哈爾濱工業大學

实验报告

姓　　名：何亮　　学　号：15S005053

院　　系：电信学院　　班　级：信息一班

课程名称：模式识别

实验序号：1　　实验日期：2016.06.30

实验名称：神经网络用于模式识别

实验成绩：　　总成绩：

教师评语

教师签字：

2016 年　　月　　日

1 实验目的

1）掌握神经网络的基本原理和工作机制，并学会将其用于模式识别。

2）掌握三层前馈神经网络的基本原理以及其反向传播算法（BP 算法）原理，并熟悉如何编程实现该算法。

3）仿真 BP 算法在不同参数下的收敛情况，根据结果分析学习效率、惯性系数、总的迭代次数、训练控制误差、初始化权值以及隐层节点数对网络性能的影响。

2 实验原理

2.1 人工神经元

我们知道生物神经元如图 2-1 所示，包含如下四个部分：

胞体：是神经细胞的本体（可看成系统）；

树突：长度较短，接受自其他神经元的信号（输入）；

轴突：它用以输出信号；

突触：它是一个神经元与另一个神经元相联系的部位，是一个神经元轴突的端部将信号（兴奋）传递给下一个神经元的树突或胞体；对树突的突触多为兴奋性的，使下一个神经元兴奋；而对胞体的突触多为抑制性，其作用是阻止下一个神经元兴奋。

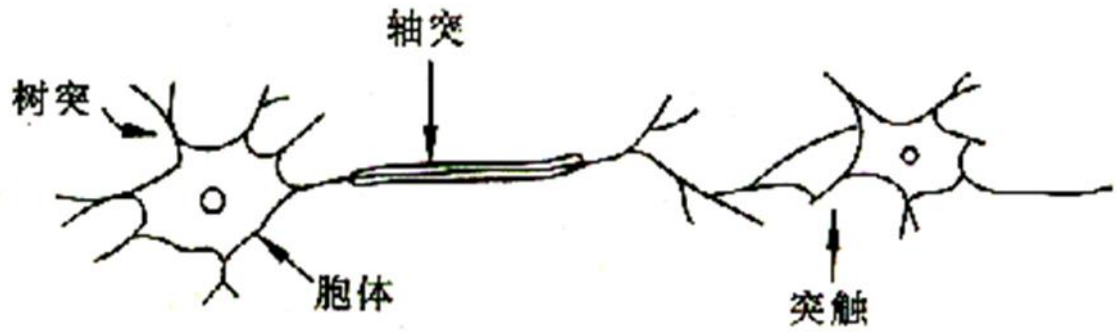


图 2-1 生物神经元示意图

神经元包含两种状态——兴奋和抑制，其工作机制是神经元平时处于抑制状态的神经元，其树突和胞体接收其他神经元经由突触传来的兴奋电位；多个输入在神经元中以代数和的方式叠加；如果输入兴奋总量超过某个阈值，神经元就会

被激发进入兴奋状态，发出输出脉冲，并由轴突的突触传递给其他神经元。神经元被触发之后有一个不应期，在此期间内不能被触发，然后阈值逐渐下降，恢复兴奋性。人体的所有反应都是基于这样的原理传递信息并处理信息的，而人脑可看作是由大量神经元组成的巨大的神经网络。

从神经元的基本功能出发，科学家们提出了人工神经元的模型用于模拟生物神经元的功能。而若干人工神经元组成人工神经网络，是一种模仿动物神经网络行为特征，进行分布式并行信息处理的算法数学模型。这种网络依靠系统的复杂程度，通过调整内部大量节点之间相互连接的关系，从而达到处理信息的目的。人工神经网络具有自学习和自适应的能力，可以通过预先提供的一批相互对应的输入－输出数据，分析掌握两者之间潜在的规律，最终根据这些规律，用新的输入数据来推算输出结果，这种学习分析的过程被称为“训练”。人工神经元是人工神经网络的基本单元，其数学模型如图 2-2，其中输入 x_i 相当于其他神经元的输出，权值 ω_i 相当于突触的连接强度，f 是一个非线性函数，如阈值函数或 Sigmoid 函数用于模拟生物神经元的刺激处理操作。

