附录

支撑材料列表

文件夹：题目1结果（包含21个组别的回归程序及运行结果）、神经网络与遗传算法（源代码）、SPSS数据（内含数据表）

代码

因代码程序大体相同，第一题以$A1$组程序为例：

```
%A1_1
x=[250 275 300 325 350];
y=[2.07 5.85 14.97 19.68 36.80 ];
m1=polyfit(x,y,2);
n1=248:1:370;
scatter(x,y);
hold on;
plot(n1,polyval(m1,n1));
xlabel('温度T');
ylabel('乙醇转化率');
title('A1乙醇转化率');
x=[250 275 300 325 350;
    1 1 1 1 1];
[b,bint,r,rint,stats] = regress(y',x');
x=[250 275 300 325 350;
    250*250 275*275 300*300 325*325 350*350;
    1 1 1 1 1];
[b1,bint1,r1,rint1,stats1] = regress(y',x');
rcoplot(r,rint);
%set(gca,'Color','w');
%rcoplot(r1,rint1);
%set(gca,'Color','w');
```

接下来是神经网络与遗传算法的源代码（本程序利用了谢菲尔德大学的遗传算法工具包）：

```
%train.m(以本程序为例，共四个相似程序)
%神经网络构建
s1=[-10000 -10000 -10000 -10000 -10000];%数据最大值储存
s2=[-10000 -10000];
s3=[10000 10000 10000 10000 10000];%数据最小值储存
s4=[10000 10000];
trainIn = begin';%神经网络输入
trainOut =begin1'.*begin2';%神经网络输出

nhidden = 80; % 隐藏层神经元个数
learningRate = 0.01; % 学习率
times = 30000; % 训练次数
checkInterval = 100;%时间间隔

nin = length(trainIn(:, 1));
nout = length(trainOut(:, 1));
trainSize = length(trainIn(1, :));
```

```
for i=1:nin
    for j=1:trainSize
        if(trainIn(i,j)>s1(i))
            s1(i)=trainIn(i,j);
        end
        if(trainIn(i,j)<s3(i))
            s3(i)=trainIn(i,j);
        end
    end
    for j=1:trainSize
        trainIn(i,j)=-1+2*(trainIn(i,j)-s3(i))/(s1(i)-s3(i));
    end
end
%输入归一化
for i=1:nout
    for j=1:trainSize
        if(trainOut(i,j)>s2(i))
            s2(i)=trainOut(i,j);
        end
        if(trainOut(i,j)<s4(i))
            s4(i)=trainOut(i,j);
        end
    end
    for j=1:trainSize
        trainOut(i,j)=-1+2*(trainOut(i,j)-s4(i))/(s2(i)-s4(i));
    end
end
%输出归一化
l1 = layer(nin, nhidden);
l2 = layer(nhidden, nout);
%控制层
% x,y记录训练
if nin == 1 && nout == 1
    x = [];
    y = [];
end

% 开始训练
for i = 1 : times
    % 误差
    loss = 0;
    tloss = 0;

    for j = 1:trainSize
        % 推进
        tempHidden = l1.run(trainIn(:, j));
        tempOut = l2.run(sigmoid(tempHidden));
        loss = sumsqr(tempOut - trainOut(:, j)); % 误差计算
        tloss = tloss + loss;

        % 反馈
        a = 2 * (tempOut - trainOut(:, j));
        l2.b = l2.b - learningRate * a;
        l2.w = l2.w - learningRate * (sigmoid(tempHidden)' .* a);
        b = (dsigmoid(tempHidden) .* (l2.w' * a));
        l1.b = l1.b - learningRate * b;
        l1.w = l1.w - learningRate * (trainIn(:, j)' .* b);
    end
    loss = tloss / (trainSize); % 'loss' is the average loss of the training set

    % 验证训练结果
    if mod(i, checkInterval) == 0
        if nin == 1 && nout == 1
            % record the present training result
            x = [x, trainIn'];
            y = [y, trainOut'];
            for k = 1 : length(x(:, 1))
