单隐层神经网络实验报告

题目要求

1. 使用 C 语言（或伪代码）编写实现“单隐层神经网络”及其训练过程。
2. 自己构建大于等于 100 个数据的训练数据集，并进行训练。
3. 给出训练后神经网络的性能评价值。
4. 讨论更改学习率 η 的值、循环次数对训练的影响。

实现思路

1. 搭建单隐藏层的前馈网络（Multilayer Feedforward Network）。一个输入层（Input Layer），一个隐藏层（Hidden Layer），一个输出层（Output Layer）。
2. 训练采用监督式学习，提供多组样例输入与输出。
3. 权重的更新采用误差反向传播算法（Backpropagation，简称 BP 算法）。

实现细节

变量定义

构建 InputLayer、HiddenLayer、OutputLayer 结构体。其中包含权值 w 数组和输出 output 数组。

其中 $w_{i,j}$ 表示当前层第 i 个节点与上一层第 j 个节点之间的权值。$output_i$ 表示当前层第 i 个节点的输出。

全局变量 eta 为学习率。

初始化

通过预设或输入的输入层，隐藏层，输出层的大小构建神经网络。初始权值使用 0 到 1 之间的随机数。

但在实际训练中，为了适应我们训练数据，我们把全部权值改为 0，得到了更好的训练结果。

```
void init(){
    for(int i = 1; i <= hiddenLayer.siz; ++i){
        hiddenLayer.w[i][0] = (double)rand() / RAND_MAX * inputLayer.siz;
        for(int j = 1; j <= inputLayer.siz; ++j){
            hiddenLayer.w[i][j] = (double)rand() / RAND_MAX;
        }
    }
    for(int i = 1; i <= outputLayer.siz; ++i){
        outputLayer.w[i][0] = (double)rand() / RAND_MAX * hiddenLayer.siz;
        for(int j = 1; j <= hiddenLayer.siz; ++j){
            outputLayer.w[i][j] = (double)rand() / RAND_MAX;
        }
    }
    return;
}
```

运行

依次计算节点的输出值并向后传递。

```
void run(){
    inputLayer.output[0] = -1;
    for(int i = 1; i <= inputLayer.siz; ++i){
        inputLayer.output[i] = input[i];
    }
    hiddenLayer.output[0] = -1;
    for(int i = 1; i <= hiddenLayer.siz; ++i){
        double ans = 0;
        for(int j = 0; j <= inputLayer.siz; ++j){
            ans += inputLayer.output[j] * hiddenLayer.w[i][j];
        }
        ans = f(ans);
        hiddenLayer.output[i] = ans;
    }
    for(int i = 1; i <= outputLayer.siz; ++i){
        double ans = 0;
        for(int j = 0; j <= hiddenLayer.siz; ++j){
            ans += hiddenLayer.output[j] * outputLayer.w[i][j];
        }
        ans = f(ans);
        outputLayer.output[i] = ans;
    }
    return;
}
```

更新

```
void update(){
    for(int i = 1; i <= outputLayer.siz; ++i){
        g[i] = outputLayer.output[i] * (1.0 - outputLayer.output[i]) * (output[i]
- outputLayer.output[i]);
    }
    for(int i = 1; i <= hiddenLayer.siz; ++i){
        double sum = 0;
        for(int j = 1; j <= outputLayer.siz; ++j){
            sum += outputLayer.w[j][i] * g[j];
        }
        e[i] = hiddenLayer.output[i] * (1.0 - hiddenLayer.output[i]) * sum;
    }
    for(int i = 1; i <= outputLayer.siz; ++i){
        for(int j = 0; j <= hiddenLayer.siz; ++j){
            outputLayer.w[i][j] += eta * g[i] * hiddenLayer.output[j];
        }
    }
    for(int i = 1; i <= hiddenLayer.siz; ++i){
        for(int j = 0; j <= inputLayer.siz; ++j){
```

```
                hiddenLayer.w[i][j] += eta * e[i] * inputLayer.output[j];
            }
        }
        return;
    }
```

