

**软件可靠性课程实验报告**

**题 目: BP神经网络模型与学习算法**

**院 系:** 计算机科学与技术学院

**专 业:** 软件工程

**学生姓名:** 徐宇顺

**学 号:** 031720106

**二零一九 年 十 月 六日**

目录

[1.引言 3](#_Toc529726265)

[1．1编写目的 3](#_Toc529726266)

[2.模型理论 3](#_Toc529726267)

[2.1 模型背景 3](#_Toc529726268)

[2.2 模型假设 3](#_Toc529726269)

[2.3 模型推导 4](#_Toc529726270)

[3.算法实现 5](#_Toc529726271)

[3.1 流程图 5](#_Toc529726272)

[3.2 算法伪代码 6](#_Toc529726273)

[3.3 算法实现 7](#_Toc529726274)

[3.5 数据来源 8](#_Toc529726275)

[3.6 结果展示 8](#_Toc529726276)

[4.总结 9](#_Toc529726277)

[参考文献 9](#_Toc529726278)

# 1.引言

## 1．1编写目的

随着软件规模越来越大，结构日趋复杂，应用日趋广泛。软件危机依然是我们难以逾越的障碍， 加强软件工程管理，势在必行，势在必然！改进和提高软件可靠性，为部队提供可靠顶用的装备是我们的义务和责任！

# 2.模型理论

## 2.1 模型背景

## Rumelhart，McClelland于1985年提出了BP网络的误差反向后传BP(Back Propagation)学习算法

## 2.2 模型假设

利用输出后的误差来估计输出层的直接前导层的误差，再用这个误差估计更前一层的误差，如此一层一层的反传下去，就获得了所有其他各层的误差估计

## 2.3 模型推导

激活函数

必须处处可导

一般都使用S型函数

使用S型激活函数时BP网络输入与输出关系

输入



输出



输出的导数



学习的过程：

神经网络在外界输入样本的刺激下不断改变网络的连接权值,以使网络的输出不断地接近期望的输出。

学习的本质：

对各连接权值的动态调整

学习规则：

权值调整规则，即在学习过程中网络中各神经元的连接权变化所依据的一定的调整规则。

网络结构

输入层有n个神经元，隐含层有p个神经元,

输出层有q个神经元

变量定义

输入向量：

隐含层输入向量：

隐含层输出向量：

输出层输入向量：

输出层输出向量：

期望输出向量：

输入层与中间层的连接权值:

隐含层与输出层的连接权值:

隐含层各神经元的阈值:

输出层各神经元的阈值:

样本数据个数:

激活函数: 

误差函数：

第一步，网络初始化

给各连接权值分别赋一个区间（-1，1）内的随机数，设定误差函数e，给定计算精度值和最大学习次数M。

第二步,随机选取第k个输入样本及对应期望输出





第三步，计算隐含层各神经元的输入和输出



第四步，利用网络期望输出和实际输出，计算误差函数对输出层的各神经元的偏导数。



第五步，利用隐含层到输出层的连接权值、输出层的和隐含层的输出计算误差函数对隐含层各神经元的偏导数。



第六步，利用输出层各神经元的和隐含层各神经元的输出来修正连接权值。



第七步，利用隐含层各神经元的和输入层各神经元的输入修正连接权。



第八步，计算全局误差



第九步，判断网络误差是否满足要求。当误差达到预设精度或学习次数大于设定的最大次数，则结束算法。否则，选取下一个学习样本及对应的期望输出，返回到第三步，进入下一轮学习。

