河 北 工 业 大 学

毕 业 论 文

作　　者：闫帅帅　　学　号：120087

学　　院：理学院

专　　业：信息与计算科学

题　　目：基于神经网络的手写数字识别算法与实现

指导者：饶道娟　　讲师

(姓 名)　　(专业技术职务)

评阅者：

(姓 名)　　(专业技术职务)

2016 年 6 月 2 日

毕业论文中文摘要

基于神经网络的手写数字识别算法与实现

摘要：

手写体数字识别是字符识别的一个分支，具有非常广泛的应用。在手写体数字识别领域，往往需要比普通的文字识别更高的识别率，这是因为数字识别一般应用在金融等要求十分严谨的领域，因此具有较高误识率的手写体识别算法并不能实际应用。BP 神经网络是最基本的一种前馈神经网络，它由简单的神经元组成，可以逼近任意复杂的非线性连续函数，可以用于解决分类和拟合问题，具有非常广泛的应用。

本文利用 MATLAB 神经网络工具箱，实现了基于 BP 神经网络的手写体数字识别算法，详细研究了数据预处理方法、不同的训练方法、不同的参数设置对识别率的影响，找出了一种具有很高识别率的 BP 神经网络结构，在 MINIST 手写数字库上进行了实验，在 10000 个测试样本上，识别率达到 96.6%。

关键词： BP 神经网络，模式识别，手写体数字识别，PCA 降维，白化

毕业论文外文摘要

Title Handwritten Digit Recognition based on Neural Network: Algorithm and its Implementation

Abstract

Handwritten numeral recognition is a branch of character recognition which has very extensive application. The field of handwritten numeral recognition often require higher recognition rate than normal character recognition, because the digital identification of general application in the field of financial requirement is very strict. So, the algorithm that has a higher rate of mis-recognition cannot being applied. BP neural network is one of the most basic of feedforward neural networks, it is composed of simple neurons, can approximate any complex nonlinear continuous function, and can be used to solve the problem of classification and fitting. So it has very extensive application.

This paper, by using the MATLAB neural network toolbox, realized the handwritten digit recognition based on BP neural network algorithm.And data preprocessing method is studied in detail, the different training methods, the influence of different parameters set on recognition rate, find out a high recognition rate of BP neural network structure, on the MINIST handwritten digital library, a experiment is carried out on the 10000 test samples, the recognition rate is 96.6%.

Keywords： BP neural network,Pattern recognition, Handwritten numeral recognition, PCA reduce dimention, Whitening

目　　录

1 引言

1.1 手写体数字识别的研究背景及意义[15]

手写体数字识别在生活中的应用领域十分广阔，而且随着全球信息化进程的高速发展，需要将大量的信息进行数字化[1]，因此手写体数字识别的需求也随之增加，各种新出现的应用大幅增多[2]。

（一）应用于数据统计范畴：在范围很大的数据统计中，比如工业年鉴、人口的普查和各种用到手写体数字的调研等，需要输入计算机大量的数据[3]，在完全是手工输入的情况下，需要耗费极大的人力物力，不仅费时费力，而且人类容易出现疲劳后出错量大的问题，而自动的手写体数字识别就很好地规避了这一问题。通过在这类问题中采用手写体识别技术已经是一种趋势[4]。在这类问题中，数据的录入是集中组织的，因此可以通过设计出来专门的表格和对书写加以限制来提高机器的自动识别率，因此可以很有效地使用自动的手写体数字识别[3]。

（二）应用于快递行业：随着淘宝、京东以及亚马逊等电商的兴起，全世界范围内的生活商品成交量有大半都转到了线上。例如，在双十一期间，仅淘宝的成交额一天之内就达到了 600 亿元，这些巨量的商品绝大部分是实体商品，需要快递行业进行邮寄，但大量的快递分拣工作量十分巨大，而且也容易导致漏分错分的情况，因此研究快递的自动分拣系统尤为重要，快递最重要的就是时效，这对整个行业来说十分重要，通过将快递和信件上的区号等进行数字识别，实现自动分拣，将是对分拣效率的巨大提高。

（三）应用于财政和金融领域：随着我国经融领域的高速发展，票据业务也在不断发展壮大，各种票据如支票、报款单付款单等数量大幅增加，手工将票据内容录入电脑的工作量已经大到无法持续的地步，因此迫切需要能够自动完成识别票据手写体识别数字的系统，减轻工作人员的工作量，从而将大量的时间省出来应用到更加需要的地方。但是这个方面的应用难度更大，首先是日常处理的信息十分庞杂，需要设计出大量的不同样式的表格，系统需要同时支持匹配大量的表格类型，再者就是识别率的问题，识别率需要达到十分高的水平，否则对金融业来说根本无法有效普及。

1.2 发展历史及研究现状

20 世纪 50 年代的美国首先出现了光学字符识别，十多年以后，蓝色巨人 IBM 生产出了第一款光学阅读机，其能够将标准的英文字母准确识别出来，这种阅读器催生出了一些应用，例如邮政业实现了利用光学字符识别的自动化信件分拣机等。20 世纪 60 年代，日本开始引入商业性的光学字符识别系统[5]，1968 年，日立公司已经能够制造他们的字符识别系统用于数字和字母印刷字符。随着手写体识别的发展，20 世纪七十年代 IBM 推出了存款处理系统。其能够精确地识别手写体支票金额数。

二十世纪八十年代，半导体技术发生了巨大的飞跃，出现了如 CCD 图像传感器等新技术，使光学字符识别系统变得更小更适合桌面办公，还有越来越便宜的存储器等，能够使扫描的图像被整页的存储到存储器中可以用来进行进一步的处理，从而可以应用到更普遍的领域。

成熟的手写数字识别系统出现在 20 世纪 80 年代，这是一个办公自动化的繁荣时代。十多年后，国际模式识别协会在九十年代开始举行会议。这些会议深入讨论了当时最新的研究方法。诸如人工神经网络；SVM；多项式函数分类；改进二次判别函数分类；分类组合；信息集成和词汇字符串识别等。这些技术仍然在现今发挥着至关重要的作用。

1.3 数字识别的方法综述

在印刷体识别上，机器可以认出几乎所有的数字，是因为印刷体字符十分工整，可以直接通过匹配算法得到结果，但是手写体的写法千变万化，每个地域的人们写法又各具特色，同一个数字的写法就又有很多种，因此想要识别手写体要做到很好地识别并不简单。

目前的识别算法有以下几种：

- 模板匹配法：对每个定义的模式分类树立一个标准的模板，和所有地将要被识别的字符进行比较。由于手写体的数字千奇百怪，具有各种样式，同一个人在不同时期写出来的数字都是不同的，而且不同地区也有很明显的差别，因此显然这种方法不适合手写体数字的识别。
- 统计决策法：抽取数字字符的特征进行概率学上的比较，概率最大的就是识别出的结果[6]。这是基于数学计算的方法，需要提取出数字的特征，但不易识别精细结构，因此也是不合适的。

- 句法结构法：需要抽取字符基元，不适于干扰能力较强的手写体识别。
- 模糊判别法：是一种由模糊数学发展出来的算法。它是根据择近原则进行分类。由于手写体数字建立隶属度十分困难，因此也不适合此方法。
- 逻辑推理法：是基于人工智能的推理方法，但对于手写数字难以奏效。
- 神经网络法：此算法是基于神经网络不需要人为干预，可以自己学习。通过大量的训练，能够把手写体隐含的数字特征提取出来，然后进行相应的识别。由于不需要人工给出手写数字的各种特征，因而可以针对不同的样本进行训练，从而使训练具有针对性，因此此种方法适用手写体地数字。

1.4 本文内容安排

本文研究的是基于神经网络的手写体数字识别算法和实现。我用的是 BP 神经网络。并在 MATLAB 上实现我做出的算法。

本文共有四章，第一章是引言部分，介绍了手写体数字识别在日常生活中的广阔应用范围，短暂地发展历史和对手写体识别的方法概述，以及我所做的工作。

第二章简要介绍了 BP 神经网络的发展历史、原理、特点以及应用。

第三章介绍了 MATLAB 软件的神经网络工具箱。包括有：在 MATLAB 上使用命令进行的 BP 神经网络的创建；训练创建好的神经网络；对训练好的网络进行仿真。

第四章介绍了我做的工作，主要是完成手写体数字识别算法的基本思路，完成方法以及最终的实现的识别率问题。我还分析了不同的参数设置以及对前期数据的一些操作比如归一化等对最终的效果地影响。

