哈爾濱工業大學(深圳)
HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

《模式识别》课程

实验报告

实验一：线性分类器的设计与实现

姓　　名：杨承翰

学　　号：210210226

班　　级：通信 2 班

指导教师：顾术实

哈尔滨工业大学（深圳）

实践环节评分及标准

考察项目	考察内容	教师评价	备注（评语）
A 预习（可选）100 分	预习认真，数据记录表格设计全面、准确（90-100 分）		
	预习较认真，数据记录表格设计较为全面（80-89 分）		
	预习深度不够，数据记录表格设计不全面（60-79 分）		
	未充分预习（60 分以下）		
B 过程表现（必选）100 分	态度认真、积极主动、操作规范，实验完毕后仪器断电、归位（90-100 分）		
	态度认真、积极主动、操作正确，实验完毕后仪器断电、归位（80-89 分）		
	态度较认真、操作基本正确，实验完毕后仪器断电、归位（60-79 分）		
	态度不端正、操作不规范，实验完毕后仪器未断电、归位（60 分以下）		
C 结果验收（必选）100 分	实验数据处理准确规范、结果合理、结论正确（90-100 分）		
	实验数据处理准确、结果合理、结论正确（80-89 分）		
	实验数据处理较准确、结果基本合理、结论基本正确（60-79 分）		
	有大量错误，未达到实验目的（60 分以下）		
D 答辩（可选）100 分	理解充分，表达清晰，问题回答全面准确（90-100 分）		
	理解较充分，表达清晰，问题回答正确（80-89 分）		
	理解较充分，表达基本清晰，问题回答基本正确（60-79 分）		
	表达模糊，问题回答不正确（60 分以下）		
E 实验报告内容（必选）100 分	格式规范；内容充实；分析深刻，见解独到（90-100 分）		
	格式规范；内容完整；分析正确，有个人见解（80-89 分）		
	格式较规范；内容较完整；分析基本正确（60-79 分）		
	格式不规范；内容不完整（60 分以下）		
综合评分	$\lambda_1 A+ \lambda_2 B+ \lambda_3 C+ \lambda_4 D+ \lambda_5 E$ 式中 $\Sigma \lambda_i=1$		
教师评语	备注：比较高分数和比较低分数的实验报告必须要给出评语，指明给分的具体依据 （此条打印时请删除）		

实验一　线性分类器的设计与实现

1、实验目的

（1）掌握 Fisher 线性判别方法
（2）掌握感知器算法
（3）掌握分类器错误率的统计方法

2、实验内容及要求

1）仿真产生两类服从正态分布的训练样本，各类样本数均为 500。 两类的均值向量、协方差矩阵如下：

$$\boldsymbol{\mu}_1=[-2,-2]^T,\ \boldsymbol{\Sigma}_1=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix},\ \boldsymbol{\mu}_2=[2,2]^T,\ \boldsymbol{\Sigma}_2=\begin{bmatrix}1 & 0\\ 0 & 4\end{bmatrix}$$

2）根据仿真产生的数据统计各类的均值向量、协方差矩阵、类内离散度矩阵、类间离散度矩阵，并按 FISHER 准则求解最优投影方向 **W***； 计算投影后的阈值权，阈值权计算方法自定，为简单起见，可以采用 $W_0=-\dfrac{m_1^{'}+m_2^{'}}{2}$，这里，$m_1^{'},m_2^{'}$ 分别是投影后两类的均值。

3）各类重新产生 10000 个测试样本，并利用 2）设计得到的分类器对测试样本进行分类，统计各种错误率。并与理论计算的错误率比较。

4）以蠓虫分类问题作为实验对象，采用训练样本设计分类器，分别利用 FISHER 准则及感知器算法（感知器算法可以采用批处理感知器算法或者逐个样本修正的感知器算法）对两类蠓虫（A_f 及 A_{pf}）数据进行分类，得到判别函数。并对训练得到的判别函数，用测试数据进行测试评价。

蠓虫分类实验数据如下：

1）训练样本（线性可分）

号										0	1	2	3	4	5
别	A_f	A_f	A_f	A_f	A_f	A_f	A_f	A_f	A_{pf}	A_{pf}	A_{pf}	A_{pf}	A_{pf}	A_{pf}	A_{pf}
角长度	1.24	1.36	1.38	1.38	1.38	1.40	1.48	1.54	1.56	1.14	1.18	1.20	1.26	1.28	1.30
膀长度	1.27	1.74	1.64	1.82	1.90	1.70	1.82	1.82	2.08	1.82	1.96	1.86	2.0	2.0	1.96

