
        printf("1.输入层和中间层之间的连接权为：w[i , j]\n");//完成学习后输入层和中间层之间的
连接权
        for(i = 0; i < n; i++)
            for(j = 0; j < 4 ; j++)
                printf("w[%d , %d]: %1.4f\n" , i , j , w[i][j]);

        printf("\n2.中间层和输出层之间的连接权为：v[i , j]\n");//完成学习后中间层和输出层之间的
连接权
        for(i = 0; i < p ; i++)
        {
            for(j = 0; j < q ; j++)
            {
                printf("v[%d , %d]: %1.4f\n" , i , j , v[i][j] );
            }
        }

        printf("\n3.完成学习后中间层的新阈值为：t1(j):\n");//完成学习后中间层的新阈值
        for(j = 0 ; j < p ; j++)
        {
            printf("t1[%d]:%1.4f\n" , j , t1[j]);
        }

        printf("\n4.完成学习后输出层的新阈值为：t2(j):\n");//完成学习后输出层的新阈值
        for(j = 0 ; j < q ; j++)
        {
            printf("t1[%d]:%1.4f\n" , j , t2[j]);
        }

        printf("\n 误差值为  error = %1.8f\n" , er);//误差值
        printf("完成学习的次数为  Counter = %d\n" , cnt+1);//完成学习的次数
```