2 BP 神经网络

2.1 概述

神经网络是上世纪 40 年代由外国地科学家根据人类脑神经的工作原理，仿照人类大脑的工作方式创立的一种算法。这种算法能够实现许多建立数学模型算法无法解决的问题。其后，反向传播神经网络在 20 世纪 80 年代被提出来。它是一种按照误差反向传播算法训练的多层前馈网络，是目前应用最广泛的神经网络模型[7]。反向传播，顾名思义，就是每一次的传播的误差反过来向前传递的神经网络。它的神经元传递函

数是类似英文字母 S 的可导函数，能够输出 0 到 1 之间的连续数，并且能够不用人为干预地学习以及存储大量的 IO 映射关系，而且不需要给出解释这种映射的数学方程，因此应用空间十分广阔。权值调整用反向传播算法 Back Propaganda，故名 BP[8]。

2.2 原理

如图 2-1 所示，是一个单个 BP 神经网络单元的结构示意图。它具有 R 个输入。每一个输入输出都有一个适当的权值 w 和下一层相连。网络输出可以表示为：

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

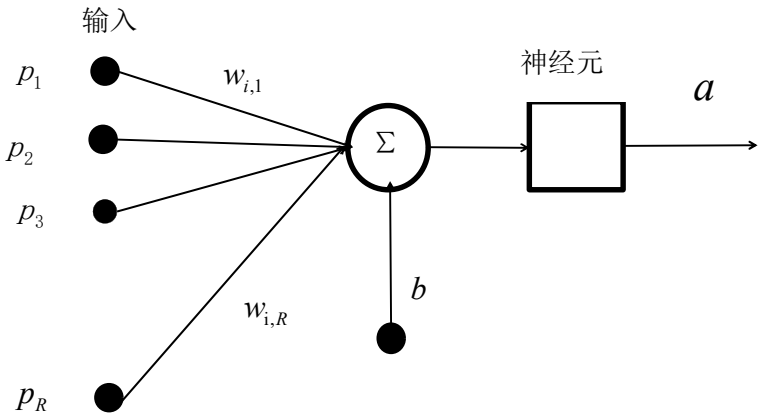


图 2-1 单个神经元信息传递过程

上式中，就是表示输入和输出关系的传递函数。p 就是 R 维输入向量，b 就是偏置（或者叫阈值），$w_{i,j}$ 是第 i 层的每个神经元突触的权值，a 为神经元输出。

BP 神经网络中的隐含层神经元传递函数是无限可微的 sig-moid 函数或者是线性的函数 purelin。sigmoid 输出的数值可以包含负数值。logsig 函数的输出只能为正数，tansig 可以输出负数。如图 2-2 所示为 logsig 的函数图像，可以看出，它的值域范围是 0 到 1。而 tansig 的输出值域是-1 到 1。可以看出他们是相似的。

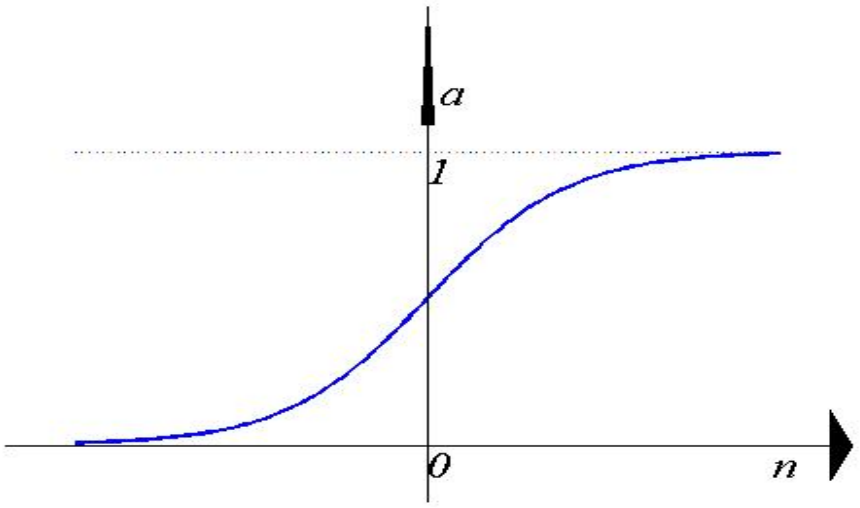


图 2-2 sigmoid 函数 logsig

前馈型神经网络通常有很多个隐含层。隐含层中的传递函数都是非线性的无限可导的函数 logsig 和 tansig。输出层地神经元使用线性地传递函数 purelin。

典型的多层神经网络具有三层。包括输入层、输出层和隐含层。其各层的作用都是不同的、呈现出一种递进的关系。具体关系如下：

- 输入层：这一层的作用是接受输入地数据流，并且能够将数据送给中间一些神经单元[9]；
- 隐含层:中间层是进行信息的映射形成的区域，输入层的信息传递到中间层时，进过相应权值的作用，根据要求，中间层可以设置成一层，也可以超过一层[9]；最后一个隐层传递到输出层各神经元的信息，经过多次进行相关的过程后，就能够完成一次正方向地传播学习步骤；
- 输出层:神经网络的输出层是用来输出信息的最终成果的；

BP 神经网络的学习过程，如图 2-3 所示。

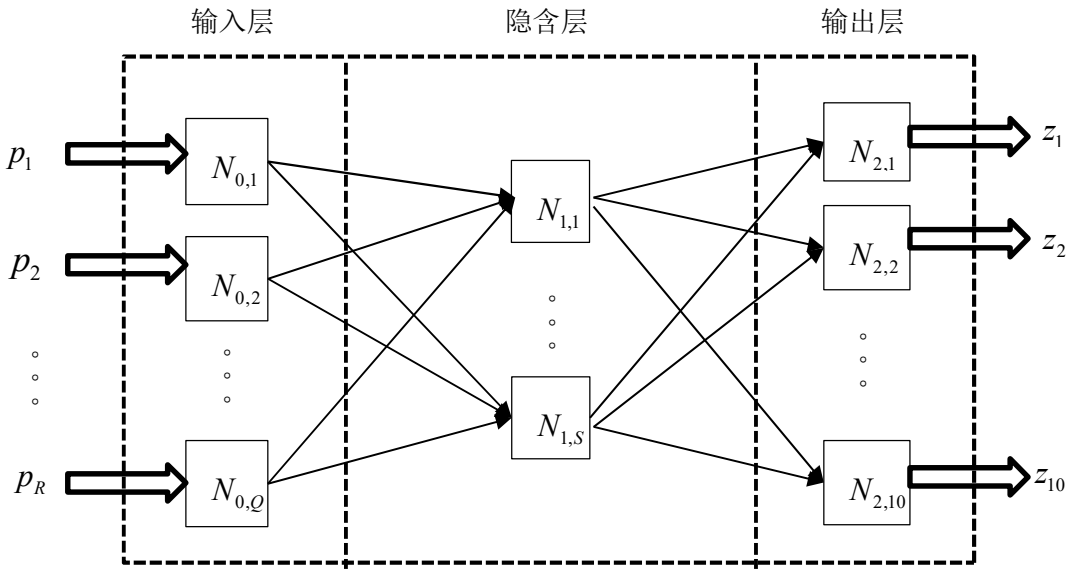


图 2-3 BP 神经网络学习过程

如图 2-3 所示，BP 神经网络的输入样本为{(P,T)|P 为输入向量，T 为对应于 P 的输出向量}，输入向量为 P,隐含层输出向量为 $N_{1,i}$,输出层输出向量为 $N_{2.i}$，期望输出为 T，实际输出为 Z_i，因此误差就是 Z-T，均方误差 MSE 就是 Z 和 T 的均方误差[9]。

BP 神经网络进行训练时，首先由输入端向神经网络的输入节点赋值，经过隐含层的矩阵乘法操作，再经由输出节点输出 Z，当实际输出与期望输出不符时，进入误差的反向传播阶段[9]。比较向外传播的向量 Z 和期望地传出向量 T 的误差，误差通过输出层，按误差梯度下降的方式逐级修正每一层地权值。再向隐层、输入层一层一层进行反传。循环往复地信息正向传播和误差的反向传播过程。每一次的权值都在不断地进行调整，这就是神经网络的训练过程。这个过程一直进行到 BP 神经网络的均方差降低到足够使用的地步。亦或预先设定的训练次数为止。

训练完毕后，神经网络内各层的权值就已经固定，就可以输入测试数据进行测试了。

2.3 特点

信息的分布式存储：人类大脑地神经存储信息的特点是利用突触和突触的链接来调整信息。即通过大量的神经元之间的连接强度的分布来储存大量的信息[10]，而 BP 神经网络利用这一特点，通过大量的连接权值来储存信息。