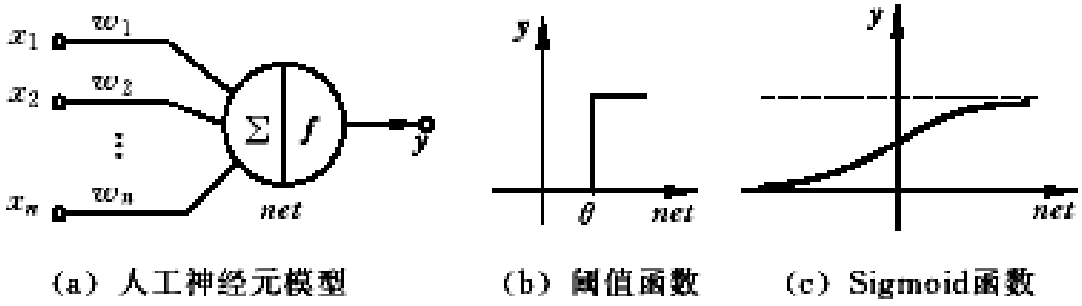


图 2-2 人工神经元基本模型

图 2-2(a)的基本模型表述为数学形式如式(2-1)，当其中 f 为阈值函数时，其输出如式(2-2)所示。

$$net = \sum_{i=1}^{n} w_i x_i, \quad y = f(\text{net}) \tag{2-1}$$

$$y = \text{sgn}\left(\sum_{i=1}^{n} w_i x_i - \theta \right) \tag{2-2}$$

将其表述为矩阵形式如式(2-3)

$$y = f(\mathbf{W}^T \mathbf{X}), \mathbf{W} = (\text{w}_0, \text{w}_1, ..., \text{w}_n)^T, \mathbf{X} = (1, \text{x}_1, ..., \text{x}_n)^T \tag{2-3}$$

其中 $\theta = -w_0$。

选择不同的输出函数 f，y 的取值范围也不同，而函数 f 通常要求选择可微的

函数，此时通常选择 Sigmoid 函数，如式(2-4)所示。该函数具有如下特性：1）非线性，单调性。2）无限次可微。3）当权值很大时可近似阈值函数。当权值很小时可近似线性函数。

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

2.2 前馈神经网络

构成前馈神经网络的神经元接受前一级输入，并输出到下一级，无反馈，如图 2-3。其节点分为两类：输入节点与计算单元。每个计算单元可有任意个输入，但只有一个输出，而输出可藕合到任意多个其他节点的输入。前馈网络通常分为不同的层，第 i 层的输入只与第 i-1 层的输出相联。输入节点为第一层。输入和输出节点由于可与外界相连，称为可见层，而其他的中间层则称为隐层。而其中三层前馈网络又是其特例，三层前馈网络由输入层、中间层和输出层构成。有两个计算层，也称三层感知器，能够求解非线性问题。三层或三层以上的前馈网络通常又被叫做多层感知器（Multi-Layer Perceptron．简称 MLP）。由三部分组成：①、一组感知单元组成输入层；②、一层或多层计算节点的隐藏层；③、一层计算节点的输出层。MLP 的表示：输入节点数-第 1 隐层节点数-第 2 隐层节点数-…-输出节点数。如图 2-3 可表示为：4-4-3 网络

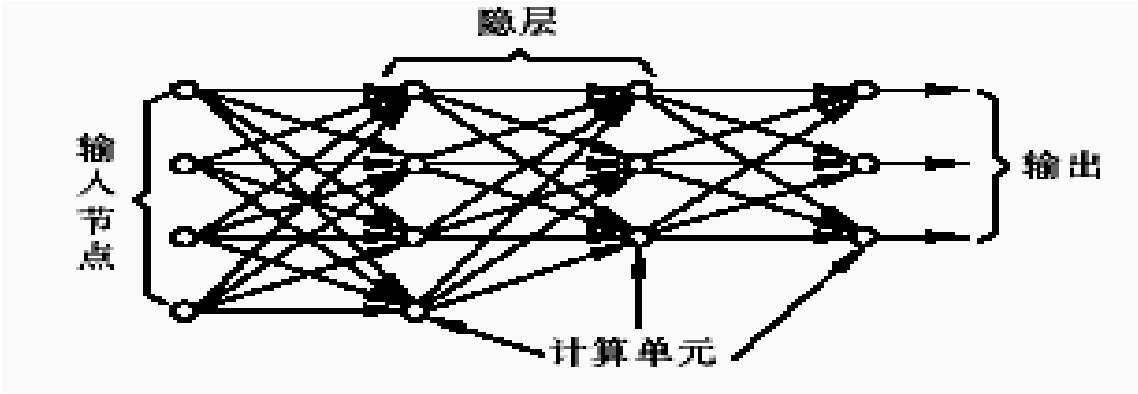


图 2-3 前馈神经网络模型

当使用阈值函数作为神经元输出函数时，任何逻辑函数都可以使用一个三层的前馈网络实现，例如与非门可以使用如下数学公式 $y=\mathrm{sgn}(-0.5x_1-0.5x_2+0.75)$实现。当神经元的输出函数为 Sigmoid 函数时，上述结论可以推广到连续的非线性函数。在很宽松的条件下，三层前馈网络可以逼近任意的多元非线性函数，突破了二层前馈网络线性可分的限制。