2）测试样本

序号	1	2	3
类别	A_f	A_f	A_{pf}
触角长度	1.24	1.28	1.40
翅膀长度	1.8	1.84	2.04

3、实验原理

1）Fisher 准则基本原理：

如果在二维空间中一条直线能将两类样本分开，或者错分类很少，则同一类别样本数据在该直线的单位法向量上的投影的绝大多数都应该超过某一值。而另一类数据的投影都应该小于(或绝大多数都小于)该值，则这条直线就有可能将两类分开。

准则：向量 W 的方向选择应能使两类样本投影的均值之差尽可能大些，而使类内样本的离散程度尽可能小。这就是 Fisher 准则函数的基本思路。 $y=\mathbf{W}^{T}X+W_0$

评价投影方向 **W** 的函数 ： $J_F(W)=\dfrac{W^T S_b W}{W^T S_W W}$

最佳 **W** 值的确定：求取使 J_F 达极大值时的 w*： $W^*=S_W^{-1}(m_1-m_2)$

阈值权 W_0 确定 ： $W_0=-\dfrac{m_1^{'}+m_2^{'}}{2}$ ，这里： $m_i^{'}=W^{*T}m_i \quad i=1,2$

多元正态分布概率密度函数：

$$p(X)=\frac{1}{(2\pi)^{d/2}\,|\Sigma|^{1/2}}e^{-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)}$$

其中： μ 是 d 维均值向量： $\mu=E\{X\}=[\mu_1,\mu_2,...,\mu_d]^T$

Σ 是 d×d 维协方差矩阵： $\Sigma=E[(X-\mu)(X-\mu)^T]$

（1）估计类均值向量和协方差矩阵的估计

各类均值向量 $m_i=\dfrac{1}{N_i}\sum_{X\in\omega_i}X$

各类协方差矩阵 $\Sigma_i=\dfrac{1}{N_i}\sum_{X\in\omega_i}(X-\mu_i)(X-\mu_i)^T$

（2）类间离散度矩阵、类内离散度矩阵的计算

类内离散度矩阵： $S_i=\sum_{X\in\omega_i}(X-m_i)(X-m_i)^T$ ， i=1,2

总的类内离散度矩阵： $S_W=S_1+S_2$

类间离散度矩阵： $S_b = (m_1 - m_2)(m_1 - m_2)^T$

2）感知器算法

感知器（Perception）是一种神经网络模型，用于对两类线性可分样本进行分类。

对于两类线性可分的样本，感知器算法使用增广样本向量，因此，首先需要将样本变成增广向量；其次需要对样本做规范化处理，即：对第二类的样本全部乘以（-1）。

对于训练样本，判别函数满足： $g(Y) = \mathbf{A}^T\mathbf{Y} > 0$

其中： $\mathbf{A} = [\omega_0, \omega_1, \omega_2, ..., \omega_d]^T$ ， $\mathbf{Y} = [1, x_1, x_2, ..., x_d]^T$ （规范化增广样本向量）

感知器算法（Perception Approach）通过对已知类别属性的训练样本集的学习，找一个满足上式的权向量，步骤如下：

（1） 将训练样本变成增广向量形式，并做规范化处理。

（2） 初始化权向量：**A**。

（3） 用全部训练样本进行一轮迭代。依次输入样本，记录错分类样本，设错分类样本集为 y_k

（4） 若 y_k 是空集，则 **A**=**A**(k)，迭代结束； 否则，转 5）

（5） ρ_k， 令 k=k+1 ，
$$\nabla J_p(\mathbf{A}) = \sum_{y \in y^k} -y,$$
$$\mathbf{A}(k+1) = \mathbf{A}(k) - \rho_k \nabla J_p$$

（6） 转 3）

上述算法是批处理感知器算法。

4、实验结果及分析

FISHER 分类器

Fisher 分类器是一种经典的模式识别和机器学习方法，用于进行有监督的分类任务。它的原理基于线性投影，旨在找到一个最佳的投影方向，使得不同类别的样本在投影后能够更好地分离开来。

数据预处理：首先，需要对输入数据进行预处理，包括去除噪声、归一化等操作，以确保数据的质量和可比性。

类内散度和类间散度计算：接下来，计算每个类别的类内散度矩阵和整体数据的类间散度矩阵。类内散度矩阵衡量了同一类别内样本之间的差异程度，而类间散度矩阵衡量了不同类别之间的差异程度。