巨量的并行处理。人脑神经元之间传递脉冲信号的速度只有千分之一秒到百分之一秒，远远低于 CPU 或者 GPU 的工作速度（低于微秒级别）。但是却可以做出快速的判断和决策。这是由于人脑是一个大规模并行加串行组合的处理系统，人脑能在 100u=vfdjbnks,lqdfbvgfbgvcc 的时间内做出极其复杂的判断，这是计算机所不能比拟的。BP 神经网络的基本结构十分类似于生物大脑，可以进行并行处理，能够增强网络性能。

强大的容错性。根据某些研究发现，人脑的神经系统部分损伤时并不影响整体功能，同样，BP 神经网络也具有这种特性，大量的权值连接可以使一些较小的误差不影响结果的正确性，少量神经元的损坏不会导正整体无法工作，它可以自动修正误差，使容错性大大加强。

具有自学习、自组织、自适应能力。神经网络具有自适应与自组织能力。在训练过程中能够在没有人为干预下学习相关的规则。它是通过不同链接之间地权值变化，不断学习完善功能。

2.4 应用

BP 网络主要用于以下四个方面：函数逼近、模式识别、分类以及数据压缩等[11]。

- 函数逼近：用输入和输出逼近一个函数[12]。
- 模式识别：建立输出与输入地联系[13]。
- 分类：按照输入向量定义地方式进行合理分类。
- 数据压缩：减少输出向量维数。即减小数据大小，可以使传输或存储更省力。

3 MATLAB 神经网络工具箱

3.1 网络创建

3.1.1 BP 网络的简单创建

在 MATLAB 中创建一个 BP 神经网络，可以使用 MATLAB 的图形界面。在本论文中全部使用函数创建。创建 BP 神经网络的函数为：

net=newff(PR,[S1 S2...SN],{TF1 TF2...TFN},BTF,BLF,PF)

其参数解释如下：

PR:由 R 维的输入样本的值域组成的 R*2 维地矩阵[14]。

Si:第 i 层的神经元数量。

Ti:第 i 层的神经元传递函数。tansig()和 logsig()两个传输函数

BTF:训练函数名称，默认是 trainlm 函数

BLF:权值/阈值函数，默认是 learngdm 函数，详细情况见表 1。

PF:性能函数，默认是 MSE 函数

这个命令不但建立了 BP 神经网络，而且初始化了权值 w 和偏置 b，就可以进行下一步的操作了。

3.1.2 BP 网络的重新初始化

有时候，直接使用默认的初始化并不够，我们可能需要修改一些参数，或者初始化成上一次初始化的权值。在这种情况中，直接使用默认的会导致准确度不高，没有可比性等，因此需要重新进行初始化。可以利用 init 函数对网络进行初始化，例如：

```
net.layers{1}.initFcn = 'initwb';
net.inputWeights{1,1}.initFcn = 'rands';
net.biases{1,1}.initFcn = 'rands';
net.biases{2,1}.initFcn = 'rands';
net = init(net);
```

net=init(net,i) %net 是建立的神经网络，i 是层索引，返回值是更新后的 i 层的权值和 b。

也可以采用下面的方式对某一层的权值进行初始化：

```
net.IW{i}=[0.2;0.3;0.4;0.5];    %设置第 i 层权值
```

例如：

```
net.iW{1,1}=reshape(w1,hiddennum,inputnum);
net.lW{2,1}=reshape(w2,outputnum,hiddennum);
net.b{1}=reshape(B1,hiddennum,1);
net.b{2}=reshape(B2,outputnum,1);
```

初始化函数默认已经被 newff 所调用了，init 不需要单独的调用。但是有时候我们需要重新初始化创建的 BP 神经网络的权重 w 和偏置 b 或者称为自定义的初始化。例如，使用 newff 命令新建 BP 神经网络时，它默认是 initnw 来给第一层赋值。假设要用随机数进行初始化第一层的权值和偏置值，应该使用：

```
net.layers{1}.initFcn = 'initwb';
net.inputWeights{1,1}.initFcn = 'rands';
net.biases{1,1}.initFcn = 'rands';
net.biases{2,1}.initFcn = 'rands';
net = init(net);
```

net.initFcn 用来决定整个网络的初始化函数。前馈网络的默认值为 initlay，可以对每一层进行定制初始化[15]。

表 3-1 BP 网络的训练函数

训练方法	训练函数 PF
梯度下降法	traingd
有动里的梯度下降法	traingdm
自适应 lr 梯度下降法	traingda
自适应 lr 动里梯度下降法	traingdx
弹性梯度下降法	trainrp
Fletcher-Reeves 共轭梯度法	traincgf
Ploak-Ribiere 共轭梯度法	traincgp
Powell-Beale 共轭梯度法	traincgb
里化共轭梯度法	trainscg
拟牛顿算法	trainbfg
一步正割算法	trainoss
Levenberg-Marquardt	trainlm

3.2 网络训练

3.2.1 网络训练函数

在创建和初始化神经网络后，需要对神经网络进行训练。训练 BP 神经网络的命令是 train。它的使用格式如下：

[net,tr,Y,E,Pf,Af]=train(net,P,T,Pi,Ai,VV,TV)

net：已创建的网络

P：输入矩阵

T：输出矩阵

由于其它参数目前用不上，故省略。

3.2.2 网络训练参数

BP 神经网络的训练一般采用默认参数就可以。但是有时候需要一些设置参数，用来进行特定的训练和调试,这就需要修改训练参数了,常用的训练参数有以下几种：

net.trainParam.epochs－最大训练次数；

net.trainParam.goal－ 训练收敛误差；

net.trainParam.show－显示间隔；大小会对结果有影响

上述参数在所有神经网络训练中都要用到。但是如果使用的训练函数是 trainlm，还需要设置如下的一些参数：

net.trainParam.mu－Levenberg－Marquart 优化算法中的

net.trainParam.mu_dec－ 的缩减因子；

net.trainParam.mu_inc－ 的增大因子；

net.trainParam.mu_max－ 的最大值；

net.trainParam.min_grad－性能函数的最小梯度；

如表 3-2 所示，最右侧的训练函数是对应的训练函数，前面就是相应训练函数对应的训练参数，只有修改对应函数包含的参数，才会使用修改后的参数。

表 3-2 BP 网络的训练参数

训练参数	参数介绍	训练函数
net.trainParam.epochs	最大训练次数（缺省为 10）	traingd traingdm traingda traingdx trainrp traincgf traincgp traincgb trainscg trainbfg trainoss trainlm
net.trainParam.goal	训练要求精度（缺省为 0）	
net.trainParam.lr	学习率（缺省为 0.01）	
net.trainParam.max_fail	最大失败次数（缺省为 5）	
net.trainParam.min_grad	最小梯度要求（缺省为 1e-10）	
net.trainParam.show	显示训练迭代过程（NaN 表示不显示，缺省为 25）	
net.trainParam.time	最大训练时间（缺省为 inf）	
net.trainParam.mc	动量因子（缺省 0.9）	traingdm traingdx
net.trainParam.lr_inc	学习率 lr 增长比（缺省为 1.05）	traingda traingdx
net.trainParam.lr_dec	学习率 lr 下降比（缺省为 0.7）	
net.trainParam.max_perf_inc	表现函数增加最大比（缺省为 1.04）	
net.trainParam.delt_inc	权值变化增加量（缺省为 1.2）	trainrp
net.trainParam.delt_dec	权值变化减小量（缺省为 0.5）	
net.trainParam.delt0	初始权值变化（缺省为 0.07）	
net.trainParam.deltamax	权值变化最大值（缺省为 50.0）	
net.trainParam.searchFcn	一维线性搜索方法（缺省为 srchcha）	traincgf traincgp traincgb trainbfg trainoss
net.trainParam.sigma	因为二次求导对权值调整的影响参数（缺省值 5.0e-5）	trainscg
net.trainParam.lambda	Hessian 矩阵不确定性调节参数（缺省为 5.0e-7）	
net.trainParam.men_reduc	控制计算机内存/速度的参量，内存较大设为 1，否则设为 2（缺省为 1）	trainlm
net.trainParam.mu	μ 的初始值（缺省为 0.001）	
net.trainParam.mu_dec	μ 的减小率（缺省为 0.1）	
net.trainParam.mu_inc	μ 的增长率（缺省为 10）	
net.trainParam.mu_max	μ 的最大值（缺省为 1e10）	