要使神经元合理的组合输出，需要先让其学习以模拟生物的学习过程。神经元的学习算法通常采用 Hebb 学习规则，如果神经元 u_i 接收来自另一神经元 u_j 的输出，则当这两个神经元同时兴奋时，从 u_j 到 u_i 的权值 w_{ij} 就得到加强。具体到前述的神经元模型，可以将 Hebb 规则表现为如下的算法 $\Delta\omega_i = \eta y x_i$。式中 $\Delta\omega_i$ 是对第 i 个权值的修正量，η 是控制学习速度的系数。太大会影响训练的稳定性，太小则使训练的收敛速度变慢，一般取 $0 < \eta \leqslant 1$；人工神经网络首先要以一定的学习准则进行学习，然后才能工作。

2.3 BP 学习算法

二层前馈网络的学习算法通常采用感知器的学习规则，其规则如下：可以用已知类别的模式向量或特征向量作为训练集，若 $X \in \omega_j$，应使对应于该类的输出节点的输出 $O_j=1$，而其他节点的输出则为 $O_i=0$ （或-1）。

1）设理想的输出为：$Y=[y_1,y_2,y_3,\ldots,y_m]^T$；

2）某次迭代(k)上的实际输出为： $Y'=[y'_1,y'_2,y'_3,\ldots,y'_m]^T$

3）对权值 ω 利用 Hebb 规则作如下的修改：$w_{ij}(k+1)= w_{ij}(k)+ \Delta w_{ij}(k)$，其中：$\Delta w_{ij}(k)=\eta(y_j-y'_j)x_j$， $i=1,2,\ldots,n$; $j=1,\ldots,m$

其中 η 为学习率，用于控制调整速度，太大会影响训练的稳定性，太小则使训练的收敛速度变慢，一般 $0 < \eta \leqslant 1$。

三层前馈网络的适用范围大大超过二层前馈网络，但学习算法较为复杂，主要困难是中间的隐层不直接与外界连接，无法直接计算其误差。 而 BP 算法可解决这一问题。其主要思想是从后向前（反向）逐层传播输出层的误差，以间接算出隐层误差。因此算法分为两个阶段：

第一阶段（正向过程）：输入信息从输入层经隐层逐层计算各单元的输出值；

第二阶段（反向传播过程）：输出误差逐层向前算出隐层各单元的误差，并用此误差修正前层权值。

在反向传播算法中通常采用梯度法修正权值，为此要求输出函数可微，通常采用 Sigmoid 函数作为输出函数。下面考虑一个如图 2-4 所示的三层前馈网络，其中 i 表示前层第 i 个单元，k 表示后层第 k 个单元，O_j 表示本层的输出，w_{ij} 表示前层到本层的权值。其中的输出函数选择 Sigmoid 函数。

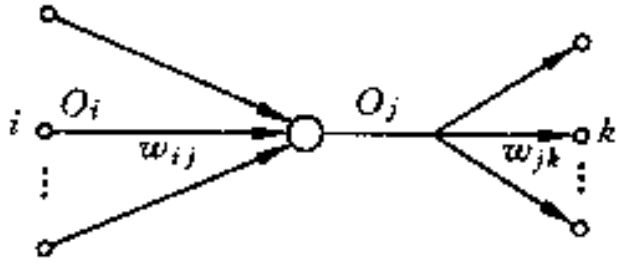


图 2-4 三层前馈网络简化模型

在第一阶段的正向过程中，某层的输出为式(2-5)所示，第一阶段从输入层逐层往前计算输出，直到之后一层。

$$net_j = \sum_{i=1}^{n} w_{ij} O_i, \quad O_j = f(\text{net}_j) = \frac{1}{1+e^{-net_j}} \tag{2-5}$$

在第二阶段的反向传播过程中，对于输出层，O_k 是实际输出，与理想输出 y_k 的总误差可以通过式(2-6)计算，其中 N 表示训练样本格式。

$$E = \frac{1}{N} \sum_{k} (y_k - O_k)^2 \tag{2-6}$$

输出层的局部梯度为式(2-7)

$$\delta_j = \frac{\partial E}{\partial net_j} = -(y_j - O_j) O_j (1 - O_j) \tag{2-7}$$

因此可以计算权值对误差的影响如式(2-8)

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial net_j} \frac{\partial net_{ij}}{\partial w_{ij}} = \delta_j O_j \tag{2-8}$$

采用负梯度修正权值，但是实际计算时，为了加快收敛速度，一般要加上前次权值修正量 α，称为惯性量，因此实际的权值修正量如式(2-9)

$$\Delta w_{ij} = -\eta \delta_j O_i + \alpha \Delta w_{ij}(k-1) \ \Delta w_{ij} = -\eta \delta_j O_i + \alpha \Delta w_{ij}(k-1) \tag{2-9}$$

修正算法可以写为式(2-10)

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}(k) \tag{2-10}$$

若节点 j 不是输出单元，则其局部梯度计算采用式

$$\delta_j = \frac{\partial E}{\partial net_j} = \sum_k \frac{\partial E}{\partial net_k}\frac{\partial net_k}{\partial O_j}\frac{\partial O_j}{\partial net_j} = \sum_k \delta_k w_{jk} f'(net_j) = O_j(1-O_j)\sum_k \delta_k w_{jk} \qquad (2\text{-}11)$$