投影向量计算：通过对类内散度矩阵和类间散度矩阵进行特征值分解，可以得到一组特征向量。这些特征向量被称为投影向量，它们定义了一个新的特征空间。选择特征值较大的投影向量，可以使得投影后的数据能够更好地保留类别信息。

降维和分类：根据选择的投影向量，将原始数据映射到新的特征空间中。在新的特征空间中，可以使用简单的分类算法（如最近邻算法）进行分类。

Fisher 分类器的优点在于它能够通过投影找到一个最佳的分离超平面，使得不同类别的样本更好地分开。然而，Fisher 分类器的一个限制是它假设数据服从高斯分布，并且对

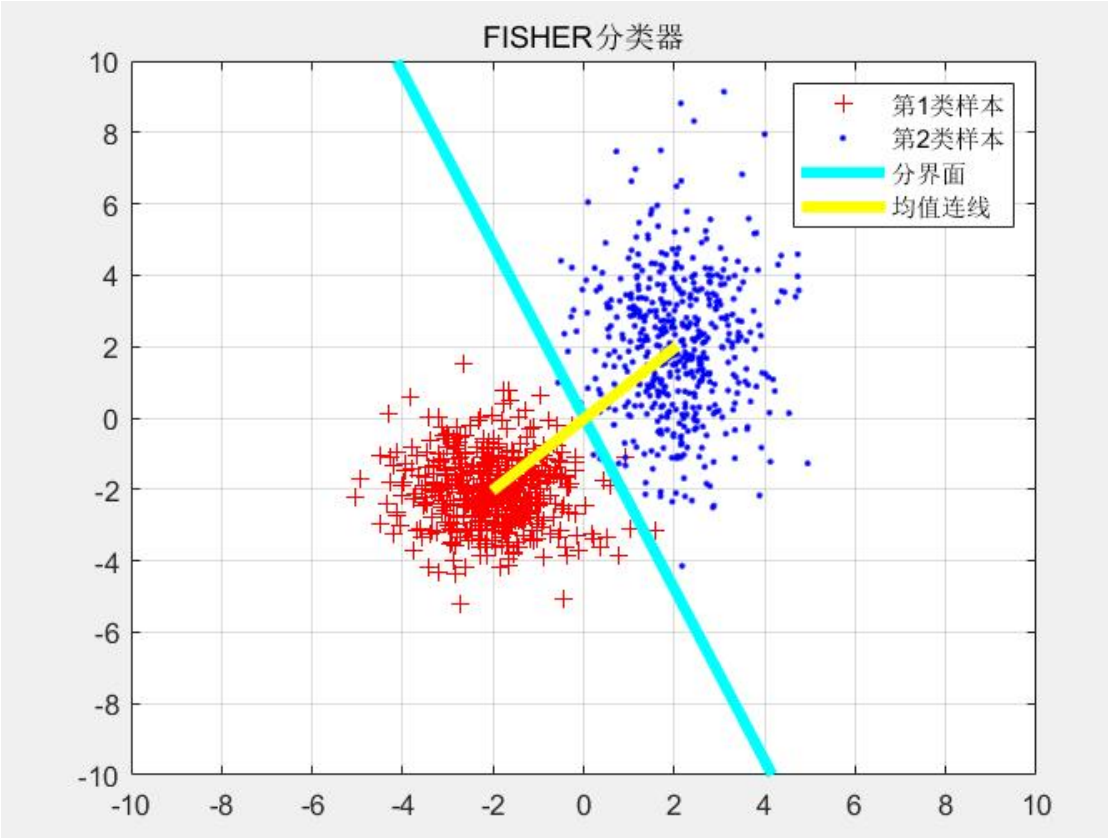
于非线性可分的数据，效果可能不佳。在实际应用中，可以结合其他方法来改进 Fisher 分类器的性能，例如核技巧或者使用非线性变换。

