**《神经网络导论》实验报告**

-Adaline**的**LMS**算法**

专 业：控制科学与技术

班 级： 7060班

学 号： 3117041022

姓 名： 赵宇轩

《神经网络导论》 实验一

Adaline的LMS算法

# 一、实验目的

1. 通过实验了解Adaline的工作原理
2. 对比LMS的三种算法，并通过上机实验掌握具体实现方法
3. 与采用硬限幅函数的单个神经元模型进行对比，比较其异同

# 二、实验原理

采用硬限幅函数的单个神经元，通过简单的学习算法，可以成功实现两类线性可分类的分类功能。但对于大多数的非线性可分类来说，则无法完成分类功能，为此我们转而采用具有线性功能函数的神经元Adaline（Adaptive Linear Element）方法。

设输入矢量X=[]，加权矢量W=[]，则神经元的输出可以通过下式来计算：

(1)

要实现Adaline的分类功能，按照最小二乘法的统计意义而言，就是要求所有样本的实际输出值与理想预期值之间的误差的均方值最小。设输入观察矢量**X**的期望输出是d，当权向量为**W**时的实际输出是，定义。考虑所有可能出现样本的均方误差：

(2)

将(1)式代入，可得：

(3)

其中，是输入向量的自相关矩阵，是输入向量与期望输出的互相关向量。由(3)式可知必定存在最佳的加权矢量使均方误差达到最小，对(3)式求梯度可得：

(4)

由(4)式可解最优权向量：

(5)

(5)式给出了求最佳加权矢量的方法，但是需要做大量的统计计算，而且当输入矢量***X***的维数很大时，需要解决高阶矩阵求逆的问题，这些都是非常困难的。于是我们给出下面三种递推求解的方法。

## 2.1 LMS学习问题的严格递推学习算法

1. 任意设置初始权向量W(0);

2. 对于每一个时序变量k，按下式调整权向量W：

(6)

(6)式的含义为，应该向梯度的负方向调整加权向量W(k)，只要选定合适的步幅系数就能保证学习的收敛性。

求出(6)式中的梯度：

(7)

于是(6)式变为：

(8)

用这种方法可以保证求得严格的最佳解，而且避开了矩阵求逆的困难，但学习过程中的每一步仍需完成大量的统计计算，统计计算的困难尚需解决。

## 2.2 LMS学习问题的随机逼近算法

将(8)是修正为如下形式：

(9)

即得到随机逼近算法，与其严格递推算法的区别在于：用代替，由此避免了统计计算的困难，但同时也给加权矢量的变化趋势带来了随机性。

## 2.3 LMS学习问题的基于统计的算法

这是一种具有一定统计特性的学习算法

假设学习进行到第k步时，可能出现的样本有P个，分别用表示，下标p=1,2,…P表示在第k步学习过程中可能出现的不同样本编号。第p个样本的理想输出和实际输出分别用表示。我们定义“误差平方和”J(k)如下：

(10)

相对于W的梯度为：

(11)

令误差朝J(k)减小的方向调整，可得如下递推算法：

(12)

当P的数量非常大，使上式右端的求和项足以代表输入的统计特性时，(12)式与(8)式(严格递推算法)一致，当P取1时，(12)式与(9)式(随机逼近算法)一致。

# 三、实验内容及步骤

## 3.1 LMS算法

根据实验原理，即输入矢量的自相关阵，可得：



是输入向量与期望输出的互相关向量，可得：



最佳权向量，则有：



档W=时，均方误差取得最小值：可得：

=0.26233

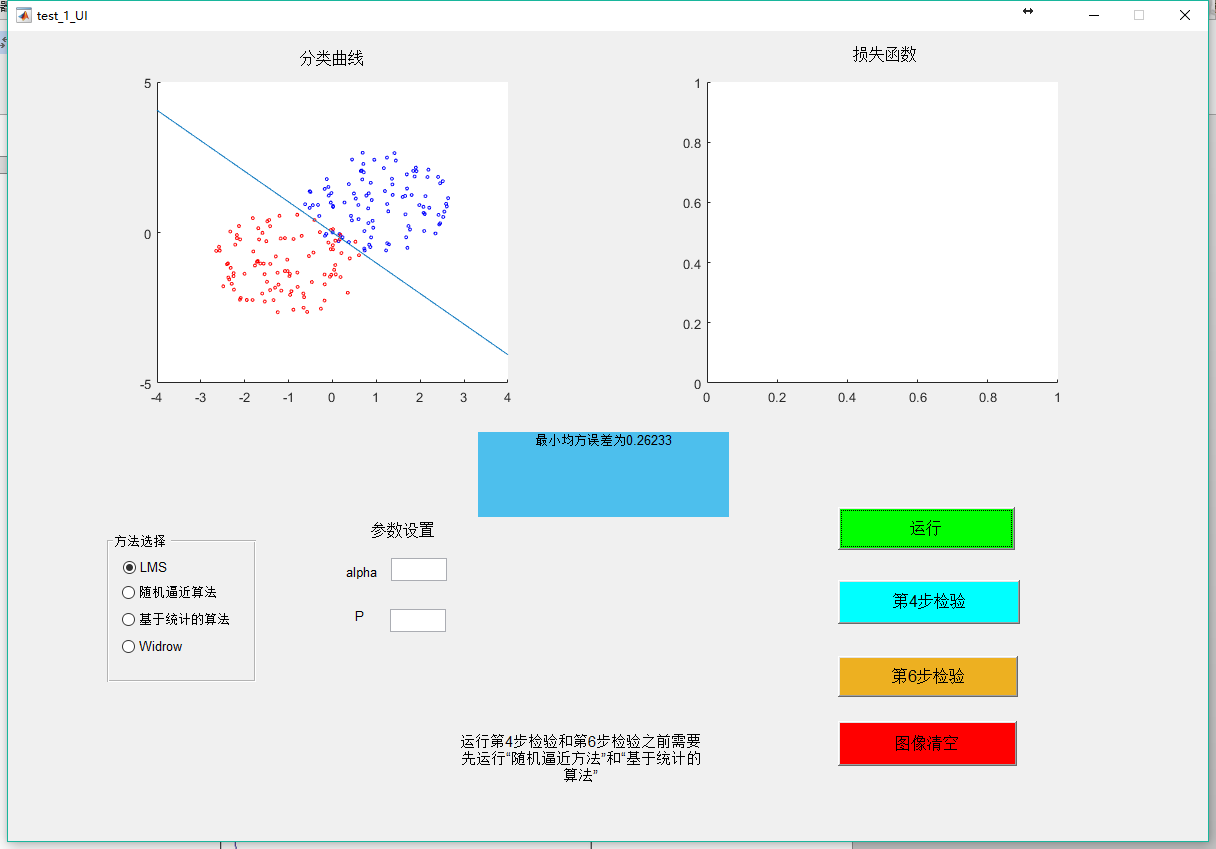


Figure LMS算法结果

## 3.2 随机逼近算法

随机逼近算法，根据给出的权向量初始值，每步从样本中随机的选择一个，按照迭代公式(9)，计算下一步的权向量，根据计算出的权向量，计算此时系统的输出矢量Y，与理想输出比较，得到此时系统的均方误差，若均方误差满足要求，则迭代结束，否则再选择样本进行训练。

下图为随机逼近算法的简单结构框图：



Figure 随机逼近算法流程图

其中，步幅系数，加权系数的初始值选择为W(1)=random(1,2)，学习结束的条件为随机逼近算法的均方误差

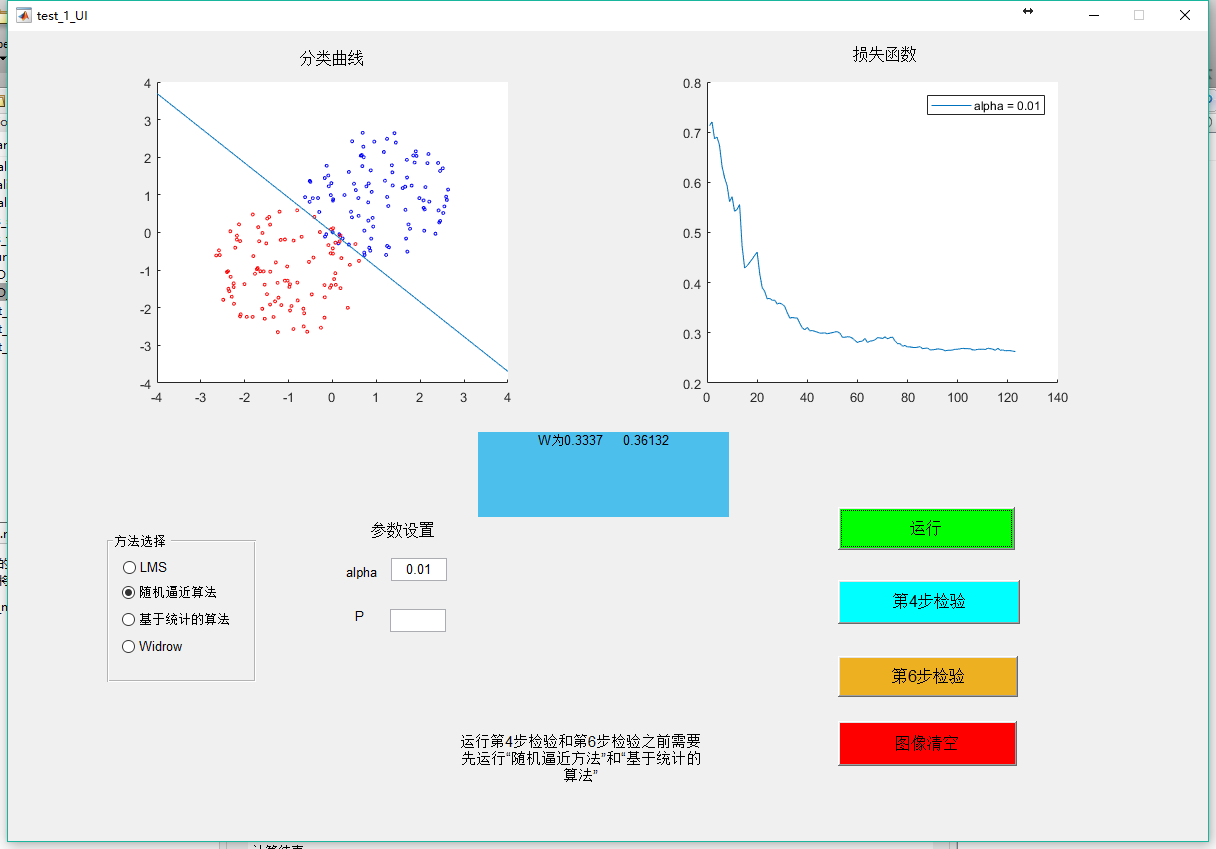
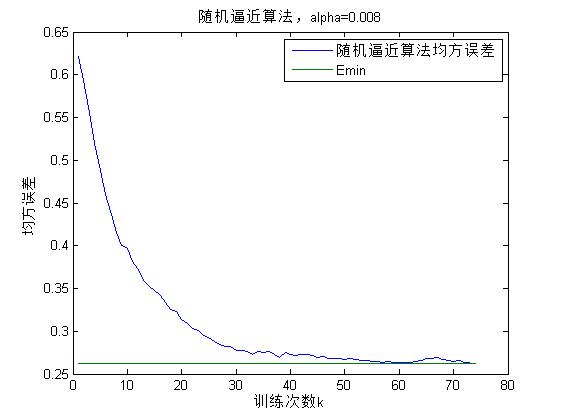