3.3 网络仿真

上一节已经训练完创建好的神经网络。这一步就是需要做一些仿真操作，来验证训练得到的网络是否符合要求。仿真函数 sim 通过对预定的输入进行仿真，和目标结果进行比对，验证训练是否合适。仿真函数调用格式如下：

[Y,Pf,Af,E,perf]=sim(net,P,Pi,Ai,T)

net:已经训练过的神经网络；

P:将要仿真的元数据；

Y:仿真后的到的仿真结果，用于比较或者应用；

其余参数目前并不会使用到，因此不做解释。

4 基于 BP 神经网络的手写体数字识别

4.1 基本思路

本章介绍论文完成的主要工作。首先本文确定使用 BP 神经网络来完成对 MNIST 数据的训练和识别。下一步就是需要确定一个最佳的 BP 神经网络结构，找出每个训练参数对训练速度、收敛情况和识别率的影响。

4.1.1 实验所用的数据集

论文采用 MNIST 数据集进行手写数字识别实验。该数据集共包含 70000 幅图像，其中 60000 幅用于训练，其余的 10000 幅图像用于对算法进行测试。每个 MNIST 图像是一个单一的手写数字字符的灰度图像,图像尺寸为 28x28 像素。

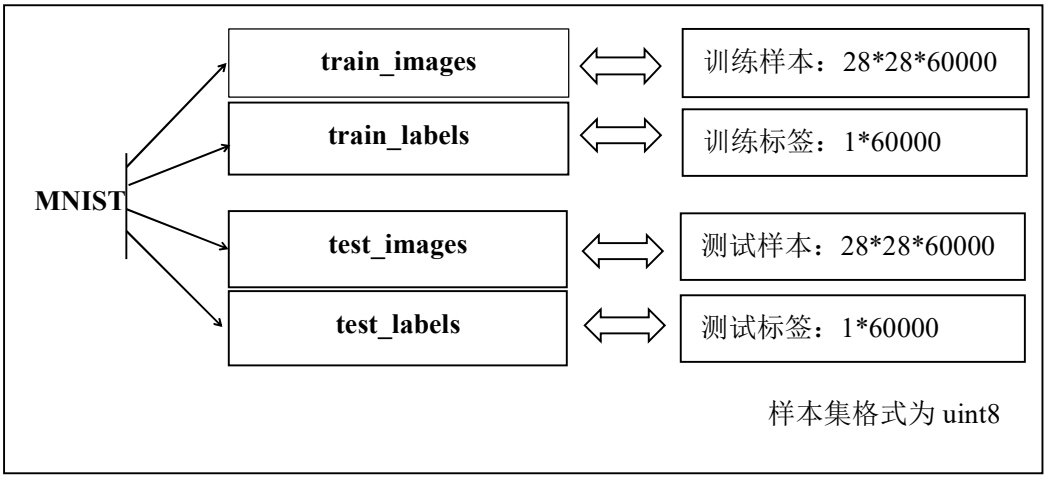


图 4-1 MNIST 数据集结构

MNIST 数据集调用格式为：mnist.train_labels。在调用数据的时候，将数据集文件 mnistAll.mat 直接放在 MATLAB 的当前工作目录下。在命令窗口中键入 load mnistAll 即可读取相关的数据，以供使用。MNIST 数据集中的样本均为图像。其在 MATLAB 中显示为 28*28*60000（训练数据）或 28*28*10000（测试数据）的三维矩阵。而神经网络的输入最高只能是二维矩阵，因此需要将三维的样本矩阵转化为神经网络

可用的二维矩阵。在此本文用了 MATLAB 中自带的 RESHAPE 函数将三维原始矩阵转化为二维矩阵，将每个样本的 28*28 的图像矩阵拉直成一行即 1*784，然后 60000 个样本组成一个 60000*784 的二维样本矩阵。

4.1.2 输出维数的确定

由于要识别的全为数字，在一共 70000 个样本数据中，每个样本都是独立的。亦即 1 和 8 在识别的状态下不存在大小关系，1 并不比 8 大，也不比 8 小。即，识别出每一个样本的概率是基本一样的，而且每一个样本不能被正确识别出来的概率也是一样的。因此，BP 神经网络的输出如果直接是数值，将会带来如下问题。被识别的数字的识别率将会被它本身的数值所影响。因此，本文采取了 10 维输出向量的方式，输出的向量映射出每一个数字。例如数字 5，它所对应的向量就是第五维是 1 其余全为 0 的向量。采取这种做法，能够使每一个被识别数字的识别率都是等可能的，识别出的结果更具有代表性。

4.1.3 仿真结果融合

根据上一小节的方法，输出维数为 10 维，每一维最大值的位置就是仿真的结果。因此，不妨在上一节的基础上再采取一种技巧，来提高识别率。先后训练 10 个不同的神经网络，每次都是随机赋予的初始值，因此最终训练完成后的神经网络性能相似，但是权值不同，识别率也有细微差距。将每一次的识别出来的 10 维向量和其真实值相比较，就能得出当前训练的识别率。训练 10 次后，将这 10 次的初步仿真结果（即 10 维输出向量）直接简单相加，得到一个新的 10 维向量，这个新的 10 维向量具有 10 次训练结果的累加和，亦即概率和。根据概率论中的相关知识，可以预测的是，用这个新的 10 维向量进行识别比采用单独的 10 维向量识别率将会得到提升。

4.2 基于 BP 网络的数字识别算法

4.2.1 算法步骤

a）数据集的归一化和降维

b）建立 BP 神经网络

c）训练 BP 神经网络

d）数据仿真

e）重复 c、d 步骤，直至识别率稳定或者 MSE 不再出现明显的下降。

由于每次只能建立并训练一种特定输入节点数和隐层节点数以及输出节点数的

神经网络，因此对于不同输入节点数、不同隐层节点数以及不同的初始化参数都需要单独进行训练，提取出相关的特性后再进行实验。因此神经网络的训练需要大量的数据作为对照，分析并找出最佳的参数配置以及节点个数和最佳识别率的关系。