```
    }
    else
    {
        printf("Failed\n");
    }
}
```

附录 2

字符识别器的 C 语言改进程序如下：

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
int main()
{
    float a1 = 0.1 , b1 = 0.1;//a1 和 b1 是学习系数
    int n = 16 , p = 8 , q = 3;//n 是输入层的神经元个数，p 是中间层神经元的个数，q 是输出层神经元的个数
    int cnt = 0;//cnt 是学习的次数
    float emax = 0.01;//允许的最小误差
    float w[n][p];//输入层和中间层之间的连接值
    float v[p][q];//中间层和输出层之间的连接值
    float t1[8] , t2[3] ;//t1 是中间层的阈值， t2 是输出层的阈值
    int x0[4][4] ;
    int y0[4];
    double bj[p];//存放中间层的激活函数值
    double ct[p];//存放输出层的激活函数值
    float er;//er 表示误差值
    int cp ; //输入样本的个数

    //初始化输入
    int   x1[4][4] = {1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1};
    int x2[4][4] = {0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0};
    int x3[4][4] = {1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1};


    //初始化期望输出
    int y1[3] = {1, 0, 0};
    int y2[3] = {0, 1, 0};
    int y3[3] = {0, 0, 1};

    //初始化输入层和中间层的连接权
    int i,j;
    srand(time(NULL));
    printf("Initiative data of w(i,j)\n");//初始化输入层的连接权
    for(i = 0;i < n;i++)
    {
        for(j = 0 ; j < p ; j++)
        {

            float temp_r1 ;
            temp_r1 = rand()*1.0/RAND_MAX ;//生成一个 0—1 的随机数
            w[i][j] = 2 * temp_r1 - 1;//生成从-1 到 1 的随机数
            printf("%1.5f    ", w[i][j]);
        }
        printf("\n");
    }
```

```
//初始化中间层和输出层之间的连接权
printf("\nInitiative data of v(i,j)\n");
for(i = 0;i < p;i++)
{
    for(j = 0 ; j < q ; j++)
    {
        float temp_r2 ;
        temp_r2 = rand()*1.0/RAND_MAX ;//生成一个 0—1 的随机数
        v[i][j] = 2 * temp_r2 - 1;//生成从-1 到 1 的随机数
        printf("%1.5f          ", v[i][j]);
    }
    printf("\n");
}

//初始化中间层和输出层的阈值
 printf("\nInit data of t1\n"); //中间层的阈值
 int k , l;
 for(k = 0; k < p ; k++)
 {
        float temp_r1 ;
        temp_r1 = rand()*1.0/RAND_MAX ;//生成一个 0—1 的随机数
        t1[k] = 2 * temp_r1 - 1;//生成从-1 到 1 的随机数
        printf("%1.5f\n",t1[k]);
}

 printf("\nInit data of t2\n"); //输出层的阈值
 for(l = 0; l < p ; l++)
 {
        float temp_r2 ;
        temp_r2 = rand()*1.0/RAND_MAX ;//生成一个 0—1 的随机数
        t2[l] = 2 * temp_r2 - 1;//生成从-1 到 1 的随机数
        printf("%1.5f\n",t2[l]);
}
float alldt[q];
float allej[p];
float dt[q];
float ej[p];
float temp_outy[q] ;
float allbj[p];
float allct[q];
int u ,h , d , e , loc ;
do
{
    for(i = 0; i < q; i++)
      {
        alldt[i] = 0;
    }

   for(i = 0; i < p; i++)

   {
        allej[p] = 0;
   }
```

```
cp = 1;
int a, b, c;
while(cp <= 3)
{
    switch(cp)//选择学习样本
    {
    case 1:
        for(a = 0; a < 4; a++)
            for(b = 0; b < 4; b++)
            {
                x0[a][b] = x1[a][b];

            }
        for(c = 0; c < q; c++)
        {
            y0[c] = y1[c];
        }
        break;

    case 2:
        for(a = 0; a < 4; a++)
            for(b = 0; b < 4; b++)
            {
                x0[a][b] = x2[a][b];
            }
        for(c = 0; c < q; c++)
        {
            y0[c] = y2[c];
        }
        break;

    case 3:
        for(a = 0; a < 4; a++)
            for(b = 0; b < 4; b++)
            {
                x0[a][b] = x3[a][b];
             }
        for(c = 0; c < q; c++)
        {
            y0[c] = y3[c];
        }
        break;
    }
    for(i = 0; i < q; i++)
    {
      dt[i] = 0;
 }

for(i = 0; i < p; i++)

{
      ej[p] = 0;
}

 //计算中间层的实际输出
```

```
float temp_xy;
float temp_y[p];
 for(u = 0; u < p; u++)//初始化
     temp_y[u] = 0;

 for(h = 0 ; h < p; h++)
 {
     for(d = 0; d < 4; d++)
          for(e = 0; e < 4; e++)
          {
               loc = 4 * d    + e;//计算数的位置
               temp_xy = w[loc][h] * x0[d][e];