                y(k, end) = BPrun(l1, l2, x(k, end));
            end
        end

        % 打印
        fprintf(['Training Steps: ', num2str(i), '/', num2str(times), ';\t',
'loss: ', num2str(loss), '\n']);
    end

    % 误差足够小结束
    if loss < 0.00001
        break
    end
end

%遗传算法寻找最优解
opt_minmax=1;   %优化目标类型：1最大化 0最小化
num_ppu=60;     %种群规模，个体个数。
num_gen=100;    %最大遗传代数
num_v=5;        %变量个数
len_ch=20;      %基因长度
gap=0.9;        %代沟
sub=-1;        %变量取值下限
up=1;          %变量取值上限
cd_gray=1;      %是否选择格雷码编码方式 1是，0否
sc_log=0;           %是否选择对数标度：1是，0否
trace=zeros(num_gen,2);     %遗传迭代性能跟踪器
%区域描述器，rep为矩阵复制函数
fieldd=[repmat([len_ch],[1,num_v]);repmat([sub;up],[1,num_v]);repmat([1-
cd_gray;sc_log;1;1],[1,num_v])];
chrom=crtbp(num_ppu,len_ch*num_v);  %初始化生产种群
k_gen=0;
x=bs2rv(chrom,fieldd);      %翻译初始化种群为10进制
fun_v=BPrun(l1,l2,x');  %计算目标函数值
fun_v=fun_v';
while k_gen<num_gen
    fit_v=ranking(-opt_minmax*fun_v);      %计算目标函数适应度
    selchrom=select('rws',chrom,fit_v,gap);     %使用轮盘赌方式选择
    selchrom=recombin('xovsp',selchrom);    %交叉
    selchtom=mut(selchrom);                         %变异
    x=bs2rv(selchrom,fieldd);                       %子代个体翻译
    fun_v_sel=BPrun(l1,l2,x');              %计算子代个体对应目标函数值
    fun_v_sel=fun_v_sel';
    fit_v_sel=ranking(-opt_minmax*fun_v_sel);

[chrom,fun_v]=reins(chrom,selchrom,1,1,opt_minmax*fun_v,opt_minmax*fun_v_sel);
%根据目标函数值将子代个体插入新种群
    [f,id]=max(fun_v);                              %寻找当前种群最优解
    x=bs2rv(chrom(id,:),fieldd);
    f=f*opt_minmax;
    fun_v=fun_v*opt_minmax;
    k_gen=k_gen+1;
    trace(k_gen,1)=f;
    trace(k_gen,2)=mean(fun_v);
end
fprintf(['温度: ', num2str((x(1)+1)/2*(s1(1)-s3(1))+s3(1)), '；催化剂质量：', num2str((x(2)+1)/2*(s1(2)-s3(2))+s3(2)), ';催化剂比例', num2str((x(3)+1)/2*(s3(3)-s1(3))+s1(3)),'co浓度: ',num2str((x(4)+1)/2*(s3(4)-s1(4))+s1(4)),'乙醇流速',num2str((x(5)+1)/2*(s3(5)-s1(5))+s1(5)) ,'\n','烯烃收率',num2str((f+1)/2*(s2(1)-s4(1))+s4(1))]);
```

```
%ds.m
function y = dsigmoid(x)
% 双极性传递
y = sigmoid(x) .* (1 - sigmoid(x));
end
```

```
%layer.m
classdef layer
    %  层的类

    properties
        nin %输入向量大小
        nout % 输出向量大小
        w % 权衡向量
        b % 偏差矢量
    end

    methods
        function obj = layer(nin_, nout_)
            obj.nin = nin_;
            obj.nout = nout_;
            obj.w = - ones(nout_, nin_) + 2 * rand(nout_, nin_);
            obj.b = - ones(nout_, 1) + rand(nout_, 1);
        end

        function vout = run(obj, vin)
            vout = obj.w * vin + obj.b;
        end
    end
end
```

```
%run.m
function vout_ = BPrun(lin_, lout_, vin_)
% 输出函数
vout_ = lout_.run(sigmoid(lin_.run(vin_)));
end
```

```matlab
%s.m
function y = sigmoid(x)
% 单极性s函数
y = 1 ./ (1 + exp(-x));
end
```