4.2.2 算法流程图

如图 4-2，为本文进行识别的算法流程图。

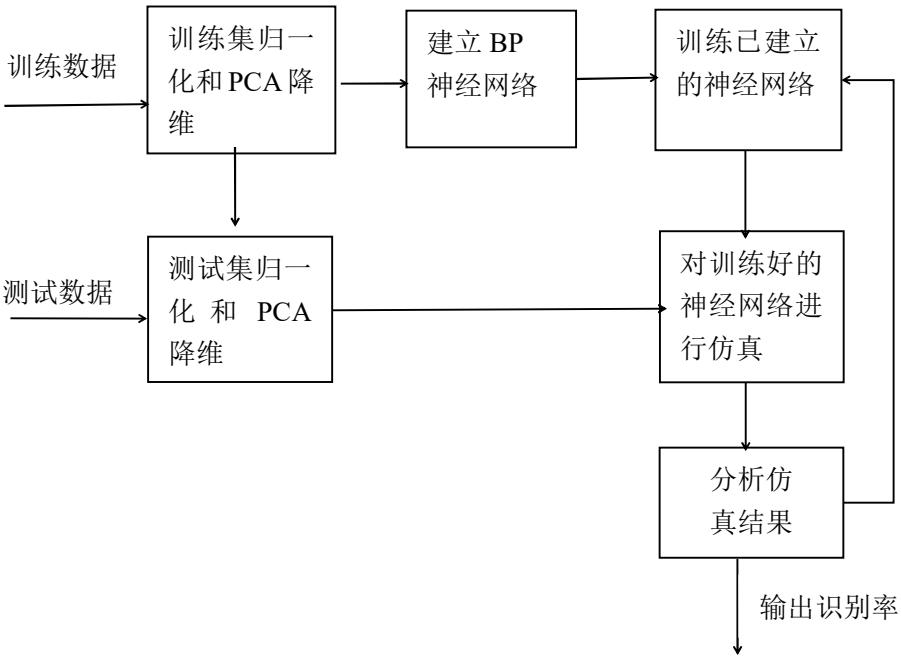


图 4-2 识别算法流程图

4.2.3 算法实现

将 mnistAll.mat 文件和程序放在同一个文件夹中，并设置为 MATLAB 的路径。代码 1 是对数据的预处理，包含对数据的归一化、PCA 主成分分析降维以及白化处理。这个操作后将会生成一个名为 data.mat 的 MATLAB 文件，里面存储着下一步需要用到的变量的值。

```
%%代码 1
%This part of program create data!
warning off all
clear; clc;
load mnistAll;
train_row=size(mnist.train_images,1);train_col=size(mnist.train_image
```

```
s,2);
train_page=size(mnist.train_images,3);
test_row=size(mnist.test_images,1);test_col=size(mnist.test_images,2)
;
test_page=size(mnist.test_images,3);

%reshape to 784*60000
train_images=double(reshape(mnist.train_images,train_row*train_col,tr
ain_page)');
test_images=double(reshape(mnist.test_images,test_row*test_col,test_p
age)');

%Normalizating..
disp('Normalizating...');
train_images=bsxfun(@times,train_images,1./sum(train_images,2));
test_images=bsxfun(@times,test_images,1./sum(test_images,2));

%PCA Reduce dimention...
disp('Reduce dimention...');[coef,~,latent]=princomp(train_images);
lat=cumsum(latent)./sum(latent);

train_labels=mnist.train_labels+1;test_labels=mnist.test_labels+1;
T=zeros(10,60000);%vertex
for i=1:60000
    T(train_labels(i),i)=1;
end
P = train_images*coef;
test_P = test_images*coef;

%whiten...
latent=latent';
P=bsxfun(@rdivide,P, sqrt(latent+1e-6));
test_P=bsxfun(@rdivide,test_P, sqrt(latent+1e-6));
%zhuanzhi
P = P'; test_P = test_P';test_T=test_labels;
save data lat P T test_P  test_T;
```

代码 2 是创建一个 BP 神经网络并修改训练参数。然后对其进行训练以及仿真。根据仿真结果，再做下一步的安排。如果训练几千次以后 MSE 不再出现明显下降或者识别率不会出现明显提升，就说明当前算法的识别率已经出现瓶颈。只有通过改变参数或者训练函数进行尝试优化，找出一个最优的参数组合，从而得到一个比较满意的

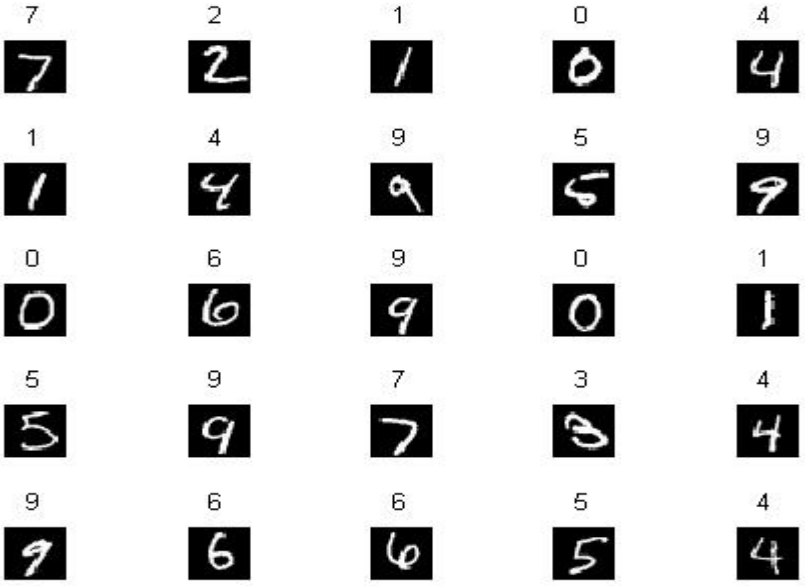
识别率。

```
%%代码 2
warning off all
clear;clc;
load data;
%a=find(lat>90);
%dim=1(1);%取贡献率达到 0.90 的前 dim 个数值
dim=50;
P=P(:,1:dim);
test_P=test_P(:,1:dim);
P = P'; test_P = test_P';
%%
disp('Create ANN: 50 inputs,50 knots,10 outputs purelin');
net=newff(minmax(P),[50,10],{'logsig','purelin'},'traingd');
disp('Training ANN...');
tic;net=train(net,P,T);toc

sim1=sim(net,P);[~ ,Y1] = max(sim1);
ratio1 = mean(Y1==double(train_labels'));
fprintf('Train ratio:  %0.4g \n',ratio1);

sim2=sim(net,test_P);[~ , Y2] = max(sim2);
ratio2 = mean(Y2==double(test_labels'));
fprintf('Train ratio:  %0.4g \n',ratio2);
```

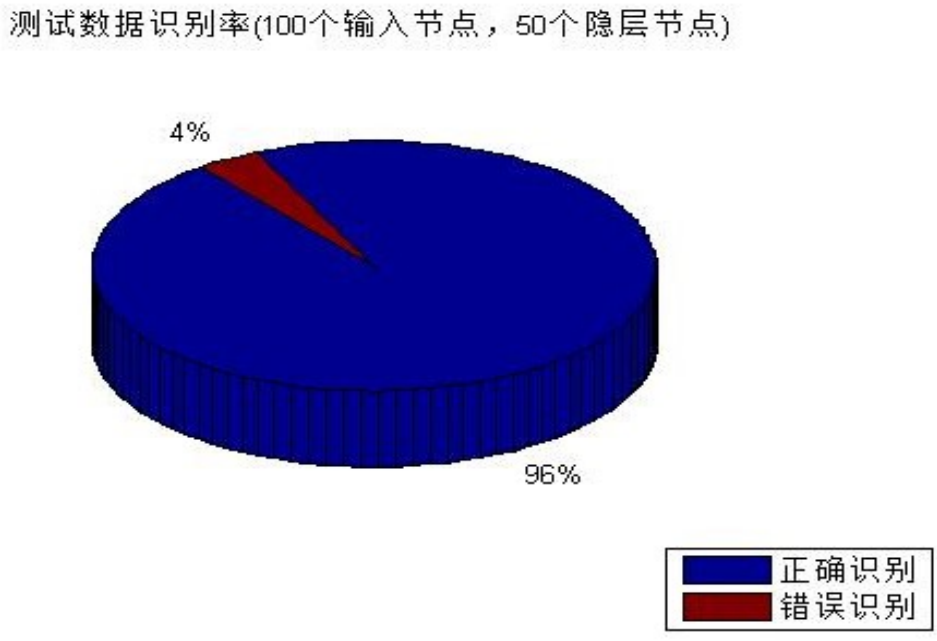
4.3 运行结果及分析

4.3.1 直接训练

经过训练，抽取其中的一个已经训练好的 BP 神经网络，其识别率如图 4-3(a)所示。可以看到，一共 25 个样本中，仅有 1 个数字出现了误识，将 4 识别成了 6。但是对于人眼来说，识别出这个 4 也不容易。因此识别效果基本达到了人眼的识别率。如图 4-3（b）所示，经过 11000 次训练，BP 神经网络的 MSE 值降低至 0.00756，以后基本没有明显的变化，识别率也已经稳定为 96.39%，因此，当前的经过直接训练的识别率为 96.39%。



(a) 随机选取的前 20 个数字识别效果



(b) 测试数据识别率

图 4-3

4.3.2 仿真结果融合

如图 4-3，绿色是十次的识别率，可以看到他们相差不大，将这 10 次训练结果仿真出来，进行融合。亦即把仿真结果简单相加，得到的一个新的 10 维向量，和真实数据相比较。其识别率如图 4-4.所示，为 96.6%。可以看出，识别率出现了并不是很大的提升，但已经较之前有了明显提高。