Figure alpha=0.01时的结果

迭代结束后，随机逼近算法计算出的加权系数矩阵：W=[0.3337,0.36132]

如图1所示为实验结果。在时，均方误差随训练次数的变化曲线，随着学习的进行，均方误差逐渐减小。但是从图中可以看见会有微小的起伏，这是因为每一步加权系数的调整量是向所选样本的误差梯度的负方向调整，但是总的趋势是误差减小的方向，最后满足误差的要求。



下列各图为取值不同时，均方误差随训练次数变化的曲线。

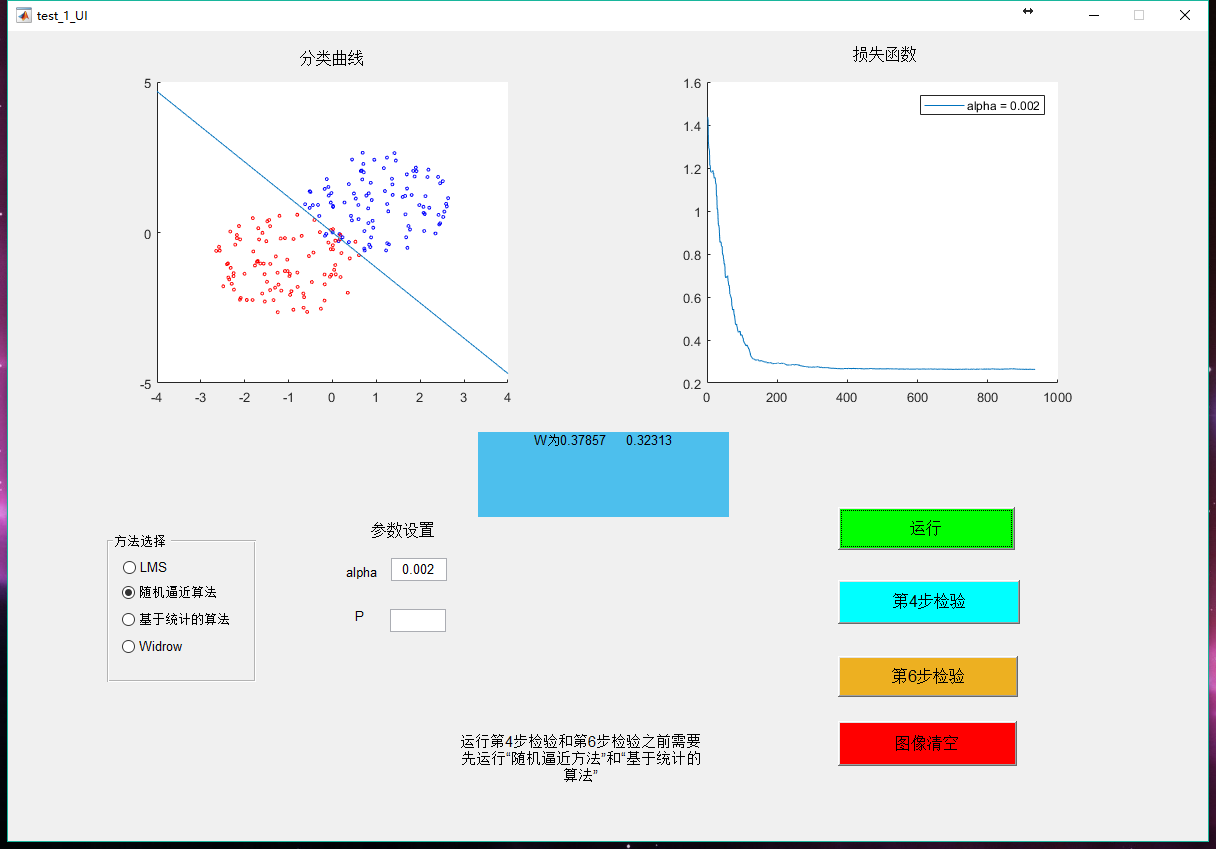


Figure alpha=0.02时的结果

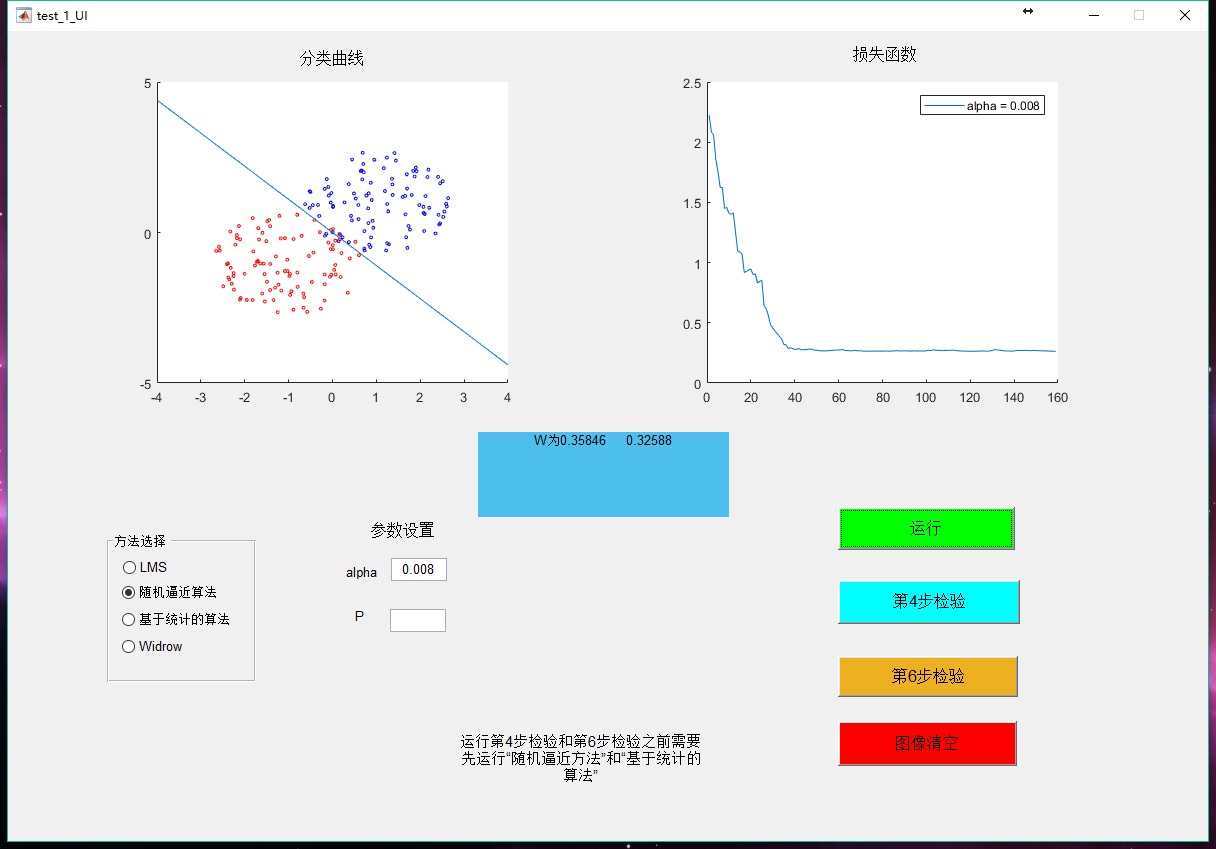