当采用逐个样本修正时，没输入一个样本就修正一次权值，程序流程参考算法 1。

算法 1 – 逐个样本修正的 BP 算法

S1: 选定(0,1)之间均匀分布的权系数 w_{ij}，步长 η，惯性系数 α；准备训练样本 Xin 及其理想输出 Yd；

S2: 根据公式(2-5)从前向后计算各层各单元输出（前向）O_j；

S3: 根据公式(2-6)计算输出层误差 E；当 E 小于一个设定的常数 E0 时，跳转到 S8，否则继续下一步；

S4: 根据公式(2-7)对输出层计算局部梯度 δ_k；

S5: 根据式(2-11)逐层向前计算各层的局部梯度 δ_j；

S6: 根据公式(2-10)和(2-9)修正权值；然后跳转到 S2。

S7: 结束修正，返回权值 w_{ij}。

采用成批样本修正时，可以在全部训练样本输入完成后统一修正权值，程序流程参考算法 2。其中成批样本修正时对于 N 个样本，全部输入完毕后统一计算权值修正了如式(2-12)，式中 p 表示 N 个样本中的第 p 个样本。

$$\Delta w_{ij} = -\left(\sum_{p=1}^{N} \eta \delta_{jp} O_{ip}\right) + \alpha \Delta w_{ij}(k-1) \qquad (2\text{-}12)$$

算法 2– 成批样本修正的 BP 算法

S1: 选定(0,1)之间均匀分布的权系数 w_{ij}，步长 η，惯性系数 α；准备训练样本 Xin 及其理想输出 Yd；

S2: 对 $p \in \{1,2,...,N\}$ 执行 S3 到 S6 步骤，完成后跳转到 S7；

S3: 根据公式(2-5)从前向后计算各层各单元输出（前向）O_{jp}；

S4: 根据公式(2-6)计算输出层误差 E；当 E 小于一个设定的常数 E0 时，跳转到 S8，否则继续下一步；

S5: 根据公式(2-7)对输出层计算局部梯度 δ_{kp}；

S6: 根据式(2-11)逐层向前计算各层的局部梯度 δ_{jp}；

S7: 根据公式(2-10)和(2-12)修正权值；然后跳转到 S2。

S8: 结束修正，返回权值 w_{ij}。

3 实验内容

3.1 实验内容

采用多输入多输出型结构，构造三层神经网络，训练它用来将表中的三类样本分开。

1) 给出训练结束时的各网络的各节点权值。给出网络迭代终止时实际的迭代次数及总误差和错误率。设隐层节点数为 4，学习效率 η=0.1，惯性系数 α=0.0；训练控制总的迭代次数 N=100000；训练控制误差：e=0.3。在采用 0~1 内均匀分布随机数初始化所有权值。

2) 分析学习效率 η，惯性系数 α；总的迭代次数 N；训练控制误差 e、初始化权值以及隐层节点数对网络性能的影响。绘出学习曲线，训练误差与迭代次数的关系曲线。并将得到的网络对训练样本分类，给出错误率。

3) 采用批处理 BP 算法重复(1)。比较两者结果。

样本如下：

表 3-1 样本数据

样本	第一类			第二类			第三类		
	X1	X2	X3	X1	X2	X3	X1	X2	X3
1	1.58	2.32	-5.8	0.21	0.03	-2.21	-1.54	1.17	0.64
2	0.67	1.58	-4.78	0.37	0.28	-1.8	5.41	3.45	-1.33
3	1.04	1.01	-3.63	0.18	1.22	0.16	1.55	0.99	2.69
4	-1.49	2.18	-3.39	-0.24	0.93	-1.01	1.86	3.19	1.51
5	-0.41	1.21	-4.73	-1.18	0.39	-0.39	1.68	1.79	-0.87
6	1.39	3.16	2.87	0.74	0.96	-1.16	3.51	-0.22	-1.39
7	1.20	1.40	-1.89	-0.38	1.94	-0.48	1.40	-0.44	0.92
8	-0.92	1.44	-3.22	0.02	0.72	-0.17	0.44	0.83	1.97
9	0.45	1.33	-4.38	0.44	1.31	-0.14	0.25	0.68	-0.99
10	-0.76	0.84	-1.96	0.46	1.49	0.68	-0.66	-0.45	0.08

3.2 数据分析

本次实验采用 3-4-3 的三层前馈神经网络来实现对上表所示三类数据进行分类，首先绘制三类样本的空间分布图如图 3-1 所示，其中不同颜色表示不同类别

（红色代表第一类、绿色代表第二类、蓝色代表第三类）。从图 3-1 中可以看出样本分布情况，各类别之间虽然存在一定的分界面，但是其交遇区样本较多，特别是红色类别存在一个错误比较严重的样本。这中分布情况对于一般的分类器很难实现，就算对于神经网络也需要较多的隐层节点过拟合方能实现很好的分类，因此针对本次的网络结构需要首先对样本进行处理，去除该红色样本后再用于网络的学习训练。