               temp_y[h] = temp_y[h] + temp_xy;

          }

 }

 //计算中间层的激活函数值
  for(i = 0; i < p; i++)
  {
     temp_y[i] = temp_y[i] - t1[i];
     bj[i] = 1/(1+exp(-temp_y[i]));
     allbj[i] += bj[i];
}


//计算输出层的实际输出
 for(u = 0; u < q; u++)//初始化
     temp_outy[u] = 0;

 for(h = 0 ; h < q; h++)
 {
     for(d = 0; d < p; d++)
          {
               temp_xy = v[d][h] * bj[d];
               temp_outy[h] = temp_outy[h] + temp_xy;
          }

 }

 //计算输出层的激活函数值
  for(i = 0; i < q; i++)
  {
     temp_y[i] = temp_outy[i] - t2[i];

     ct[i] = 1/(1+exp(-temp_outy[i]));
     allct[i] += ct[i];
}


 //计算输出层各单元的校正误差

 for(i = 0; i < q; i++)
   {
```

```
        dt[i] = (y0[i] - ct[i]) * (1 - ct[i]);
        alldt[i] += dt[i];
    }

    //计算中间层各单元的校正误差
    for(i = 0; i < p; i++)

    {
        temp_xy = 0;
        for(j = 0 ; j < q; j++)
        {
            temp_xy = temp_xy + dt[j] * v[i][j];
        }
        ej[i] = temp_xy * bj[i] * (1 - bj[i]);
        allej[i] += ej[i];
    }


    cp++;

    }
    cnt++;
    er = 0;

    //计算误差值
       for(i = 0; i < q; i++)
       {
          er = er + (y0[i] - ct[i]) * (y0[i] - ct[i]);
       }

        for(i = 0; i < q; i++)
       {
          dt[q] = alldt[q] / 3;
          ct[q] = allct[q] / 3;
       }
       for(j    = 0; j < p; j++)
       {
          ej[p] = allej[p] / 3;
          bj[p] = allbj[p] /3;
    }

        //计算下一次学习时中间层和输出层之间的新连接权和新阈值
      for(i = 0; i < q; i++)
       {
          for(j = 0; j < p; j++)
          {
                v[j][i] = v[j][i] + a1 * dt[i] * bj[j];//中间层与输出层之间的新连接权

           }
           t2[i] = t2[i] + a1 * dt[i];//输出层的新阈值
    }

    //计算下一次学习时中间层和输出层之间的新连接权和新阈值
    for(i = 0; i < p; i++)
    {
          for(j = 0; j < 4; j++)
          {
```

```
                for(k = 0; k < 4; k++)
                {
                    loc = 4 * j + k;
                    w[loc][i] = w[loc][i] + b1 * ej[i] * x0[j][k];//输入层与中间层的新连接权
                }
            }
            t1[i] = t1[i] + b1 * ej[i];//中间层的新阈值
        }
    }while(er > emax && cnt<30000);

    if(er < emax)
    {
        printf("\nSuccessful , the results are\n");
        printf("1.输入层和中间层之间的连接权为：w[i , j]\n");//完成学习后输入层和中间层之间的
连接权
        for(i = 0; i < n; i++)
            for(j = 0; j < 4 ; j++)
                printf("w[%d , %d]: %1.4f\n" , i , j , w[i][j]);

        printf("\n2.中间层和输出层之间的连接权为：v[i , j]\n");//完成学习后中间层和输出层之间的
连接权
        for(i = 0; i < p ; i++)
        {
            for(j = 0; j < q ; j++)
            {
                printf("v[%d , %d]: %1.4f\n" , i , j , v[i][j] );
            }
        }

        printf("\n3.完成学习后中间层的新阈值为：t1(j):\n");//完成学习后中间层的新阈值
        for(j = 0 ; j < p ; j++)
        {
            printf("t1[%d]:%1.4f\n" , j , t1[j]);
        }

        printf("\n4.完成学习后输出层的新阈值为：t2(j):\n");//完成学习后输出层的新阈值
        for(j = 0 ; j < q ; j++)
        {
            printf("t1[%d]:%1.4f\n" , j , t2[j]);
        }

        printf("\n 累积误差校正算法：");
        printf("\n 误差值为  error = %1.8f\n" , er);//误差值
        printf("完成学习的次数为  Counter = %d\n" , cnt+1);//完成学习的次数

    }
    else
    {
        printf("Failed\n");
    }

}
```

北京师范大学珠海分校本科生毕业论文评定表

信息技术 **学院（系）** 2009 **级** 软件工程 **专业** **姓名** 谢思路 **学号** 0901030113

题目	一个基于 BP 网络的字符识别器的设计和实现
指导教师意见	（论文评语及给出初步成绩） 指导教师签章　　201　年　月　日
答辩小组意见	（论文、答辩评语，成绩及是否推荐院级优秀论文） 成绩（百分制）_____ （四级分制）_____； 推荐申报院级优秀论文投票：赞成____人，反对____人，弃权____人。 组长签章　　201　年　月　日
院级评优意见	（是否同意评为院级优秀论文及推荐校级优秀论文） 推荐申报院级优秀论文投票：赞成____人，反对_____人，弃权_____人。 教学院长（主任）签章　　院系章　　201　年　月　日
校级评优意见	 教务处长（签章）　　201　年　月　日

注：（1）此表一式四份，一份存入学校档案，一份装入学生档案、一份装入论文封底，一份学院存档、长期保存。填写时务必字迹工整，签章俱全。（2）*如系两位教师合作指导，应同时签名。