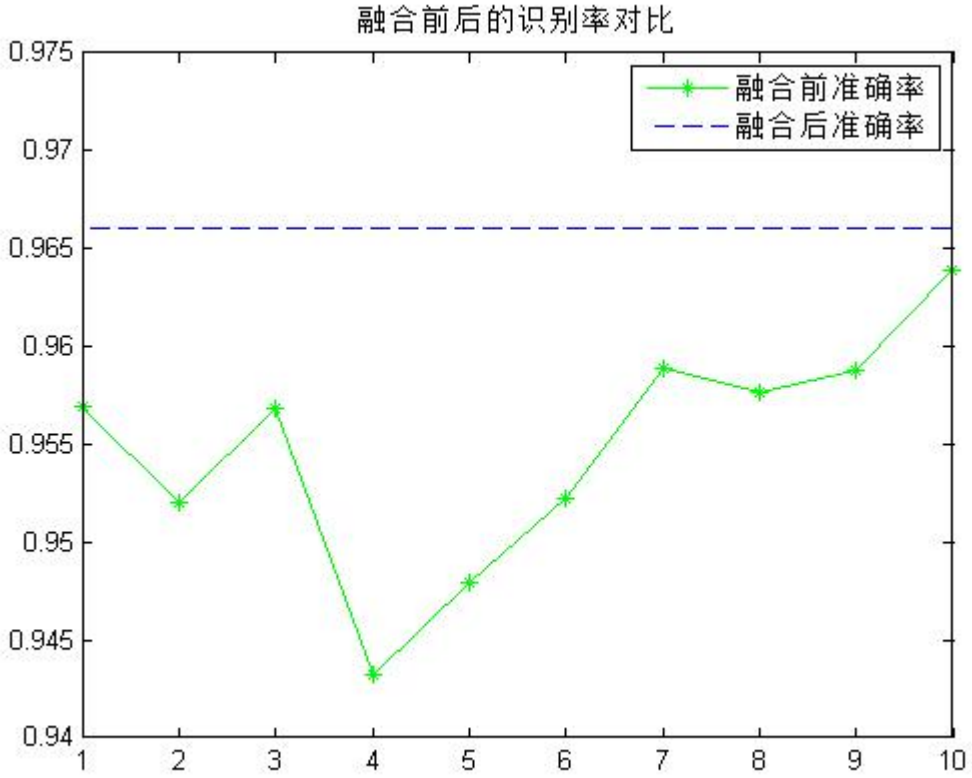


图 4-4 融合前后的识别率对比

4.4 数据预处理的影响

通常进行图像类识别都需要进行数据的降维等预处理操作，使数据具有更好的收敛性，加快训练速度。为了验证这个结论，本文选取了一种结构固定的BP神经网络。结构为单隐层，隐层节点数为30，激活函数为logsig，训练函数为带动量的变学习率梯度下降法，初始学习率设定为0.5，动量项为0.9，输出为10个节点数的神经网络。然后依次进行是否归一化、是否进行PCA降维以及是否进行白化的训练结果的分析。

4.4.1 归一化

第一次操作没有进行归一化也没有进行PCA降维和白化操作，发现训练在迭代320次时，梯度值已经达到极小为5.16e-6，但是均方误差（MSE）甚至刚刚达到0.09，距

离设定的0.01还有不小的距离。此过程耗时174s，最终的测试准确率只有11.72%。第二次实验中，进行了归一化的操作，发现训练在达到设定的1000次以后停止。此时的梯度值0.0018，距离1e-5的最小值还有距离，MSE为0.0712。说明经过归一化以后，程序是有可能到达最优解的。此过程耗时542s，最终的测试正确率为54.39%。相对于没有归一化的情况，已经有了质的提升。其对比效果如图4-5所示，注意，没有归一化时的时间短是因为提前达到了局部极小，并没有收敛。

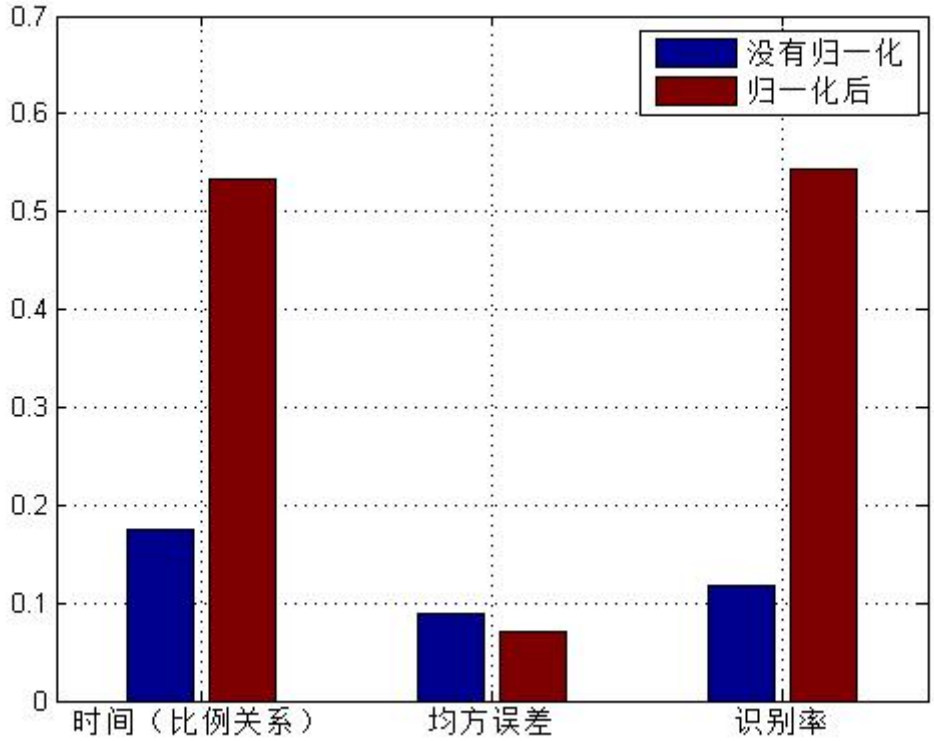


图4-5 归一化的作用影响对比

4.4.2 PCA降维

PCA降维是利用主成分分析法，能有效降低带训练数据的横向维度，降低数据量，却能大量保存数据所含的信息量，因此是一项能够大大提高训练效率的技术。作者使用的数据是经过归一化以后，分别测试它们的降维数。范围从不降维（784维）到降至10维。为了确保数据的公平性，我每组数据均测试三次再取平均值，分别分析降维带来的好处。

如图4-6所示，分析了从PCA主成分分析后没有降维（784维）到降至5维共8中情况所用的训练1000次的时间、达到的MSE值的大小以及训练后测试的准确度。通过图中数据可以看出来，经过PCA降维以后，维度越小，训练时间越短，相应训练得到的

均方误差更小。但是训练后得到的测试准确度却呈现了一种先增后降的趋势，在30到20维之间出现了峰值，因此可以判断，随着PCA降维的维度的减小，存在一个最佳的降维维度，使训练后测试准确度最高。经过大量的试验，以及经验公式、上网查询相关资料，作者将PCA降维的维度最终确定在和输入维度相近的数量，既能充分保证训练的速度，又能够保证训练后的测试精度。

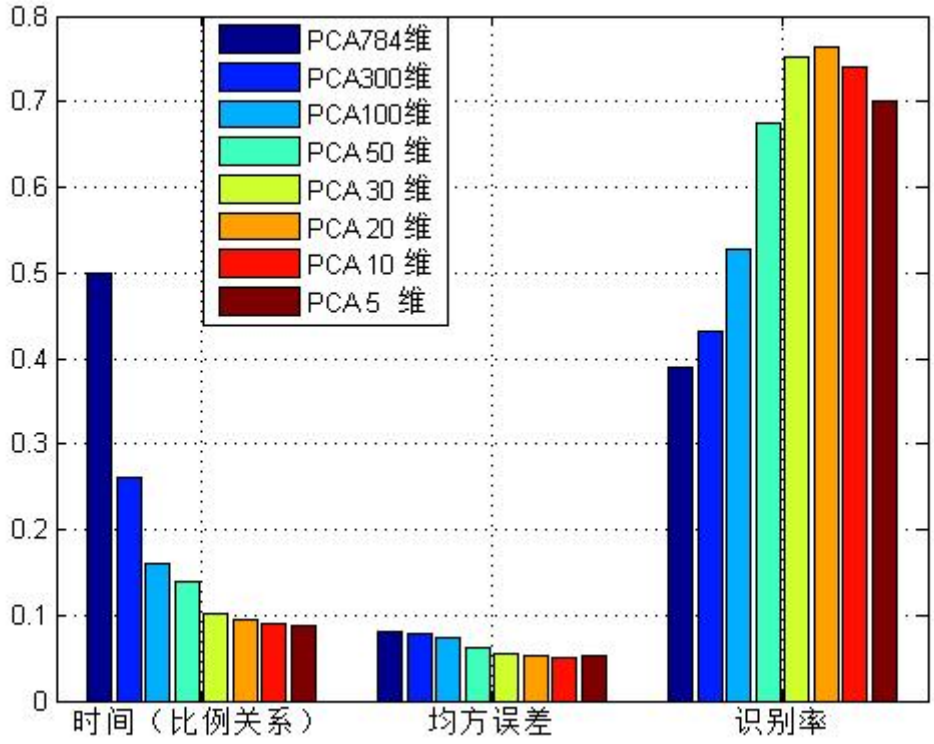


图4-6 PCA降维后的不同维度效果对比图

4.4.3 白化

白化就是将分散的数据的均值处理成0，方差处理成1。白化操作能够有效减小相邻元素的相互间联系，提高数据的可分析性。将输入数据经过归一化、PCA降维降至20维，隐层节点数为20，一组数据进行白化操作，另一组不进行白化操作。为了增加数据的可信度，在第一初始化完毕后，将初始化用的随机初始化权值保存起来，进行第二次测试时直接赋值，保证两次测试均采用相同的初始参数。