Figure alpha=0.08时的结果

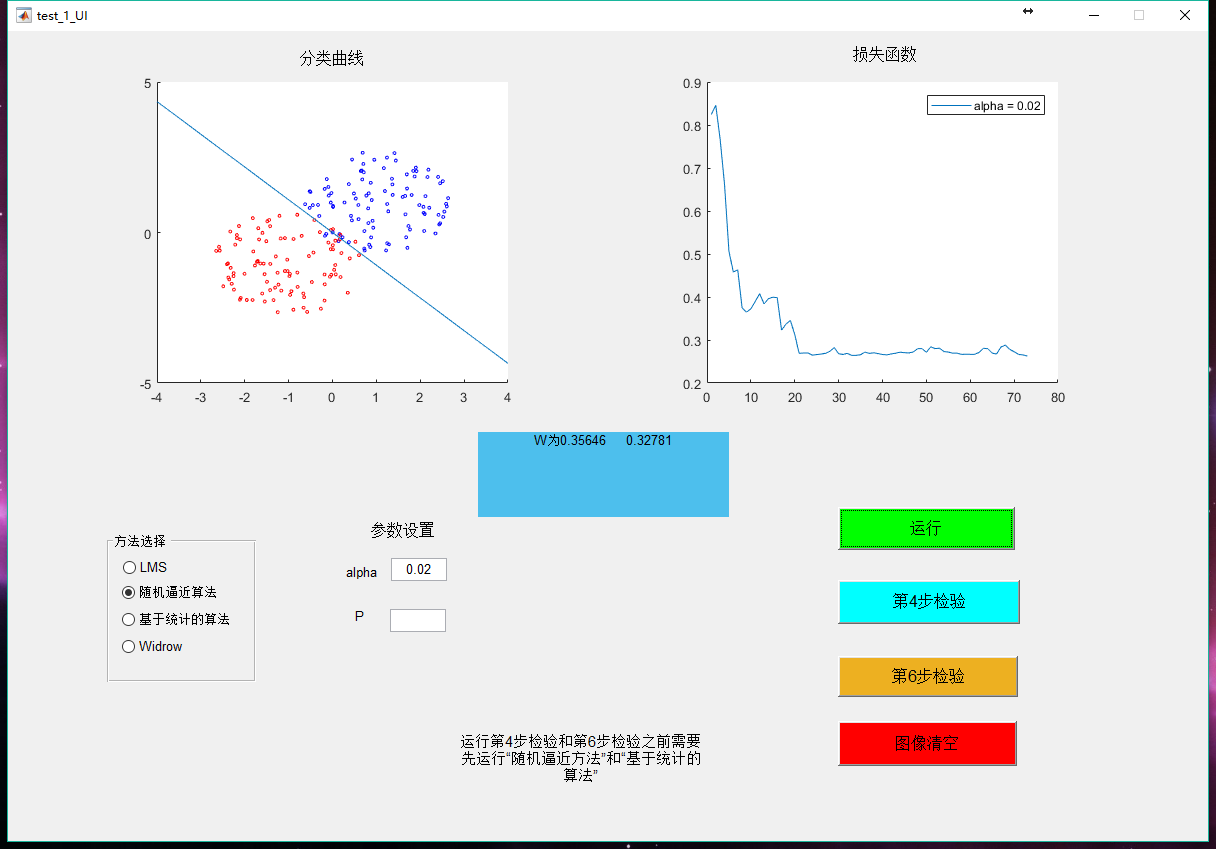


Figure alpha=0.02时的结果

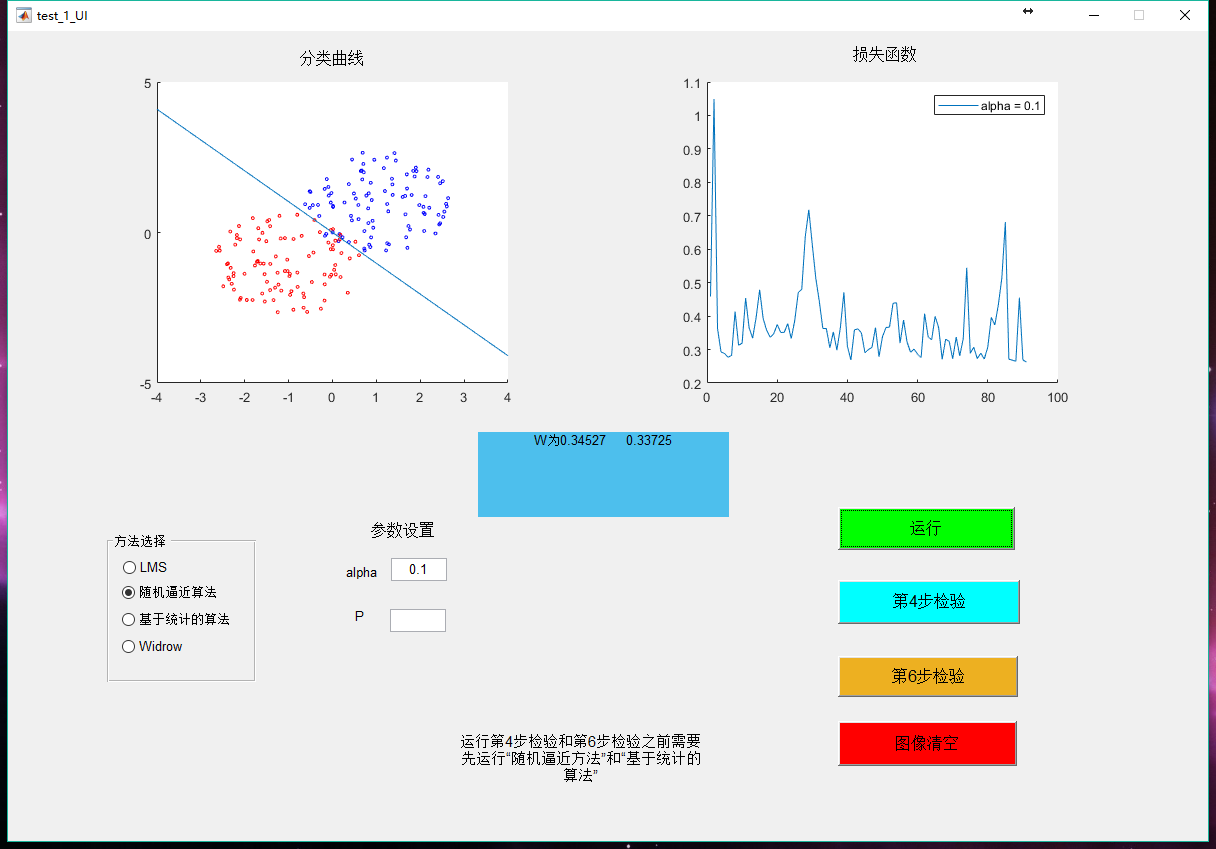


Figure alpha=0.1时的结果

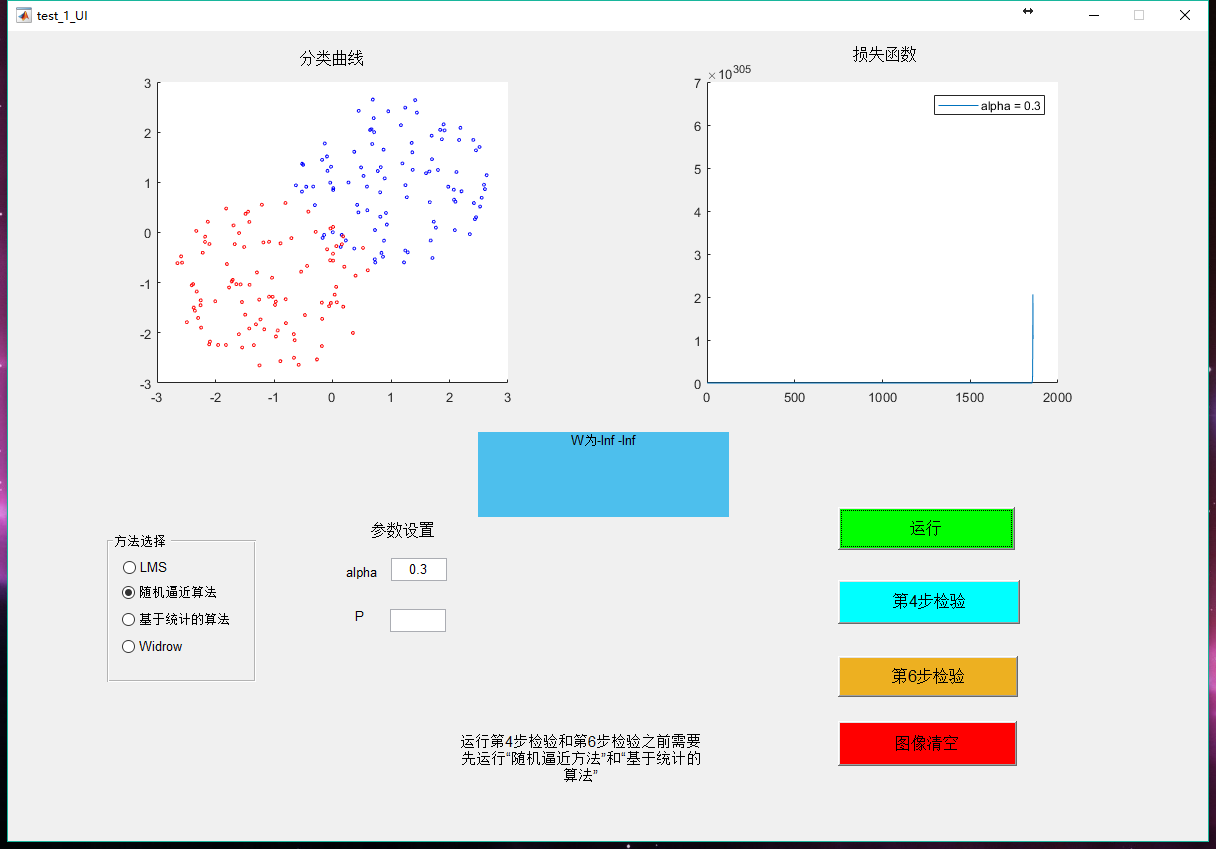


Figure alpha=0.3时的结果

从图中可以看出，在时，学习均是收敛的，只是对于不同的步幅系数，收敛速度不同。在时收敛最快，在时收敛较慢，但是收敛的方向非常稳定，震荡很小，当时，学习是震荡的，但是也会达到最终的优化目标，当时，学习是发散的，最终W取值到了无穷，无法达到收敛目标。