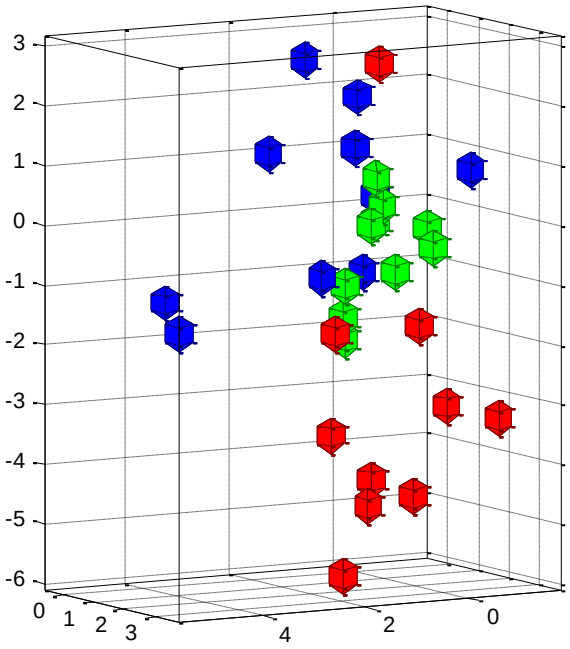


图 3-1 训练样本的空间分布

3.3 算法实现过程和二维数据的测试

编写神经网络的生成函数 neural_net_gen，根据输入的层数、各层神经元节点数、误差界限、最大迭代次数、步长、惯性系数生成神经网络结构体，并完成权值 w 矩阵的初始化操作，将其初始化权值为 0 到 1 之间均匀分布的随机数。

编写神经网络的训练函数 neural_net_train_per 和 neural_net_train_mass，分别采用逐个样本修正和成批样本修正实现神经网络的 BP 学习算法，根据输入的神经网络结构体和训练样本迭代按照算法 1 和算法 2 的步骤进行训练，最终输出训练结束后的结构体。

编写网络分类函数 neural_net_clasify 实现对输入样本的分类，输出参数就是

分类结果。

使用自己生成的两类二维样本数据进行网络训练（各类 30 个样本），采用 2-4-1 的三层神经网络，训练样本如图 3-2 所示，迭代终止条件为平均误差小于 1e-3，最大迭代步数为 15000，步长为 0.1、惯性系数为 0。训练结束后的学习曲线如图 3-3 所示（注：图中的误差采用全部样本的均方误差），最终训练样本全部分类正确，由此可见学习算法的正确性。