测试结果如图4-7所示，经过白化处理后，总的训练时间出现了少许增加，但是最终的训练结果的均方误差（MSE）出现了降低，而且测试准确率提高了10多个百分点。总的说来，白化操作能够提升神经网络的训练效率，能够在有限的训练次数内提升测试的准确率。

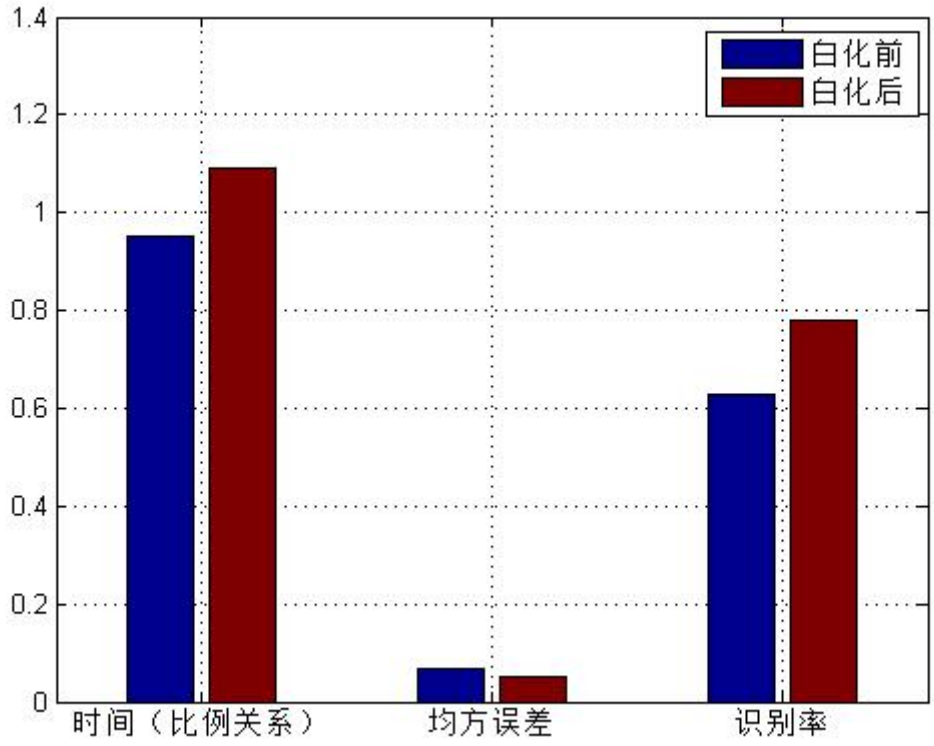


图4-7 白化前后对比

4.5 参数值的影响

作者选取了四个不同的参数，考察他们对创建的BP神经网络的影响。考察不同训练函数、不同动量项以及不同节点数的影响时，为了使整个比较具有可比性，在第一次初始化的时候把输入向量的默认随机初始权值记录在文件中，改变参数时，再将相同初始权值重新赋值，使它们具有相同的初始权值，能够使结果更具有代表性、更有信服力。

这几种类型依次是：不同的训练函数、不同的动量项、不同输入层节点个数和不同的隐层节点数。

4.5.1 训练函数

本文选取了不同的12训练函数，为了使结果更具有信服力，采用同一个方法训练三次取均值的方法，统计出训练时间、训练最终的MSE以及训练数据的仿真正确率和测试数据的仿真正确率。当前关注的是训练函数，因此输入样本设定为经过归一化、PCA降维后没有白化的数据，仅仅考虑不同的训练函数对最终效果的影响，输入向量为20维，隐层为一个隐层，节点数是20，输出为10个输出，分别用10维向量表示每一个数值。训练过程中发现，使用trainlm训练函数直接导致系统卡死，内存不足，训练失败，其余均能够正常训练，图4-8为训练1000次的结果和使用的时间。

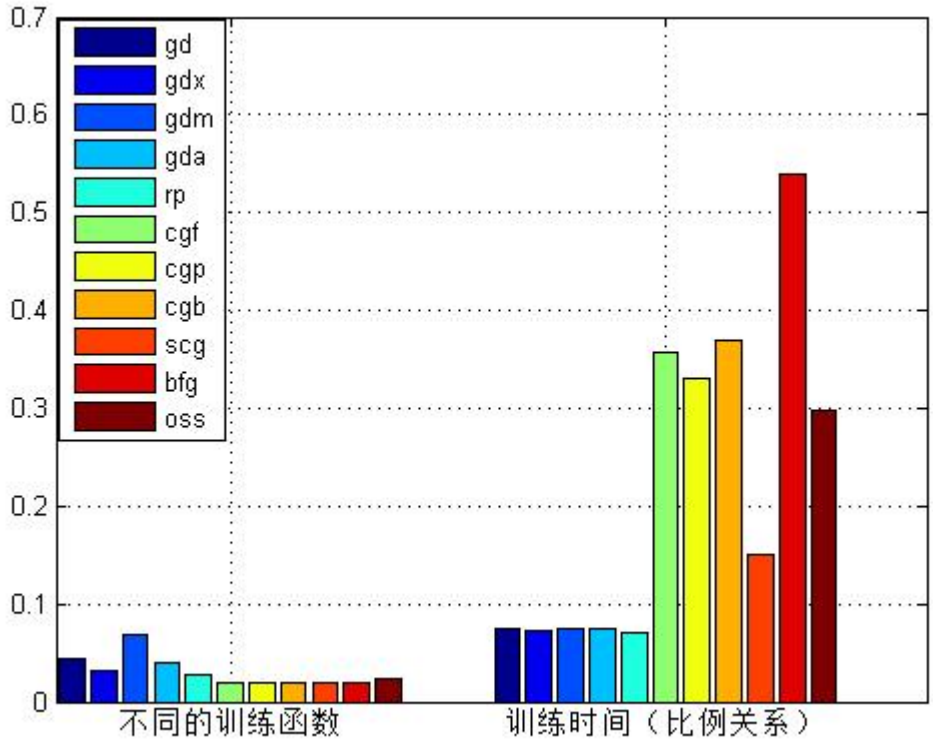


图4-8 不同训练函数的影响

由图可知，后几个训练函数的效果都不错，但是只有trainscg具有最好的训练时间效率，因此最优的训练函数就是trainscg。

4.5.2 动量项

本文选取了从0.3到0.9间隔为0.1的7个不同的动量项，选取输入节点数为50维，隐层节点数为50个，激活函数采用logsig，训练函数为带有动量项的变学习率的梯度下降算法，分别对每一个参数选取值训练了1000次，将得到的MSE记录下来，结果如图4-9.

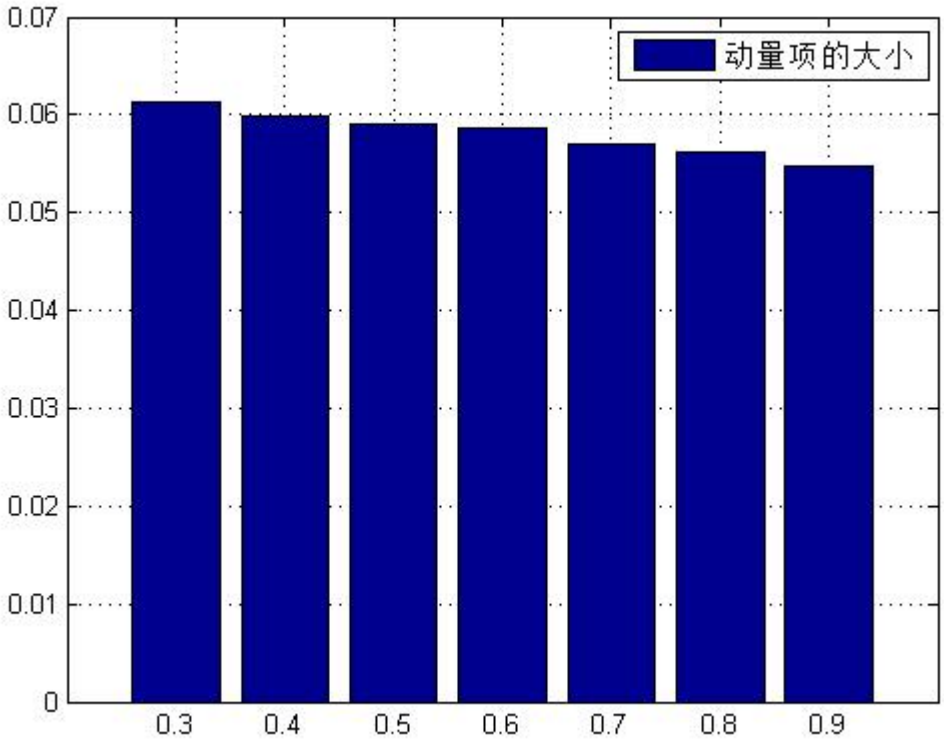


图4-9 动量项的影响

如图4-9，对比发现在其他参数均相同的情况下，随着动量项的增加，BP神经网络的训练得到的MSE是减小的。即动量项越大，训练效率越高。

4.5.3 输入节点个数

输入节点个数，在其余参数相同的情况下，输入节点个数是和PCA降维后的节点数完全相同的，PCA降维降至多少维，输入节点数就是多少，因此，此参数的结论与PCA降维部分完全相同，即随着输入节点数的减少，训练的效果也是先升后降的，会在某个数量上达到最优。

