1.bp网路介绍

BP神经网络又称误差反向传递神经网络。它是 一种依靠反馈值来不断调整节点之间的连接权值而 构建的一种网络模型。它的整个体系结构如图1所 示,分为输入层、隐藏层和输出层,其中隐藏层根据 具体情况的需要,可以是一层结构也可为多层结构。 BP算法的基本思想是:学习过程由信号的正向传播与误差的反向传播两个过程组成。正向传播时,输入样本从输入层传入,经各隐藏层逐层处理后,传向输出层。若输出层的实际输出与期望的输出(教师信号)不符,则转入误差的反向传播阶段。误差反传是将输出误差以某种形式通过隐藏层向输入层反传,并将误差分摊给各层的所有单元,从而获得各层单元的误差信号,此误差信号即作为修正各单元权值的依据。这种信号正向传播与误差反向传播的各层权值调整过程,是周而复始地进行的。权 值不断调整的过程,也就是网络的学习训练过程。此过程一直进行到网络输出的误差减少到可接受到的程度,或进行到预先设定的学习次数为止。

2.手写识别步骤

a.图像预处理（不需学生完成）为28*28的0~255的数字。

b.特征提取（我的方法是把所有大于0的变成1）

c.神经网络结构的确定

输入层：28*28=784个神经元

输出层：0~9 共10个神经元

隐藏层：利用网上查到的公式得到为74个神经元，其中n为输入层神经元个数,m为输出层神经元个数

$$s = \sqrt{0.43mn + 0.12m^2 + 2.54n + 0.77m + 0.35} + 0.51$$

d.网络的训练（总共为70000个数据）

利用其中6万个进行训练，另外1万进行测试，训练次数为10次

&具体实现：

训练时候以5000为一个间隔，对其余1万内的随机2000个左右进行测试正确率，然后输出

而对于网络权重，则每5次计算更新一次权重。

利用公式进行权重更新

```
//adjust weight
for (int j = 0; j < hiddenNumber; j++) {
    for (int i = 0; i < outputNumber; i++) {
        adjustWeightJi[j][i] += rate * (standard[i] - output[i]) * output[i] * (1 - output[i]) * hidden[j];
    }
}
for (int k = 0; k < inputNumber; k++) {
    for (int j = 0; j < hiddenNumber; j++) {
        outputTotal = 0;
        for (int i = 0; i < outputNumber; i++) {
            outputTotal += (standard[i] - output[i]) * output[i] * (1 - output[i]) * weightJi[j][i];
        }
        adjustWeightKj[k][j] += rate * hidden[j] * (1 - hidden[j]) * input[k] * outputTotal;
    }
}

//adjust bias
for (int i = 0; i < outputNumber; i++) {
    adjustBiasI[i] += rate * (standard[i] - output[i]) * output[i] * (1 - output[i]);
}
for (int j = 0; j < hiddenNumber; j++) {
    outputTotal = 0;
    for (int i = 0; i < outputNumber; i++) {
        outputTotal += (standard[i] - output[i]) * output[i] * (1 - output[i]) * weightJi[j][i];
    }
    adjustBiasJ[j] += rate * hidden[j] * (1 - hidden[j]) * outputTotal;
}
```

&训练结果保存
　　取最高正确率的进行保存
e.测试结果
默认训练方法测试结果正确率为96%

3.文件说明
dataset　　数据集
Main.java　主要训练文件
Main.java　按照error值递减训练
Initial.java　初始化文件
Renew.java 更新文件
Test.java　测试文件
Final.java　面试测试文件
TestSave.txt 最优权重保存

4.不同参数下性能比较

**默认
训练数据为1-70000（不包含50000-59999），测试数据为50000-59999
每训练5000个测试2000个，所有数据训练周期为10轮（10*60000=60万次）
每训练5次更新一次权重
隐藏层个数为74
训练率为0.05

a.不同隐藏层个数（hiddenNumber)

	30	74	200
正确率	94.5%	96.7%	98.2%
第一次达到90%花费次数	30000	25000	40000
初始训练率（最初的5000个）	70.5%	79.3%	78.9%
5000个花费的时间(s)	3.3	5.6	15.2

隐藏层数量对正确率影响较大，但是第一次到达较好值则相对变慢，但是由于隐藏层数量增加的时间花费也不容小觑。

b.训练率（rate）

	0.01	0.05	0.05-0.01随时间递减
正确率	96.5%	96.7%	97.3%
第一次达到90%花费次数	95000	25000	55000
初始训练率（最初的5000个）	24.0%	79.3%	79.6%

训练率为0.01时初始训练率极低，但最终正确率也并不高，而随时间递减的训练率也并没有很好的效果，故采用0.05较为合理。

c.训练次数（trainNumber)

	1	10	30
正确率	91.8%	96.7%	97.0%
达到最高值时训练百分比	83%	75%	93.1%

花费时间显而易见是和训练次数成正比的，故没有写出，训练次数为1 时候明显正确率比较低，而训练过多时候，既会消耗更多时间，也没有有效提高正确率，故不采取。

d.更新权重间隔次数（section）

	1	10	20
正确率	96.1%	96.7%	96.6%
第一次达到90%花费次数	35000	25000	35000
初始训练率（最初的5000个）	78.2%	79.3%	78.8%

更新权重间隔次数并没有有效影响各个因素

e.不同测试集(bound)

	1-9999	30000-39999	50000-59999
正确率	96.3%	96.5%	96.7%
第一次达到90%花费次数	35000	25000	25000
初始训练率（最初的5000个）	72.9%	77.6%	79.3%

不同选集影响不大

f.每次训练完测试不同数据量(testScale)

	1000	2000	5000
正确率	95.8%	96.7%	96.7%
第一次达到**90%**花费次数	35000	25000	35000
初始训练率（最初的**5000**个）	77.5%	79.3%	78.5%

明显测试越多，越会符合真实的，但是实际测试，5000并没有比2000有提高正确率，而且其他数据反而变得更弱，而对于只测试1000，则正确率有所下降。

g.每次训练不同数据量进行测试(length)

	2000	5000	20000
正确率	96.8%	96.7%	95.9%
第一次达到**90%**花费次数	24000	25000	40000
初始训练率（最初的**5000**个）	56.7%	79.3%	88.0%

初始训练率随着训练量增加而增加，而正确率则显示出微弱的负相关。