原因：步幅系数影响每次对于加权系数W的调整量，因此，在步幅系数较小时（例），每次学习对于加权系数的调整很小，因此训练的次数较多，损失函数降低较为平稳，会收敛。在步幅系数较大时（例），每次学习对于加权系数的调整也会较大，所以，损失函数总是在最优值的附近震荡，当很大，如0.3时，已经完全无法收敛，每次的该变量都在越来越大，最终达到了无穷值。

## 3.3 基于统计的算法

基于统计的算法，根据给出的权向量初始值，每步随机的选择几个样本，（在本次实验中，随机的选择5个样本）按照迭代公式(12)，计算下一步的权向量，根据计算出的权向量，计算出此时系统的输出矢量Y，与理想输出比较，得到此时的均方误差，若均方误差满足要求，则迭代结束，否则再选择样本进行训练。

下图为基于统计算法的简单结构框图：



Figure 基于统计的算法流程图

其中，步幅系数，输入矢量样本P=5，加权系数的初始值选择为W(1)=random(1,2)，学习结束的条件为此算法的均方误差

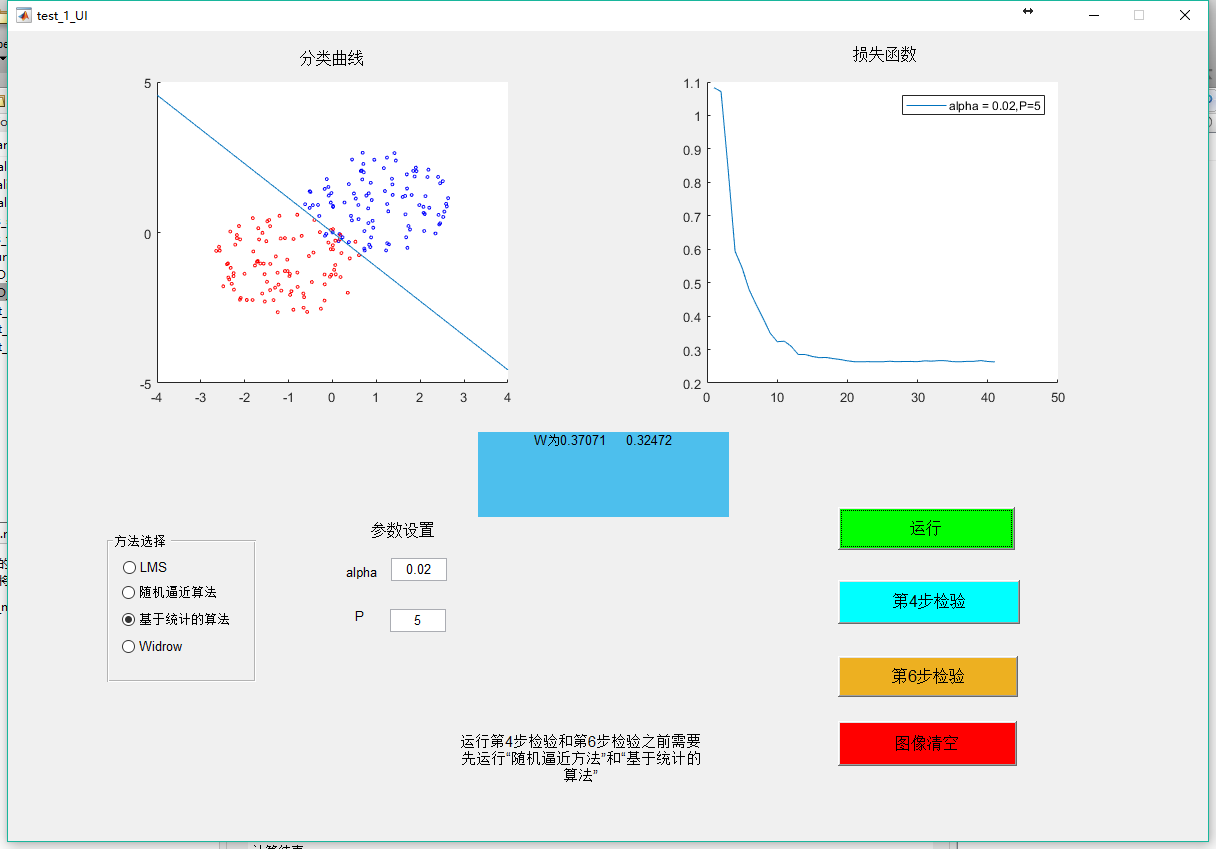


Figure alpha=0.02，P=5的结果

迭代结束后，基于统计的算法计算出的加权系数矩阵为：W=[0.37071，0.32472].

如图10所示为基于统计的算法的实验结果。在P=5时，随着学习的进行，均方误差逐渐减小。在整体的优化过程中，优化速度很快，并且基本没有震荡，只需要40多次的优化就能达到目标。