BP算法直观解释

情况一直观表达

当误差对权值的偏导数大于零时，权值调整量为负，实际输出大于期望输出，权值向减少方向调整，使得实际输出与期望输出的差减少。

**who**



**e**

**>0，此时Δwho<0**

情况二直观表达

当误差对权值的偏导数小于零时，权值调整量为正，实际输出少于期望输出，权值向增大方向调整，使得实际输出与期望输出的差减少。



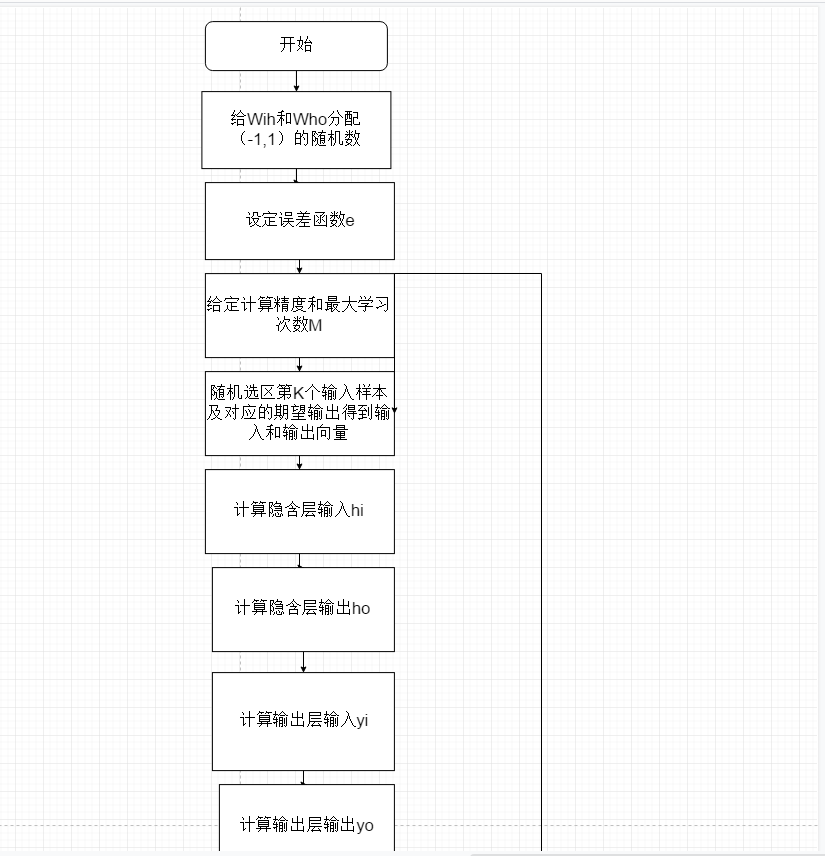
**e**

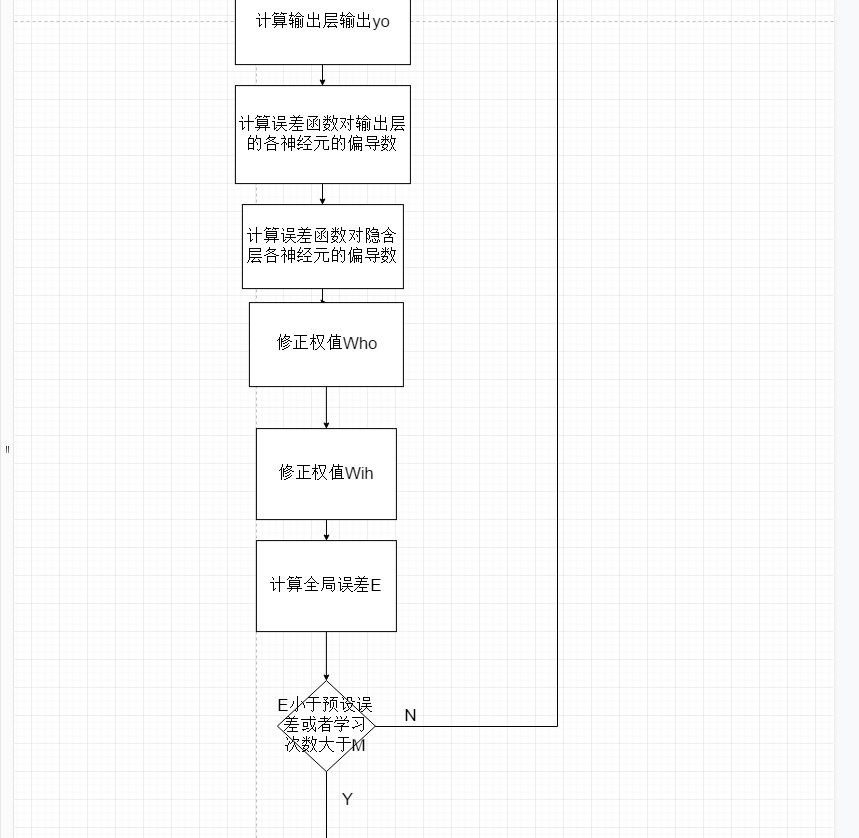
**<0, 此时Δwho>0**

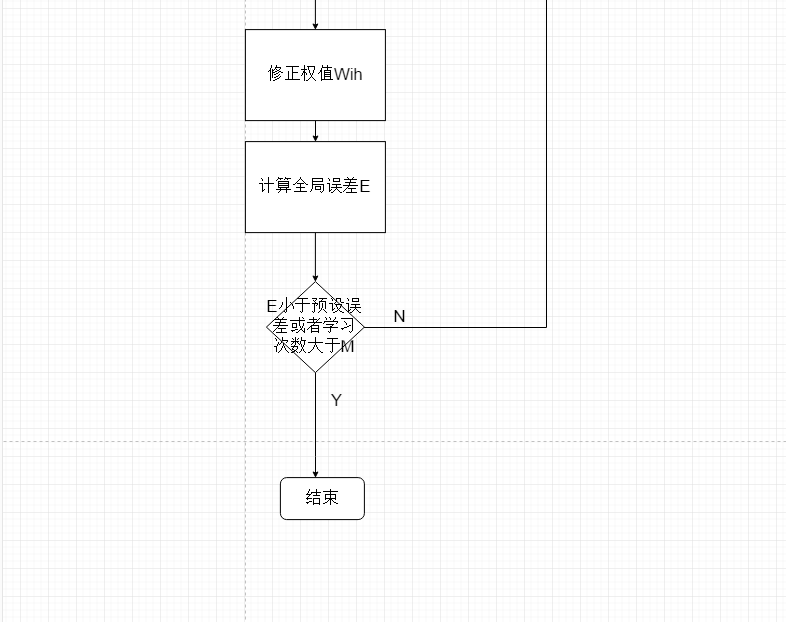
**who**

# 3.算法实现

## 3.1 流程图

****

****

****

## 3.2 算法伪代码

## 

## 3.3 算法实现

本次实验采用Java语言编写，运行的IDE是IDEA。

**import** org.omg.Messaging.SYNC\_WITH\_TRANSPORT;  
  
**import** java.io.BufferedReader;  
**import** java.io.FileNotFoundException;  
**import** java.io.FileReader;  
**import** java.io.IOException;  
**import** java.util.ArrayList;  
  
**public class** BP {  
 ArrayList<Double> **data** = **new** ArrayList<Double>();  
  
 ArrayList<Double> **X** = **new** ArrayList<Double>(); *//输入* ArrayList<Double> **hi** = **new** ArrayList<Double>(); *//隐含层输入* ArrayList<Double> **ho** = **new** ArrayList<Double>(); *//隐含层输出* ArrayList<Double> **yi** = **new** ArrayList<Double>(); *//输出层输入* ArrayList<Double> **yo** = **new** ArrayList<Double>(); *//输出层输出* ArrayList<Double> **d0** = **new** ArrayList<Double>(); *//期望输出* ArrayList<Double> **Wih** = **new** ArrayList<Double>(); *//输入层与中间层的连接权值* ArrayList<Double> **Who** = **new** ArrayList<Double>(); *//隐含层与输出层的连接权值* ArrayList<Double> **bh** = **new** ArrayList<Double>(); *//隐含层各神经元的阈值* ArrayList<Double> **bo** = **new** ArrayList<Double>(); *//输出层各神经元的阈值* ArrayList<Double> **delta\_o** = **new** ArrayList<Double>();  
 ArrayList<Double> **delta\_h** = **new** ArrayList<Double>();  
  