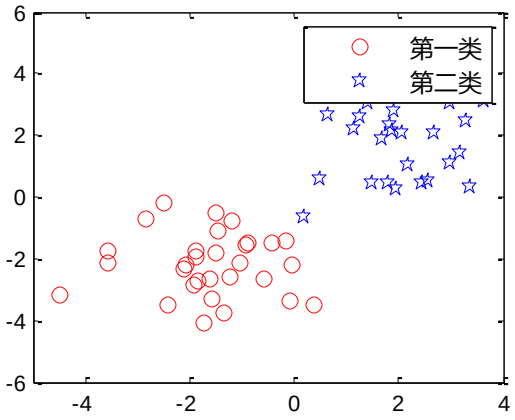


图 3-2 二维训练样本

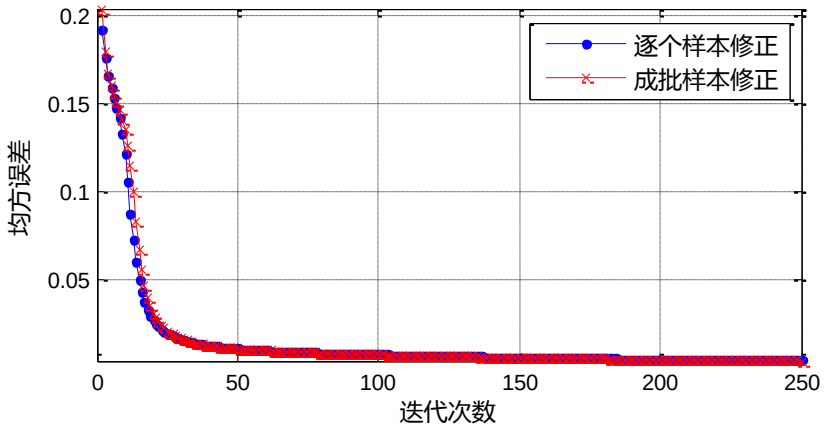


图 3-3 二维样本训练误差

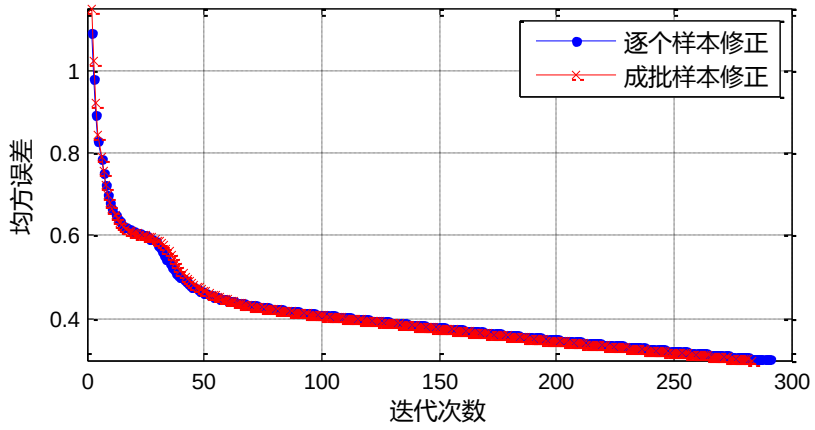
因此下面将用本次编写的三个函数实现对表 3-1 所示的三类样本进行分类，根据前面的介绍，去除第六个第一类样本后用于网络的学习训练，给出学习曲线，之后将全部样本输入计算网络的错误率。

4 实验结果和分析

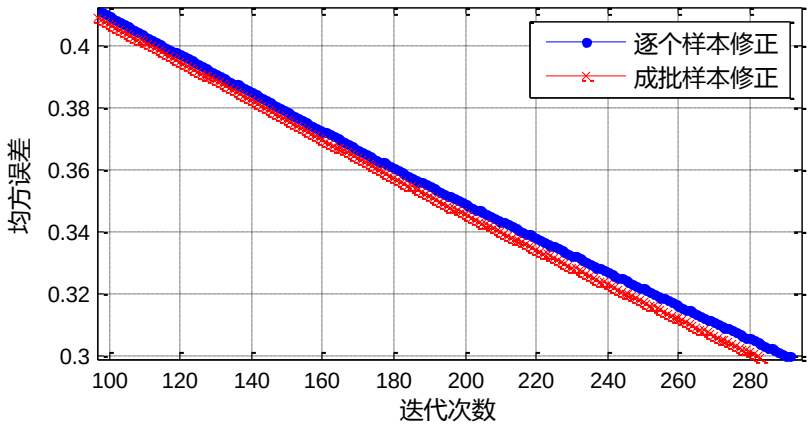
1）针对表 3-1 的数据，设隐层节点数为 4，学习效率 η =0.1，惯性系数 α

=0.0；训练控制总的迭代次数 N=100000；训练控制误差：**e=0.3**。在采用 0~1 内均匀分布随机数初始化所有权值。

分别采用成逐个样本修正和成批样本修正的 BP 算法，最终对于全部数据，错误率为 16.7%（也即 30 个样本中错分数量为 5 个），逐个样本修正的 BP 算法在第 8411 步时均方误差小于 0.3，成批样本修正在 8179 时均方误差小于 0.3，得到的学习曲线如图 4-1 所示。



(a) 完整学习曲线



(b) 学习曲线局部放大

图 4-1 基本参数下的学习曲线

从图 4-1(a)中可以看出，整体而言逐个样本修正和成批样本修正的学习曲线相差不大，而从图 4-1(b)中可以看出，成批样本修正较逐个样本修正而言在当前情况下收敛速度稍微快一些，原因是因为成批样本修正每次修正权值时综合考虑了所有样本的局部梯度，因此收敛速度较快。另外可以看出，采用均方误差的学习曲线设置误差控制误差为 0.3 较大，因此(2)中将其设置更小以便进一步分析。

2）针对(1)中的分析，训练控制误差设置为 e=0.03，其它参数同(1)，对网络

进行训练。

分别采用成逐个样本修正和成批样本修正的 BP 算法，最终对于全部数据，错误率为 10%（也即 30 个样本中错分数量为 3 个），逐个样本修正的 BP 算法在步数用完时均方误差为 0.121，成批样本修正在步数用完时均方误差为 0.118，得到的学习曲线如图 4-2。