下列各图为P取值不同时，均方误差随训练次数变化的曲线。

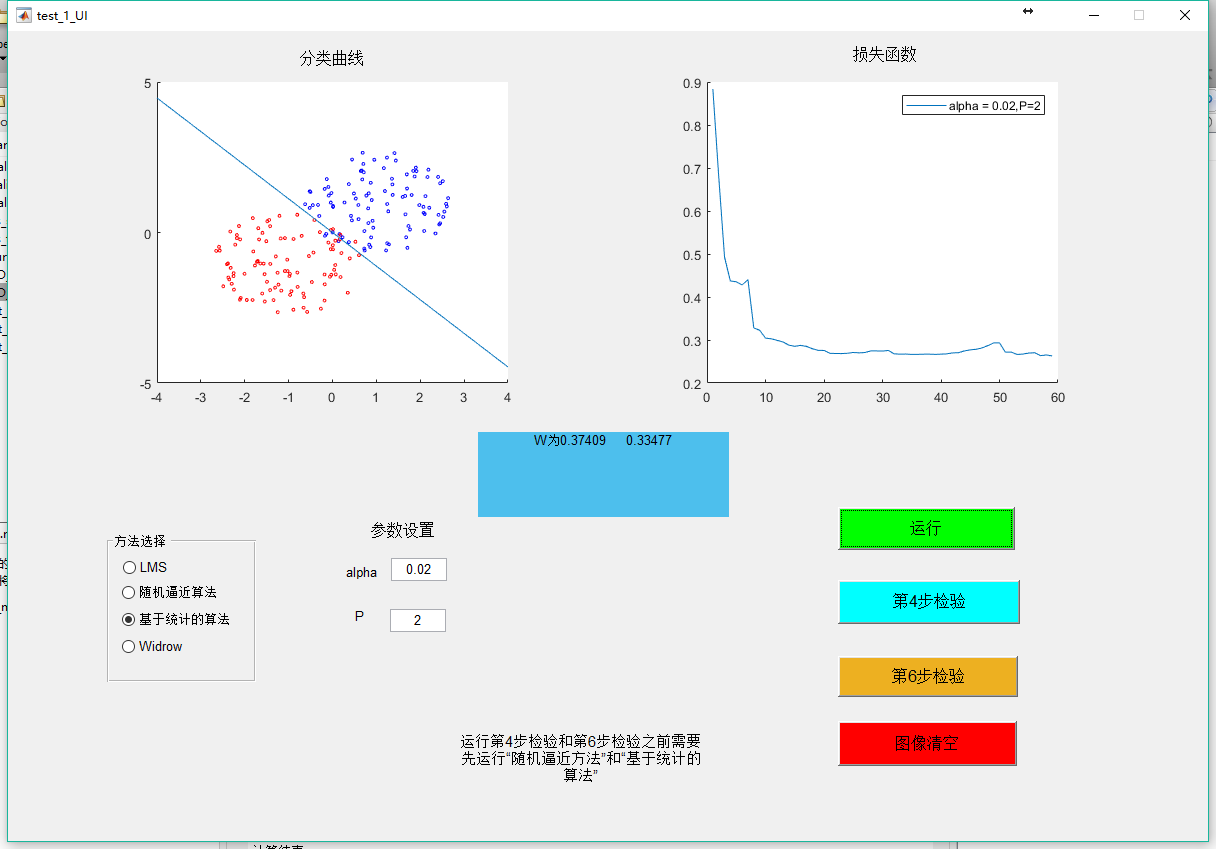


Figure alpha=0.02，P=2的结果

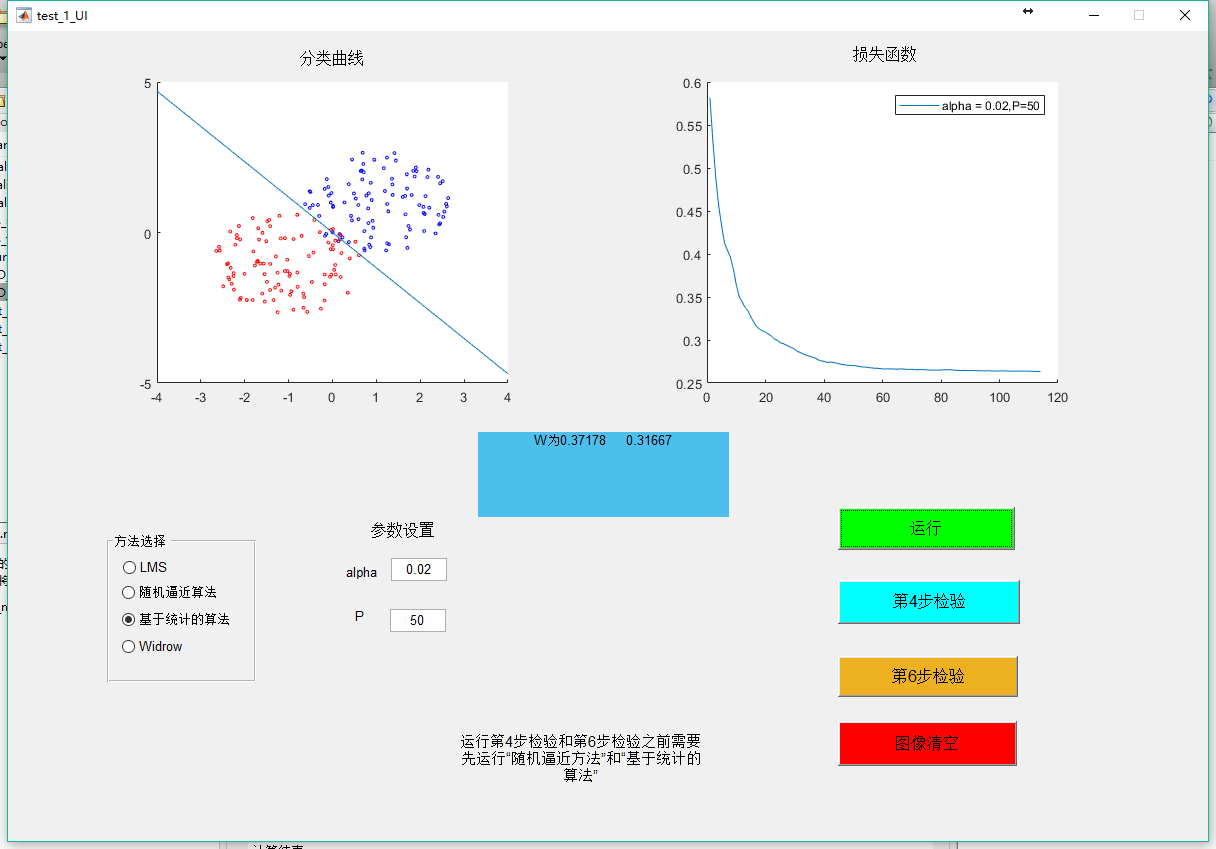


Figure alpha=0.02，P=50的结果

从图中可以看出在合适的步幅系数时，学习过程均是收敛的，随着P取值的不同，训练次数不同。P取值越大，每步调整的加权系数W越精确，同时，每步优化的值也变小，这是因为多个优化值取平均的缘故。比如在图12中，由于P=50，损失函数曲线就变得非常平稳，同时训练次数相对增加一些。

## 3.4 Widrow严格递推算法

Widrow严格递推算法，与前两种方法不同的是，每一步调整都是利用所有的样本，计算其理想输出与实际输出的误差，权向量向误差梯度的负方向调整。即每一步利用所有的样本计算调整量，根据给出的权向量初始值，按照迭代公式(8)，计算下一步的权向量，再计算此时系统的输出矢量Y，与理想输出比较，得到此时系统的均方误差，若均方误差满足要求，则迭代结束，否则选择样本进行训练。

下图为Widrow严格递推算法的简单结构框图：



Figure Widrow算法流程图

其中，步幅系数，加权系数的初始值选择为W(1)=random(2,1)，学习结束的条件为此算法的均方误差

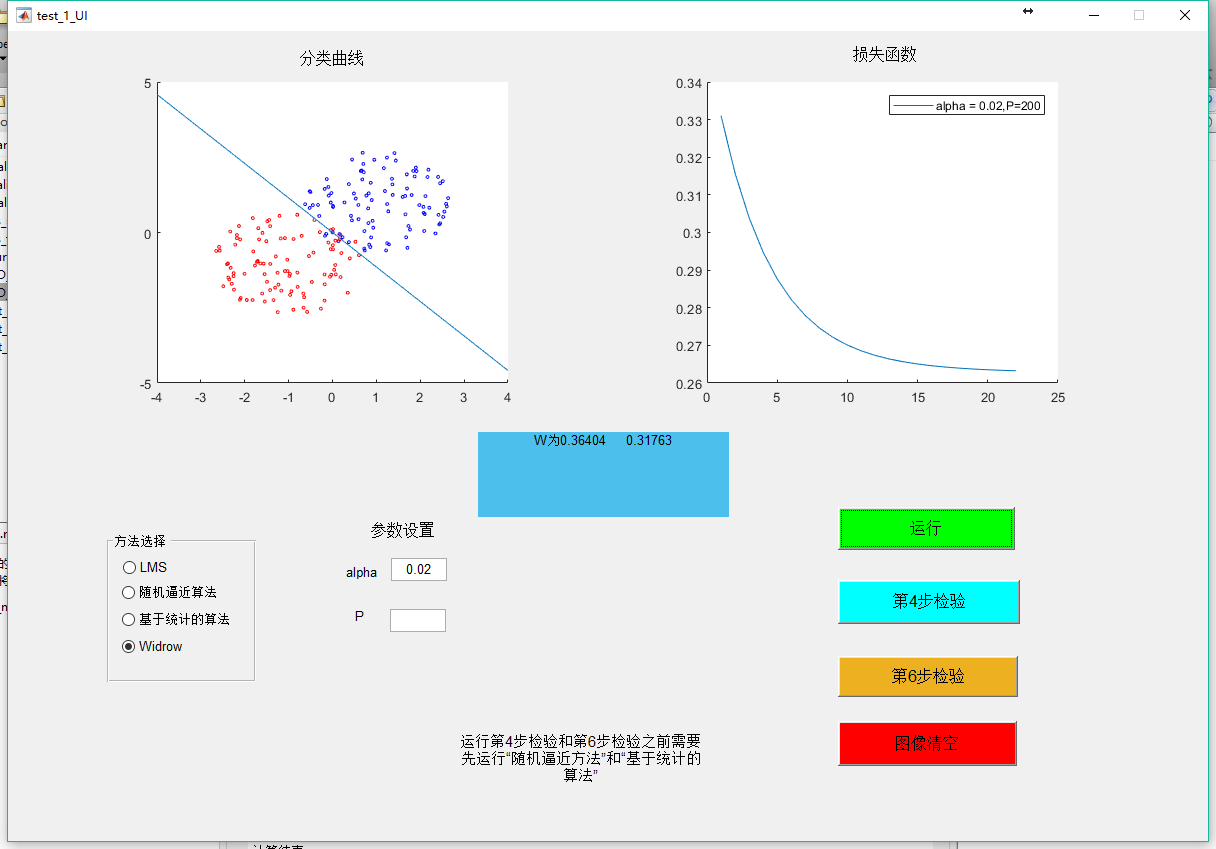


Figure alpha=0.02时的结果

迭代结束后，Widrow严格递推算法计算出的加权系数矩阵：W=[0.32953，0.37572]

如图14所示，为时，Widrow的严格递推算法计算的系统的均方误差随训练次数的变化情况，从图中可以看见，因为此算法每次对于加权系数的调整是利用所有的输入矢量，即向系统的均方误差减小的方向调整，因此在整体的优化过程中，很少出现震荡的情况，训练次数也较少。