 **double Precision\_Value** = 0.01; *//计算精度，我也不知道定多少哈哈哈哈哈* **int Max\_Learn\_Times** = 200; *//最大学习次数，我也不知道定多少哈哈哈哈hh* **int Now\_Times** = 1;  
 **double rate** = 0.99;  
 **int step** = 1;  
  
 ArrayList<Double> **e** = **new** ArrayList<Double>();  
 **double E** = 100;  
  
 *//总样本数* **int n** = 5;  
 **int p** = 5;  
 **int q** = 5;  
 **int m** = 26;  
  
 **public void** Init(BP bp) {  
 BufferedReader br = **null**;  
 **try** {  
 br = **new** BufferedReader(**new** FileReader(**"D:\\小黄片\\软件可靠性\\实验\\2\\src\\test.txt"**));  
 } **catch** (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
  
 **int** i=0;  
 String line = **null**;  
 **try** {  
 line = br.readLine();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 **while**(line != **null**){  
 String[] numbers = line.split(**"\\s+"**);  
 bp.**data**.add(Double.*valueOf*(numbers[1]));  
 i++;  
 **try** {  
 line = br.readLine();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 **try** {  
 br.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 **for**(i=0;i<bp.**n**;i++){  
 bp.**X**.add((**double**)0);  
 }  
 **for**(i=0;i<bp.**q**;i++){  
 bp.**d0**.add((**double**)0);  
 }  
 **for**(i=0;i<bp.**p**;i++){  
 bp.**bh**.add(0.0001);  
 }  
 **for**(i=0;i<bp.**q**;i++){  
 bp.**bo**.add(0.0001);  
 }  
 **for**(i=0;i<bp.**p**;i++){  
 bp.**hi**.add((**double**)0);  
 }  
 **for**(i=0;i<bp.**p**;i++){  
 bp.**ho**.add((**double**)0);  
 }  
 **for**(i=0;i<bp.**q**;i++){  
 bp.**yi**.add((**double**)0);  
 }  
 **for**(i=0;i<bp.**q**;i++){  
 bp.**yo**.add((**double**)0);  
 }  
 **for**(i=0;i<bp.**q**;i++){  
 bp.**delta\_o**.add((**double**)0);  
 }  
 **for**(i=0;i<bp.**p**;i++){  
 bp.**delta\_h**.add((**double**)0);  
 }  
 **for**(i=0;i<bp.**m**;i++){  
 bp.**e**.add((**double**)0);  
 }  
 }  
  
 **public void** setX(BP bp,**int** k){  
 **int** j = (k-1)%bp.**m**;  
 **for**(**int** i=0;i<bp.**n**;i++){  
 bp.**X**.set(i,bp.**data**.get((i+j\*bp.**n**)%136));  
 }  
 }*//输入* **public void** setd0(BP bp,**int** k){  
 **int** j = (k-1)%bp.**m**;  
 **for**(**int** i=0;i<bp.**q**;i++){  
 bp.**d0**.set(i,bp.**data**.get((i+j\*bp.**n**+bp.**step**)%136));  
 }  
 }*//期望输出* **public void** setW(BP bp){  
 *//(y - Math.random()) % y;* **int** i,j;  
 **boolean** flag = **false**;  
 **double** temp;  
 **for**(i=0;i<bp.**p**;i++){  
 **for**(j=0;j<bp.**n**;j++){  
 temp = Math.*random*();  
 **if**(flag)  
 bp.**Wih**.add(temp);  
 **else** bp.**Wih**.add(-temp);  
 flag = flag^**true**;  
 }  
 }  
  