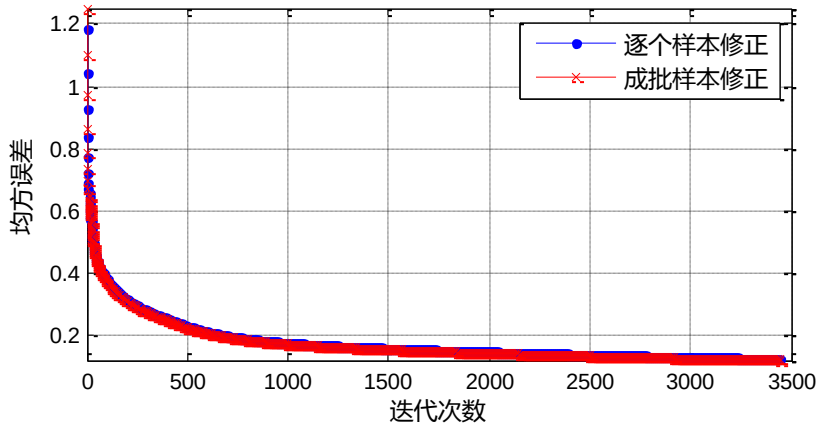


图 4-2 缩小控制误差下的学习曲线

从图 4-2 中可以看出，无论逐个样本修正还是成批样本修正的 BP 算法，在当前参数和样本下 BP 算法随着迭代次数的增加去均方误差总是不断减小的，但是减小的幅度也是不断减小的，可以推测最终逼近一个阈值，该阈值和样本本身的分布有关，而表 3-1 中的样本存在的交遇区导致了错分的必然性，因此在当前网络结构中均方误差无法逼近 0 值。

结合图图 4-1 和图 4-2 可以看出，网络的迭代次数和训练控制误差之间存在单调性，即迭代步数越多误差越小。通常学习终止的条件是通过训练控制误差来控制的，但是网络的均方误差存在一个逼近值，当控制误差小于该阈值时，不合理的网络结构可能永远无法达到该控制阈值，此时可以采用设置迭代次数来迫使训练终止。

3）针对本次的训练样本，由于其交遇区数据较多可能因为隐层节点数太少导致无法完全拟合分界面，因此本次调整网络结构为 3-20-3，其它参数设置同(2)中所述。

分别采用成逐个样本修正和成批样本修正的 BP 算法，最终对于全部数据，错误率为 3.3%（也即 30 个样本中错分数量为 1 个），逐个样本修正的 BP 算法在第 74183 步迭代时均方误差小于 0.03，成批样本修正在 72820 步迭代时均方误差小于 0.03，得到的学习曲线如图 4-3。