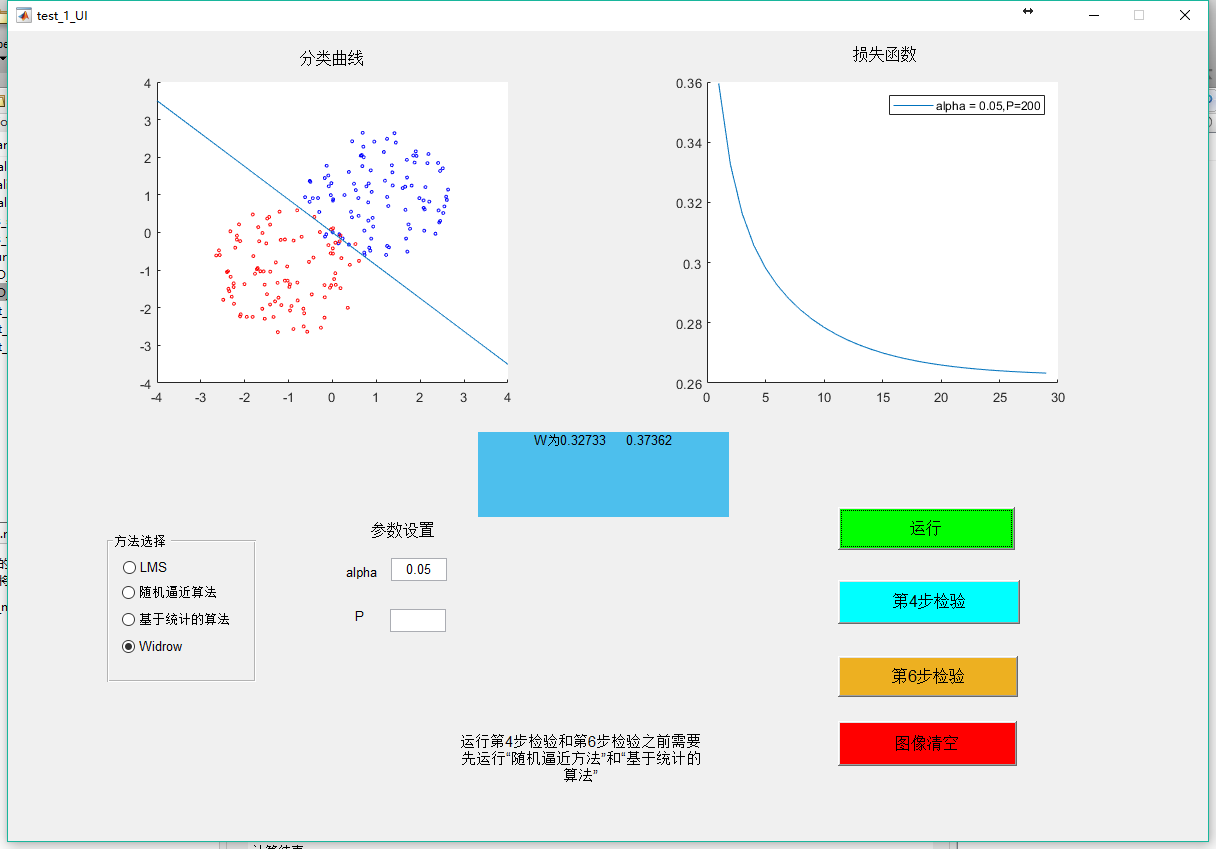


Figure alpha=0.05时的结果

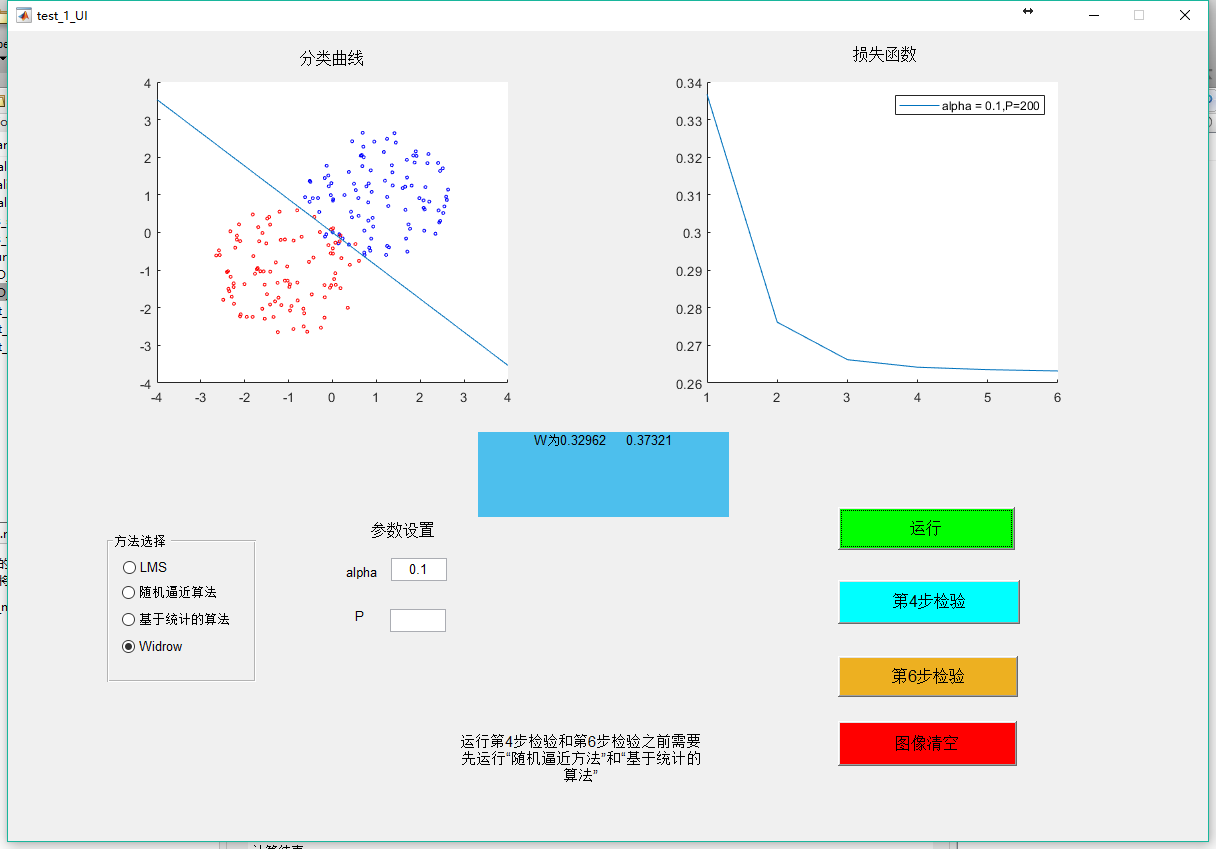


Figure alpha=0.1时的结果

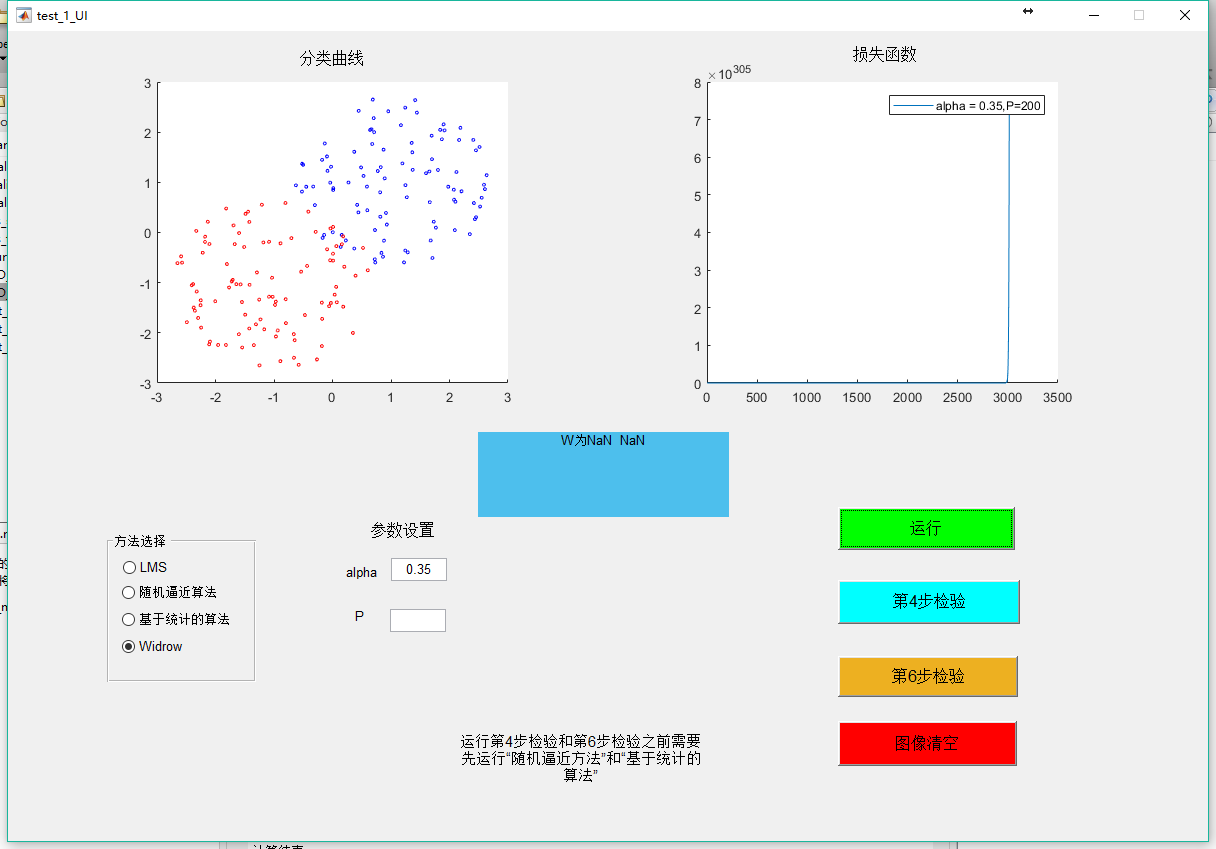


Figure alpha=0.35时的结果

如上图所示为取不同的值的时候，系统均方误差随训练次数的变化曲线，步幅系数影响每一次加权系数的调整量。从图中可以看见当取0.02，0.05，0.1，时，迭代均是收敛的，而且越大，收敛的越快，训练次数越少，这时，就可以将调整得相对大一些。取0.35时，无法收敛，因此不能把取值过大。