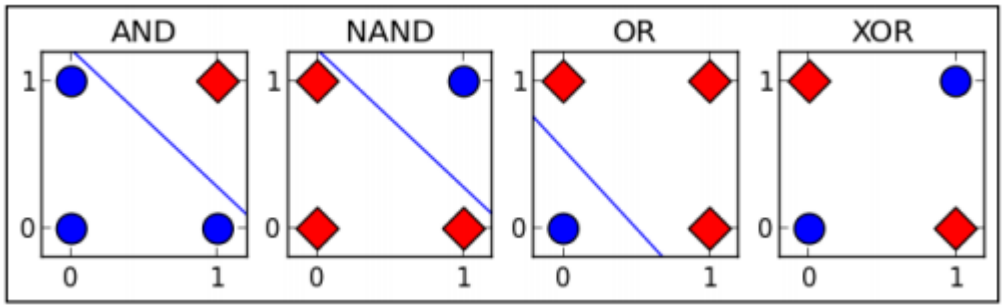
实验

为了研究更改学习率 η 的值、循环次数对训练的影响，采用两个实验分别探究学习率 η 的值、循环次数对训练的影响。

实验一 异或

实验思路

异或操作是计算机中一种常见的简单位运算，同时它具有简单，非线性可分的特点，单层神经网络（也叫感知机）做不到异或，而带有隐层的神经网络可以实现。所以异或非常适合用于验证单隐层神经网络是否成功搭建。



搭建 2 个输入节点，3 个隐层节点，改变学习率和循环次数，训练不同的神经网络来计算异或，再判断准确率，由此得到学习率 η 的值、循环次数对训练的影响。

实验结果

学习次数（次）	$\eta = 0.001$	$\eta = 0.01$	$\eta = 0.1$
1	0.5	0.5	0.5
10	0.5	0.5	0.5
100	0.5	0.5	0.5
1000	0.5	0.5	0.5
10000	0.5	0.5	1
100000	0.5	1	1

实验结论

实验二 图形识别

实验思路

构建 100 个节点输入层，100 个节点隐藏层，1 结点输出层的神经网络。输入 10×10 由 $0,1$ 组成的字符画，识别图像为圆形还是矩形。以下字符画分别为圆形和矩形。

```
0 0 0 1 1 1 1 1 0 0
0 0 1 1 1 1 1 1 1 0
0 0 1 1 1 1 1 1 1 0
0 1 1 1 1 1 1 1 1 1
0 0 1 1 1 1 1 1 1 0
0 0 1 1 1 1 1 1 1 0
0 0 0 1 1 1 1 1 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

0 0 0 1 1 1 0 0 0 0
0 0 0 1 1 1 0 0 0 0
0 0 0 1 1 1 0 0 0 0
0 0 0 1 1 1 0 0 0 0
0 0 0 1 1 1 0 0 0 0
0 0 0 1 1 1 0 0 0 0
0 0 0 1 1 1 0 0 0 0
0 0 0 1 1 1 0 0 0 0
0 0 0 1 1 1 0 0 0 0
0 0 0 1 1 1 0 0 0 0
```

实验结果

学习次数（百次）	$\eta = 0.001$	$\eta = 0.01$	$\eta = 0.1$
1000	0.5	0.5	0.69
10000	0.625	0.655	0.68
100000	0.665	0.655	---

当学习次数达到 100000 后，训练集准确率已经高达 97，而非训练集数据准确率出现下降，已出现过拟合现象，故没有继续测试。

但要指出的是，分类出错部分主要是半径为 3 的圆形，此时受限于字符画的大小，圆形和方形及其相近。准确率不足的另一个原因还在于训练集过小。

实验结论

训练集中准确率与学习率 η 的值、循环次数成正相关。

测试集准确率在学习率 η 的值、循环次数较小时，与学习率 η 的值、循环次数成正相关。但学习率 η 的值、循环次数过大时，会出现过拟合现象，导致准确率下降。

总结与反思

通过这次实验，我们用 C 语言成功构建了一个单隐层神经网络，并用它研究了更改学习率 η 的值、循环次数对训练的影响，取得初步成果。

为了提高测试集准确率，我们可以考虑使用更大的训练集，并改进训练方法，比如可以采用留出法、交叉验证法、自助法等方法。