 flag = **false**;  
 **for**(i=0;i<bp.**p**;i++){  
 **for**(j=0;j<bp.**q**;j++){  
 temp = Math.*random*();  
 **if**(flag)  
 bp.**Who**.add(temp);  
 **else** bp.**Who**.add(-temp);  
 flag = flag^**true**;  
 }  
 }  
 }*//对应的* **public void** getE(BP bp){  
 **int** q= bp.**q**;  
 **double** temp1 = 0;  
 **double** sum = 0;  
 **for**(**int** i=0;i<q;i++){  
 temp1 = bp.**d0**.get(i)-bp.**yo**.get(i);  
 sum += Math.*pow*(temp1,2.0);  
 }  
 bp.**e**.set((bp.**Now\_Times**%bp.**m**),1.0/2\*sum);  
 }*//误差函数* **public void** getHi(BP bp){  
 **double** sum = 0;  
 **double** temp1=0;  
 **for**(**int** h = 0;h<bp.**p**;h++){  
 **for**(**int** i=0;i<bp.**n**;i++){  
 sum+=bp.**Wih**.get(i\*bp.**p**+h)\*bp.**X**.get(i);  
 }  
 temp1 = sum-bp.**bh**.get(h);  
 sum=0;  
 bp.**hi**.set(h,temp1);  
 }  
 }  
  
 **public void** getHo(BP bp){  
 **double** result=0,temp1;  
 **int** i=0;  
 **int** p = bp.**hi**.size();  
 **for**(i=0;i<p;i++){  
 temp1 = 1+Math.*exp*(-bp.**hi**.get(i));  
 result = 1.0/temp1;  
 *// System.out.print("bp.hi.get(i)="+bp.hi.get(i)+" temp: "+temp1+"\n");* bp.**ho**.set(i,result);  
 }  
 }  
  
 **public void** getYi(BP bp){  
 **double** sum=0;  
 **double** temp1;  
 **for**(**int** o=0;o<bp.**q**;o++){  
 **for**(**int** h=0;h<bp.**p**;h++){  
 sum+=bp.**Who**.get(h\*bp.**q**+o)\*bp.**ho**.get(h);  
 }  
 temp1 = sum-bp.**bo**.get(o);  
 sum=0;  
 bp.**yi**.set(o,temp1);  
 }  
 }  
  
 **public void** getYO(BP bp){  
 **double** result=0,temp1;  
 **int** i=0;  
 **int** q = bp.**yi**.size();  
 **for**(i=0;i<q;i++){  
 temp1 = 1+Math.*exp*(-bp.**yi**.get(i));  
 result = 1.0/temp1;  
 *// System.out.println("\ntemp1 = "+temp1);* bp.**yo**.set(i,result);  
 }  
 */\* System.out.println("\n\n这次的所有yi和yo:");  
 for(i=0;i<bp.q;i++){  
 System.out.println(" "+bp.yi.get(i)+" "+bp.yo.get(i)+"...");  
 }\*/* }  
  
  
  
 **public double** getDerivatives(**double** temp){  
 **double** result;  
 result = temp\*(1-temp);  
 **return** result;  
 }  
  
 **public void** getdelta\_o(BP bp){  
 **int** q = bp.**q**;  
 **double** temp1,temp2;  
 **for**(**int** i=0;i<q;i++){  
 temp1 =getDerivatives(bp.**yo**.get(i));  
 temp2 = -(bp.**d0**.get(i)-bp.**yo**.get(i))\*temp1;  
 bp.**delta\_o**.set(i,-temp2);  
 }  
 }  
  
 **public void** getdelta\_h(BP bp){  
 **int** q = bp.**q**;  
 **int** p = bp.**p**;  
 **double** temp1,sum=0;  
 **for**(**int** i=0;i<p;i++){  
 temp1 = getDerivatives(bp.**ho**.get(i));  
 **for**(**int** j=0;j<q;j++){  
 sum += bp.**delta\_o**.get(j)\*bp.**Who**.get(i\*q+j);  
 }  
 temp1 = temp1\*sum;  
 bp.**delta\_h**.set(i,temp1);  
 }  
 }  
  