## 3.5 检验

本实验在检验时，对于用于测试的200个样本，按照三种方法计算出的加权系数分别计算其对应的实际输出，再与理想输出对比，确定分类是否正确。

下面为三种方法分类的结果

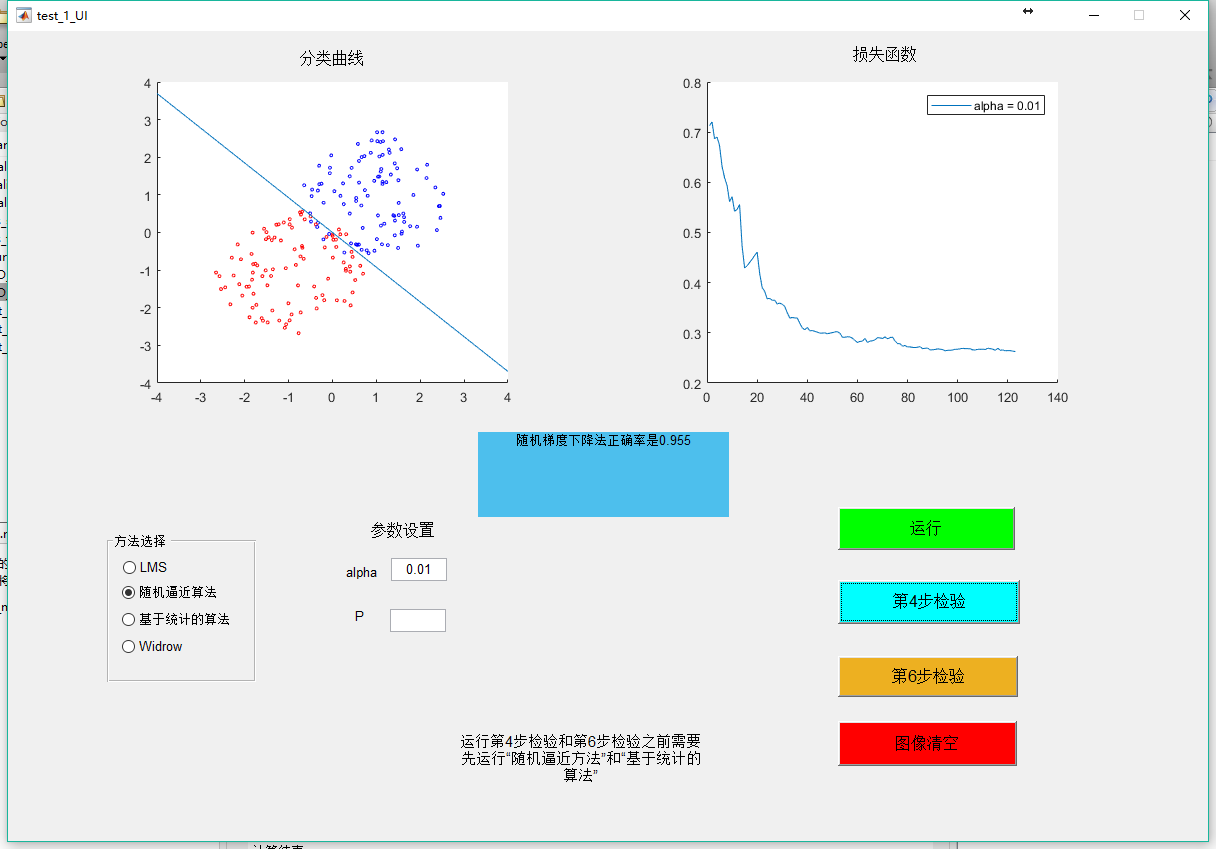


Figure 随机逼近算法的检验

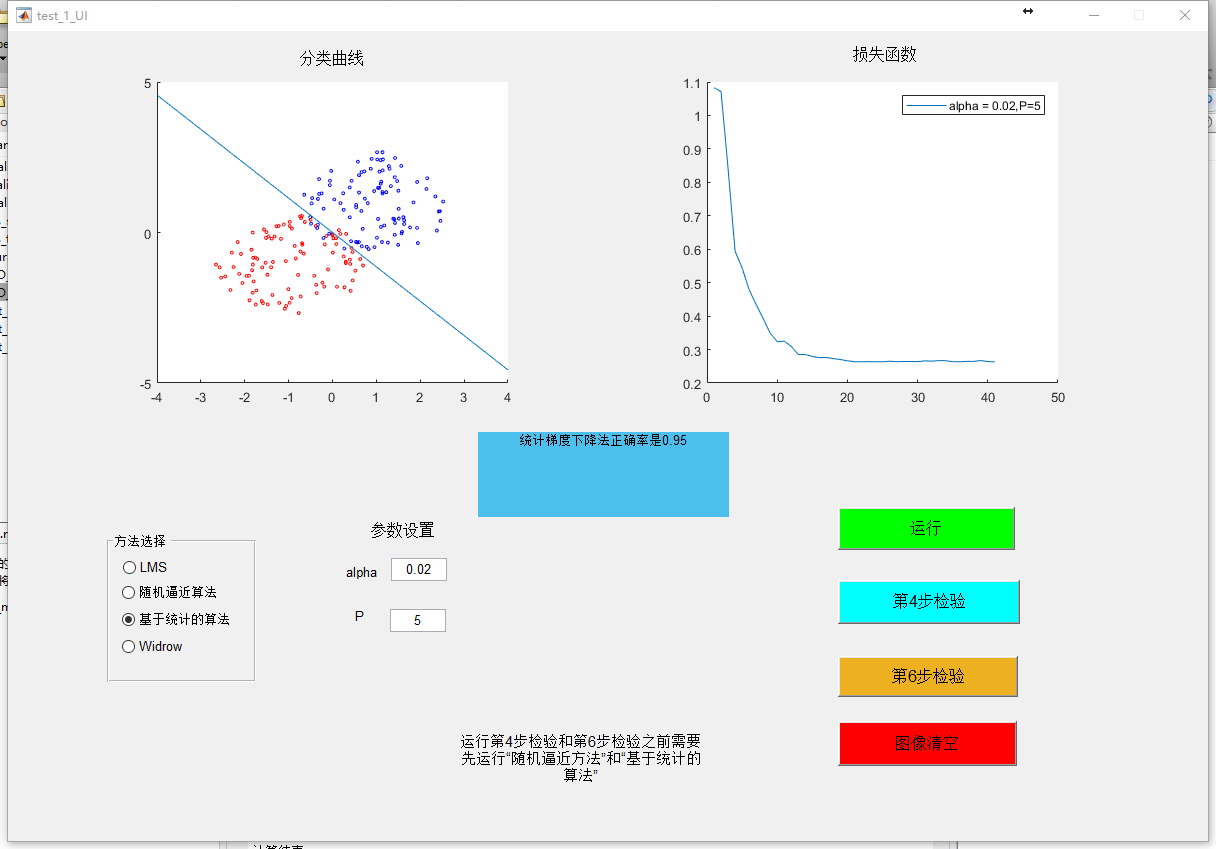


Figure 基于统计算法的检验

表格 1 不同算法的正确率比较

|  |  |  |  |
| --- | --- | --- | --- |
| 算法 | 加权系数 | 错误个数 | 正确率（%） |
| 随机逼近算法 | W=[0.3337,0.36132] | 9 | 95.55 |
| 基于统计的算法 | W=[0.37071，0.32472] | 10 | 95 |

不同算法效果基本相同。

# 四、实验总结与思考

## 4.1 实验总结

本次实验我对神经网络的Adaline的LMS算法有了基本的了解。

相比随机逼近算法，基于统计的算法能够在相同的情况下实现更加平稳的损失函数下降曲线，并且相比随机梯度下降，迭代次数更少，能够将调整的比较大。

另外，不同的优化方法最后得到的准确率基本相同。

## 4.2 实验思考题

1、如果想采用硬限幅函数的单个神经元完成该分类任务，会出现什么样的现象？

可能结果会不收敛，因为采用硬限幅函数的单个神经元只能完成线性可分类问题，但是对于非线性的分类问题可能结果会不收敛。

2、通过观察比较随机逼近算法与不同P时最陡下降算法的均方误差曲线随时序变量k的变化有什么不同，并简要解释原因。

随机逼近算法与基于统计的算法都是利用少量的样本进行加权系数的计算，每次调整都是向所选样本误差减小的方向调整，所以误差曲线都有微小的波动。但是基于统计的算法随着样本P的增加，其误差曲线趋近平滑，因为所选样本越多，越能反映总体的误差情况。

# 五、代码

## 5.1 二维输入的神经网络对象

classdef neural\_2

%nerual 有两个输入的神经元对象

% Detailed explanation goes here

properties

W

end

methods

function obj = neural\_2(Weight)

%neural\_2 Construct an instance of this class

% Detailed explanation goes here

obj.W = Weight;

end

function outputArg = goThrough(obj,X)

%goThrough 针对特定X产生神经元输出

% Detailed explanation goes here

outputArg = obj.W\*X';

end

function outputArg = goThrough\_th(obj,X)

%goThrough\_th 针对特定X产生神经元输出，并且加上了阈值限制

% Detailed explanation goes here

outputArg = obj.W\*X';

outputArg(outputArg>0)=1;

outputArg(outputArg<=0)=-1;

end

end

end

## 5.2 优化函数

本函数可用于基于统计的梯度下降算法，也可以是实现随机梯度下降算法和Widrow算法，只需要将P值传为1或样本个数。

function [W,Elog] = SGD\_mini(X,Y,alpha,Estop,P)

% 基于统计的梯度下降算法,如果想用随机梯度下降法，直接将P设置为1

E = 10000;

W = rand(1,size(X,2));%初始化随机权重

neural\_b = neural\_2(W);

Elog=[];

while(E>Estop)

if P==size(X,1) %Widrow

checkNO = 1:P;

else

checkNO = randi([1 size(X,1)],1,P);%获得P个随机抽样点

end

tmpSigma = 0;%W改变的求和中间变量

for checkIndex = checkNO

y = neural\_b.goThrough(X(checkIndex,:));

thisError = Y(checkIndex)-y;%计算误差

tmpSigma = tmpSigma + thisError \* X(checkIndex,:);%tmpSigma每次叠加

end

neural\_b.W = neural\_b.W + 2 \* alpha / P \* tmpSigma;

%测试所有样本误差

errorSum = 0;

for i=1:size(X,1)

y = neural\_b.goThrough(X(i,:));

errorSum = errorSum + (y-Y(i))^2;

end

E = errorSum/size(X,1);

Elog = [Elog,E];

end

W = neural\_b.W;

End

## 5.3 UI及主函数部分

function varargout = test\_1\_UI(varargin)

% TEST\_1\_UI MATLAB code for test\_1\_UI.fig

% TEST\_1\_UI, by itself, creates a new TEST\_1\_UI or raises the existing

% singleton\*.