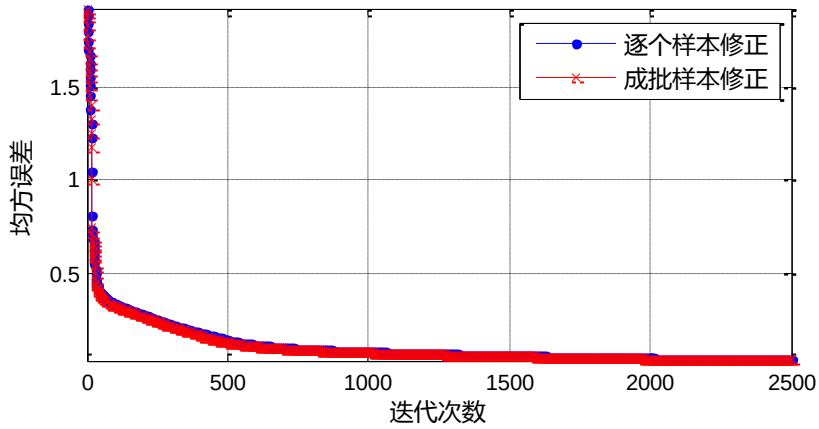


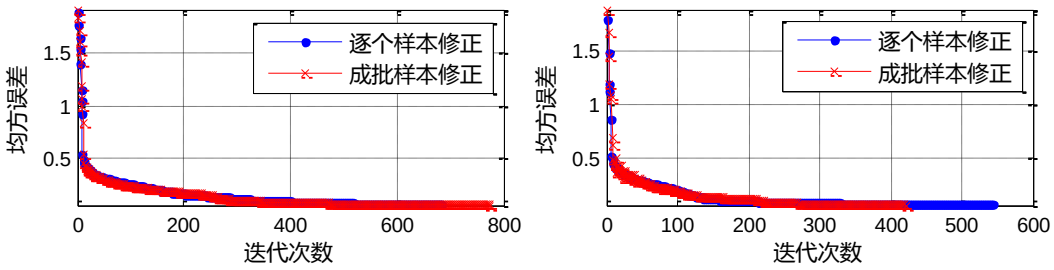
图 4-3 隐层 20 个节点下的学习曲线

对比实验图图 4-2 和图 4-3 可以看出，当隐层节点数目增加时，可以拟合更复杂的分界面（如本次样本的理想分界面），但并不一定是更好的分类器。

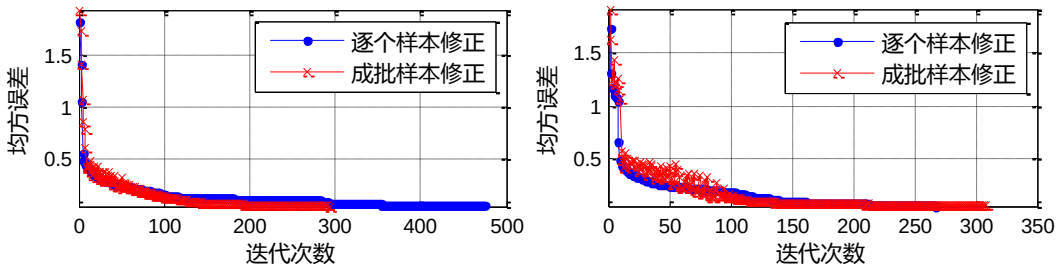
4）针对不同的学习效率和惯性系数，下面通过更改(3)中的不同 η 和 α 来对比收敛性能和迭代错误率，为了方便对比，将控制误差设置为 0.05 以获得时间和性能上的折中。针对不同学习效率 η 的实验结果如表 4-1 和图 4-4 所示（其中 α 取 0）。

表 4-1 不同学习效率下的迭代次数

学习效率 η	迭代次数（逐个样本）	迭代次数（成批样本）
0.1	67774	67252
0.3	15835	15458
0.5	19663	12819
0.7	13747	8469
0.9	7744	8904



(a) η =0.3　　(b) η =0.5



(c) η=0.7　　(d) η=0.9

图 4-4 不同学习效率下的学习曲线

从图 4-4 和表 4-1 可以看出，不同的学习效率 η 下学习曲线不同，通常 η 越大，学习曲线下降越快，但是 η 较大时学习曲线的不稳定程度也较大，因此实际应用中需要合理的选择学习效率 η 值。

针对不同惯性系数 α 的实验结果如所示，其学习曲线在图 4-4(b)的基础上略有波动（其中 η 取 0.5）。

表 4-2 不同惯性系数下的迭代次数

惯性系数 α	迭代次数（逐个样本）	迭代次数（成批样本）
0.1	11427	9078
0.3	11775	12819
0.5	18706	14298
0.7	15777	11746
0.9	14704	11601

从表 4-2 可以看出，本次实验样本和网络下，惯性系数对网络学习速率的影响较小，但是会带来一部分随机因素，因此较小的惯性系数有利于本次实验样本的收敛，原因是本次实验样本交遇区间数据较多。

5 结 论

本次实验从原理、实现和仿真上对三层神经网络及其学习算法进行了深入分析，对比了不同网络结构和算法参数下的学习曲线。根据实验可以得出如下结论：

1）无论逐个样本修正还是成批样本修正的 BP 算法，在当前参数和样本下 BP 算法随着迭代次数的增加去均方误差总是不断减小的，但是减小的幅度也是不断减小的，可以推测最终逼近一个阈值，该阈值和样本本身的分布有关。

2）成批样本修正较逐个样本修正而言在当前情况下收敛速度稍微快一些，原因是因为成批样本修正每次修正权值时综合考虑了所有样本的局部梯度，因此

收敛速度较快。

3）网络的迭代次数和训练控制误差之间存在单调性，即迭代步数越多误差越小。通常学习终止的条件是通过训练控制误差来控制的，但是网络的均方误差存在一个逼近值，当控制误差小于该阈值时，不合理的网络结构可能永远无法达到该控制阈值，此时可以采用设置迭代次数来迫使训练终止。

4）当神经网络隐层节点数目增加时，可以拟合较复杂的分界面，实现对训练样本更好的拟合。

5）不同的学习效率η下学习曲线不同，通常η越大，学习曲线下降越快，但是η较大时学习曲线的不稳定程度也较大，因此实际应用中需要合理的选择学习效率η值；而惯性系数对网络学习的收敛性能影响则相对较小。

6 附录—程序说明

函数和脚本介绍：

函数：

draw_sphere.m 绘制训练样本空间分布的函数
neural_net_clasify.m 使用训练好的网络对输入样本分类
neural_net_gen.m 根据输入的网络参数产生并初始化网络
neural_net_train_mass.m 成批样本修正的 BP 算法
neural_net_train_per.m 逐个样本修正的 BP 算法
sample_gen_2d.m 产生二维测试样本并绘制其分布

脚本：

bp_net_2ddata.m 使用二维测试样本测试网络性能
bp_net_rawdata.m 使用实验数据进行网络性能测试

运行说明：

bp_net_2ddata.m 运行该脚本运行可以产生图 3-2 和图 3-3，但是由于测试样本数随机产生的，因此不能保证曲线完全一直。
bp_net_rawdata.m 运行该脚本可以产生第四节的图，根据实验说明调节脚本中语句 net = neural_net_gen(…)函数的输入参数来产生不同实验结果。

注：

本程序支持多层网络，各层节点数量可以任意设置，已经封装完毕，各个函数的功能参考其注释，实现原理和分析参考实验报告。不排除有没有考虑到的错误，如果发现 bug 将非常感谢您联系作者改正，另外也期待您的建议：helianghit@foxmail.com