 **public void** changeWho(BP bp){  
 **double** change\_value=0,temp;  
 **for**(**int** h=0;h<bp.**p**;h++){  
 **for**(**int** o=0;o<bp.**q**;o++){  
 change\_value = bp.**delta\_o**.get(o)\*bp.**ho**.get(h);  
 temp = bp.**Who**.get(h\*bp.**q**+o);  
 bp.**Who**.set(h\*bp.**q**+o,temp+bp.**rate**\*change\_value);  
 *// System.out.print("delta\_o: "+bp.delta\_o.get(o)+" ho: "+bp.ho.get(h)+" 改变值："+change\_value+"\n");* }  
 }  
 }  
  
 **public void** changeWih(BP bp){  
 **double** change\_value=0,temp;  
 **for**(**int** h=0;h<bp.**p**;h++){  
 **for**(**int** i=0;i<bp.**n**;i++){  
 change\_value = bp.**delta\_h**.get(h)\*bp.**X**.get(h);  
 temp = bp.**Wih**.get(h+i\***p**);  
 bp.**Wih**.set(h+i\***p**,temp+bp.**rate**\*change\_value);  
 }  
 }  
 }  
  
 **public void** getLastE(BP bp){  
 **double** sum=0;  
 **for**(**int** i=0;i<bp.**m**;i++){  
 sum+=2\*bp.**e**.get(i);  
 }  
  
 bp.**E** = 1.0/(2\*bp.**m**)\*sum;  
 }  
  
 **public void** printdata(BP bp){  
 System.***out***.println(**"\n\n"**+**"第"**+bp.**Now\_Times**+**"次"**);  
  
 System.***out***.println(**"\n输入"**);  
 **for**(**int** i=0;i<bp.**n**;i++){  
 System.***out***.print(**" "**+bp.**X**.get(i)+**"..."**);  
 }  
  
 System.***out***.println(**"\n实际输出"**);  
 **for**(**int** i=0;i<bp.**q**;i++){  
 System.***out***.print(**" "**+bp.**yo**.get(i)+**"..."**);  
 }  
  
 */\*System.out.println("\nWHO:::");  
 for(int i=0;i<bp.p;i++){  
 for(int j=0;j<bp.q;j++)  
 System.out.print(" "+bp.Who.get(i\*bp.q+j)+"...");  
 System.out.print("\n");  
 }\*/  
  
 /\*System.out.println("\n期望输出");  
 for(int i=0;i<bp.q;i++){  
 System.out.print(" "+bp.d0.get(i)+"...");  
 }\*/  
  
 /\*System.out.println("\n误差:~~");  
 for(int i=0;i<bp.m;i++){  
 System.out.print(" "+bp.e.get(i)+"...");  
 }\*/* System.***out***.println(**"\n\n误差值："**+bp.**E**);  
 }  
  
  
 **public static void** main(String[] args){  
 BP bp = **new** BP();  
 bp.Init(bp);  
 bp.setW(bp);  
 **for**(bp.**Now\_Times**=1;bp.**Now\_Times**<=bp.**Max\_Learn\_Times**&&bp.**E**>bp.**Precision\_Value**;bp.**Now\_Times**++){  
 *//产生随机样本* bp.setX(bp,bp.**Now\_Times**);  
 bp.setd0(bp,bp.**Now\_Times**);  
 bp.getHi(bp);  
 bp.getHo(bp);  
 bp.getYi(bp);  
 bp.getYO(bp);  
 bp.getdelta\_o(bp);  
 bp.getdelta\_h(bp);  
 bp.changeWih(bp);  
 bp.changeWho(bp);  
 bp.getE(bp);  
 bp.getLastE(bp);  
 bp.printdata(bp);  
 }  
 }  
}

## 3.5 数据来源

数据来源是老师提供的失效错误数据SYS1(failue\_count).txt，我已放入了java工程目录下。

## 3.6 结果展示