%

% H = TEST\_1\_UI returns the handle to a new TEST\_1\_UI or the handle to

% the existing singleton\*.

%

% TEST\_1\_UI('CALLBACK',hObject,eventData,handles,...) calls the local

% function named CALLBACK in TEST\_1\_UI.M with the given input arguments.

%

% TEST\_1\_UI('Property','Value',...) creates a new TEST\_1\_UI or raises the

% existing singleton\*. Starting from the left, property value pairs are

% applied to the GUI before test\_1\_UI\_OpeningFcn gets called. An

% unrecognized property name or invalid value makes property application

% stop. All inputs are passed to test\_1\_UI\_OpeningFcn via varargin.

%

% \*See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one

% instance to run (singleton)".

%

% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help test\_1\_UI

% Last Modified by GUIDE v2.5 12-Dec-2017 16:02:34

% Begin initialization code - DO NOT EDIT

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @test\_1\_UI\_OpeningFcn, ...

'gui\_OutputFcn', @test\_1\_UI\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

% End initialization code - DO NOT EDIT

% --- Executes just before test\_1\_UI is made visible.

function test\_1\_UI\_OpeningFcn(hObject, eventdata, handles, varargin)

% This function has no output args, see OutputFcn.

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% varargin command line arguments to test\_1\_UI (see VARARGIN)

% Choose default command line output for test\_1\_UI

handles.output = hObject;

% Update handles structure

guidata(hObject, handles);

load('lms\_samp','samp');

load('lms\_tstsamp.mat','tstsamp');

global Emin samp tstsamp W\_SGD W\_SGD\_mini;

Emin = 0.2623;%提前声明Emin

W\_SGD = [1,1];%初始化

W\_SGD\_mini = [1,1];%初始化

% UIWAIT makes test\_1\_UI wait for user response (see UIRESUME)

% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.

function varargout = test\_1\_UI\_OutputFcn(hObject, eventdata, handles)

% varargout cell array for returning output args (see VARARGOUT);

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure

varargout{1} = handles.output;

function edit1\_Callback(hObject, eventdata, handles)

% hObject handle to edit1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text

% str2double(get(hObject,'String')) returns contents of edit1 as a double

% --- Executes during object creation, after setting all properties.

function edit1\_CreateFcn(hObject, eventdata, handles)

% hObject handle to edit1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit2\_Callback(hObject, eventdata, handles)

% hObject handle to edit2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text

% str2double(get(hObject,'String')) returns contents of edit2 as a double

% --- Executes during object creation, after setting all properties.

function edit2\_CreateFcn(hObject, eventdata, handles)

% hObject handle to edit2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

% --- Executes on button press in pushbutton2.

function pushbutton2\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

disp('请等待')

method = get(get(handles.uibuttongroup2,'SelectedObject'), 'String');

global Emin samp W\_SGD W\_SGD\_mini;

SColor = zeros(size(samp,1),3);

SColor(samp(:,3)==1,3) = 1;

SColor(samp(:,3)==-1,1) = 1;

switch method

case 'LMS'

%计算自相关矩阵R

sumR = [0,0;0,0];

for i=1:size(samp,1)

sumR = sumR + samp(i,1:2)'\*samp(i,1:2);

end

R = sumR/size(samp,1);

%计算互相关向量P

P = mean(samp(:,1:2).\*samp(:,3));

%计算最佳权值向量Wstar

Wstar = P\*R^-1;

%最小均方误差Emin

neural\_a = neural\_2(Wstar);

errorSum = 0;

for i=1:size(samp,1)

y = neural\_a.goThrough(samp(i,1:2));

errorSum = errorSum + (y-samp(i,3))^2;

end

Emin = errorSum/size(samp,1);

set(handles.text2,'String',['最小均方误差为',num2str(Emin)]);

scatter(handles.axes1,samp(:,1),samp(:,2),5,SColor);

line(handles.axes1,[-4,4],[4\*Wstar(1)/Wstar(2),-4\*Wstar(1)/Wstar(2)]);

case '随机逼近算法'

alpha = str2double(get(handles.edit1,'String'));

set(handles.text2,'String','');

if isnan(alpha)

set(handles.text2,'String','未输入alpha，默认值为0.01');

alpha = 0.01;

end

Estop = Emin+0.001;

[W\_SGD,Elog] = SGD\_mini(samp(:,1:2),samp(:,3),alpha,Estop,1);

set(handles.axes2,'NextPlot','add');

plot(handles.axes2,Elog,'DisplayName',['alpha = ',num2str(alpha)]);legend('show')

scatter(handles.axes1,samp(:,1),samp(:,2),5,SColor);

line(handles.axes1,[-4,4],[4\*W\_SGD(1)/W\_SGD(2),-4\*W\_SGD(1)/W\_SGD(2)]);

set(handles.text2,'String',[get(handles.text2,'String'),'W为',num2str(W\_SGD)]);

%kk

case '基于统计的算法'

alpha = str2double(get(handles.edit1,'String'));

set(handles.text2,'String','');

if isnan(alpha)

set(handles.text2,'String','未输入alpha，默认值为0.01');

alpha = 0.01;

else

set(handles.text2,'String','');

end

P = str2double(get(handles.edit2,'String'));

if isnan(P)

set(handles.text2,'String',[get(handles.text2,'String'),'未输入P，默认值为5']);

P = 5;

end

Estop = Emin+0.001;

[W\_SGD\_mini,Elog] = SGD\_mini(samp(:,1:2),samp(:,3),alpha,Estop,P);

set(handles.axes2,'NextPlot','add');

plot(handles.axes2,Elog,'DisplayName',['alpha = ',num2str(alpha),',P=',num2str(P)]);legend('show')

scatter(handles.axes1,samp(:,1),samp(:,2),5,SColor);

line(handles.axes1,[-4,4],[4\*W\_SGD\_mini(1)/W\_SGD\_mini(2),-4\*W\_SGD\_mini(1)/W\_SGD\_mini(2)]);

set(handles.text2,'String',[get(handles.text2,'String'),'W为',num2str(W\_SGD\_mini)]);

case 'Widrow'

alpha = str2double(get(handles.edit1,'String'));

set(handles.text2,'String','');

if isnan(alpha)

set(handles.text2,'String','未输入alpha，默认值为0.1');

alpha=0.1;

end

Estop = Emin+0.001;

P = size(samp,1);

[W\_SGD\_mini,Elog] = SGD\_mini(samp(:,1:2),samp(:,3),alpha,Estop,P);

set(handles.axes2,'NextPlot','add');

plot(handles.axes2,Elog,'DisplayName',['alpha = ',num2str(alpha),',P=',num2str(P)]);legend('show')

scatter(handles.axes1,samp(:,1),samp(:,2),5,SColor);

line(handles.axes1,[-4,4],[4\*W\_SGD\_mini(1)/W\_SGD\_mini(2),-4\*W\_SGD\_mini(1)/W\_SGD\_mini(2)]);

set(handles.text2,'String',[get(handles.text2,'String'),'W为',num2str(W\_SGD\_mini)]);

otherwise

error('方法选择出错')

end

disp('计算结束')

% --- Executes on button press in pushbutton2.

function pushbutton3\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global tstsamp W\_SGD;

SColor = zeros(size(tstsamp,1),3);

SColor(tstsamp(:,3)==1,3) = 1;

SColor(tstsamp(:,3)==-1,1) = 1;

scatter(handles.axes1,tstsamp(:,1),tstsamp(:,2),5,SColor);

line(handles.axes1,[-4,4],[4\*W\_SGD(1)/W\_SGD(2),-4\*W\_SGD(1)/W\_SGD(2)]);

neural\_SGD = neural\_2(W\_SGD);

Y\_SGD = neural\_SGD.goThrough\_th(tstsamp(:,1:2));

set(handles.text2,'String',['随机梯度下降法正确率是',num2str(1-size(find(tstsamp(:,3)'~=Y\_SGD),2)/size(tstsamp,1))]);

% --- Executes on button press in pushbutton2.

function pushbutton4\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global tstsamp W\_SGD\_mini;

SColor = zeros(size(tstsamp,1),3);

SColor(tstsamp(:,3)==1,3) = 1;

SColor(tstsamp(:,3)==-1,1) = 1;

scatter(handles.axes1,tstsamp(:,1),tstsamp(:,2),5,SColor);

line(handles.axes1,[-4,4],[4\*W\_SGD\_mini(1)/W\_SGD\_mini(2),-4\*W\_SGD\_mini(1)/W\_SGD\_mini(2)]);

neural\_SGD = neural\_2(W\_SGD\_mini);

Y\_SGD\_mini = neural\_SGD.goThrough\_th(tstsamp(:,1:2));

set(handles.text2,'String',['统计梯度下降法正确率是',num2str(1-size(find(tstsamp(:,3)'~=Y\_SGD\_mini),2)/size(tstsamp,1))]);

% --- Executes on button press in pushbutton4.

function pushbutton5\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton4 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

set(handles.text2,'String','');

cla(handles.axes1,'reset')

cla(handles.axes2,'reset')

%清除图像cla