4.5.4 隐层节点数

由于时间关系，本文只测试了单隐含层的BP神经网络，验证单层隐含层中的神经元数量对BP神经网络训练的效果的影响。首先将输入层节点数固定，考察在同一个输入节点数下隐层神经元数量和神经网络的训练效果的关系。同样，采取每一个参数测试三次取平均值的做法，有利于减少误差，增加可信度。

如图4-10所示，选取输入层节点数为50个，分别对隐层节点数为10、20、30、40、50、60、70、80、90和100进行测试，发现随着隐含层节点数的增加，训练时间出现了线性增长，但是均方误差（MSE）却变化很小，从而训练得到的测试识别率在隐含层节点数超过30后也是很平稳，因此可以知道，训练的隐含层节点数在超过一定的数量以后，再增加其数量并不会导致训练后识别率的大幅提升。

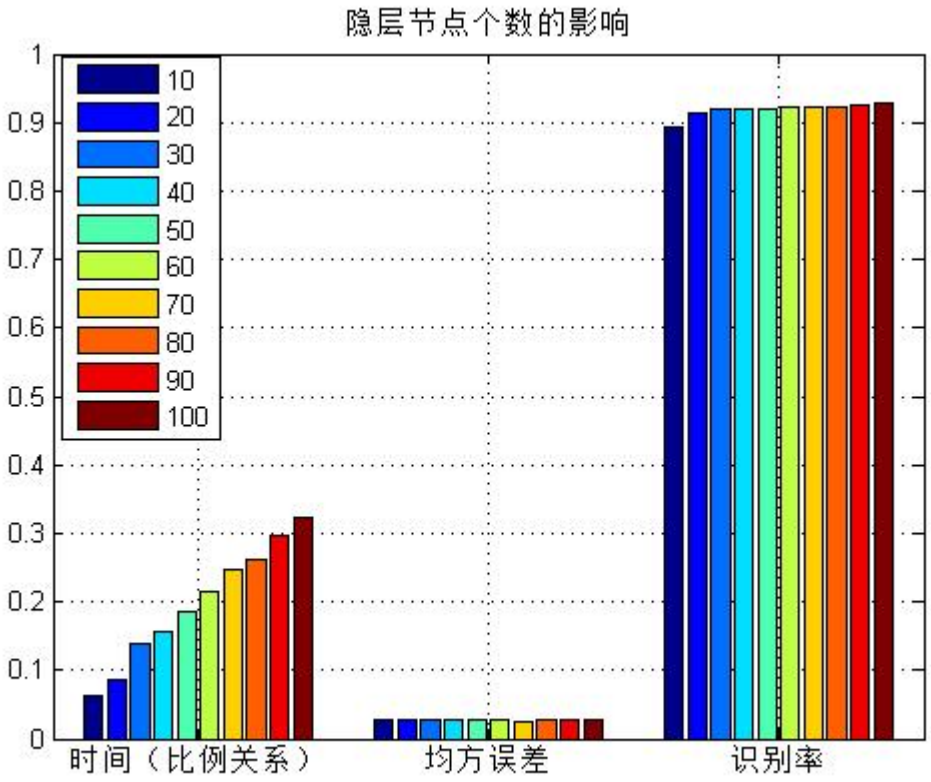


图4-10 隐层节点数的影响

结　论

手写体数字的识别是根据输入的手写体数字的图片判断其真实值的模式识别问题。随着信息化的高速发展，手写体数字识别已经成为研究神经网络和模式识别的一个重要的课题，在谷歌的数字化地球中就大量应用了这种技术。

BP 神经网络是一种可以以任意精度逼近非线性曲线的算法，而且训练前后也不需要对其作出数学解释，也不需要给出具体映射关系的数学方程。因此 BP 网络是一个十分易于上手的用于各种非线性逼近的神经网络工具。本文利用了 MATLAB 自带的神经网络工具箱，使用 BP 神经网络编写了手写体数字的识别算法，做到了接近 97% 的识别率，已经是可以实用的水平了。但还有一些方面需要做出改进，比如训练的时间问题，通常训练一个能够正常使用的神经网络需要几个小时甚至几天几个月的时间，因此会耗费大量的精力来做训练，希望在以后能够找到比较快速的训练方案，使神经网络真正做到实时训练实时使用的程度。

参 考 文 献

[1]王永乾.基于 BP 网络的手写体数字识别[D]. 山东：山东大学,2004.
[2]崔威威 田铮 赵伟.图像分割的自适应交互核图割模型[J]. 计算机工程与应用, 2013, 49(5) ：190-194
[3]李望晨.BP 神经网络改进及其在手写数字识别中的应用[D]. 哈尔滨：哈尔滨工业大学, 2006.
[4]王崇阳 傅迎华 陈玮.一种改进的稀疏表示的手写数字识别[J]. 信息技术, 2015, 0(2):5-8
[5]冯瑞.车牌识别技术的研究及实现[D]. 西安：西安工业大学, 2015.
[6]刘虎.脱机手写体阿拉伯字符识别关键技术研究[D]. 武汉：武汉理工大学, 2011.
[7]殷书彦 杜庆东.基于蚁群算法的 ACO-BP 神经网络性能研究[J]. 黑龙江科技信息, 2015, (22):178-178
[8]姜攀.基于神经网络的风电机组控制系统故障诊断研究[D]. 北京：华北电力大学, 2010.
[9]程丽萍 周文全 张艳琦 刘敬光.BP 网络技术在 LED 系统总体性能评估中的应用研究[J]. 标准科学,2013, 472(9):47-49
[10]郝中华.BP 神经网络的非线性思想[J]. 洛阳师范学院学报, 2008, 27(4):51-55
[11]黄忠明 吴志红 刘全喜.几种用于非线性函数逼近的神经网络方法研究[J]. 兵工自动化, 2009, 28(10):88-92
[12]张传垒 葛为民 宋博.机械手轨迹规划中的 BP 神经网络方法及仿真[J]. 组合机床与自动化加工技术, 2007, (10):22-24
[13]丁杏娟.基于人工神经网络的产品需求预测研究[D]. 上海：上海交通大学, 2007.
[14]段明秀 何迎生.基于 LVQ 神经网络的手写字母识别[J]. 吉首大学学报（自然科学版）, 2010, 31(2):41-43
[15]尤倩.基于 SVM 的脱机手写体数字识别的研究与应用[D]. 山东：山东师范大学, 2014.

致　谢

在论文完成之际，首先向我的毕业设计导师饶道娟老师和穆国旺教授表示最真挚的感谢。作为我最尊敬的老师，穆国旺教授具有极强的责任心和更强的专业能力，正是凭借着超群的专业能力，在我做论文初期就给了我一个正确的方向。在我论文中期遇到了瓶颈，穆老师又积极帮我分析情况，才能在最困难的时候又出现了突破。通常是穆老师的一句点评，都足以让我收获很大。

感谢睢佰龙老师，正是使用睢老师的实验室，在实验室的工作站上运行 MATLAB，节省了我大量的时间，使我能够腾出时间做更有用的工作。睢老师也提出了很多有用的见解，让我受益无穷。

感谢李波同学，是她每天陪我学习，做论文，在我坚持不下去的时候鼓励、支持我，在我最忙碌的时候给我买饭，在我最烦躁的时候开导我，谢谢。

最后，再次向所有关心、支持和帮助过我的老师、亲人和朋友们以及每个论文的作者们致以最真诚的谢意！