具体代码见下。

这段程序实现了使用 Fisher 分类器对两类数据进行分类，并计算分类的错误率。

具体步骤如下：

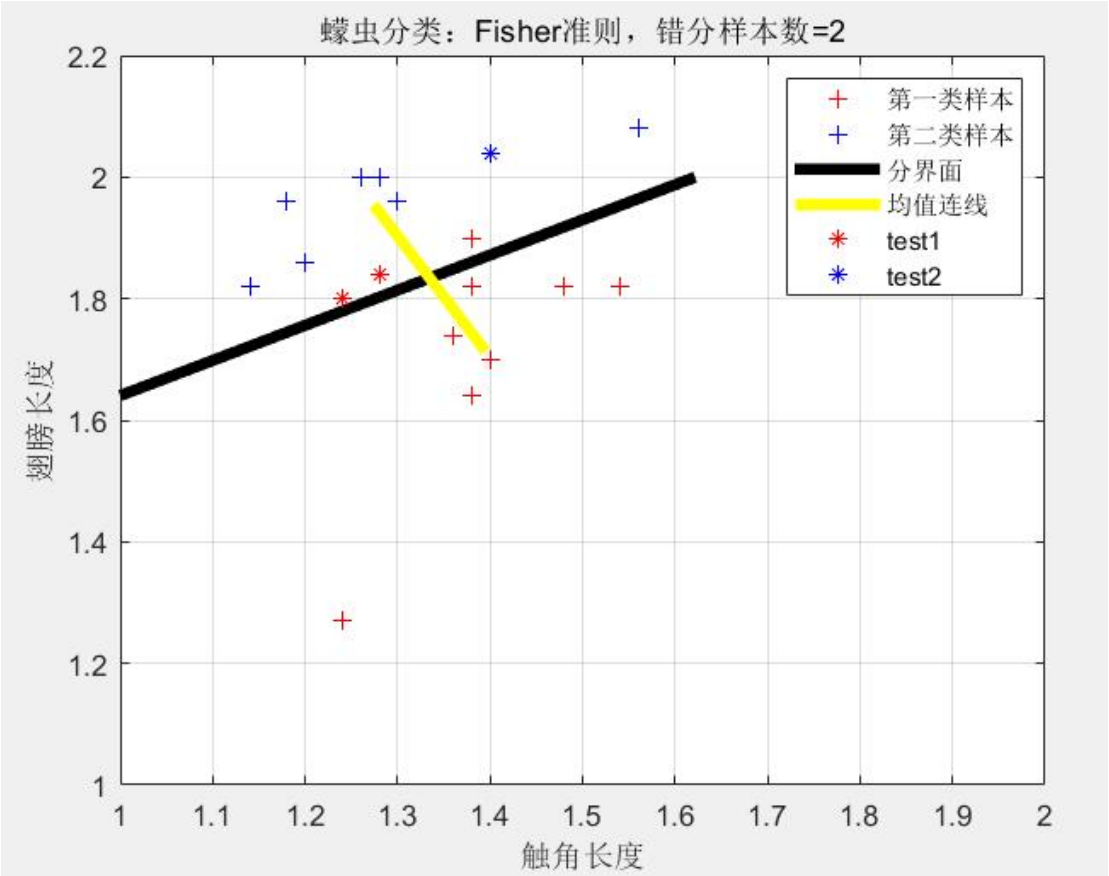
1. 定义两个高斯分布的参数 mu1、sigma1、n1、c1，以及 mu2、sigma2、n2、c2，生成两类训练样本。
2. 画出两类样本的散点图，并计算两类样本的均值 m1、m2 和协方差矩阵 S1、S2。
3. 计算 Sw=S1+S2 和权向量 W=(Swˆ(-1))(m1-m2)，并将 W 归一化。
4. 计算样本在 W 方向上的投影 m11=W'*m1'和 m22=W'*m2'，以及阈值权 W0=-(m11+m22)/2。
5. 画出分类的分界面和均值连线。
6. 定义测试样本的数量 NT1 和 NT2，并生成两类测试样本 test1 和 test2。
7. 计算 Pw1 和 Pw2，即两类样本的先验概率。
8. 对测试样本进行分类，计算 Pe1 和 Pe2，即第一类样本错分为第二类和第二类样本错分为第一类的错误率。
9. 计算平均错误率 Pe=Pe1*Pw1+Pe2*Pw2。




```
阈值权=0.0059125
第1类样本错分为第2类的错误率=0.0038
第2类样本错分为第1类的错误率=0.0157
平均错误率=0.00975
```

蠓虫分类：Fisher 准则算法

本算法原理与上述 FISHER 分类器相同，将其运用到实际的蠓虫分类问题当中，可以见到，错分样本数为 2，得知 FISHER 准则算法在蠓虫分类上很有效。




```
阈值权=0.91969
```

蠓虫分类：批量处理感知器算法

批量处理感知器算法是一种经典的二元分类算法，它基于感知器模型和梯度下降算法。以下是该算法的原理介绍：

1.可行域

- 假设有两类样本，分别为正例和负例。我们要找到一个超平面来将它们分开。
- 超平面可以表示为 $wx + b = 0$，其中 w 是权向量，x 是输入特征向量，b 是阈值。

2. 感知器模型：

- 感知器模型是一个二元分类器，它将输入特征向量 x 映射到两个类别之一。
- 对于给定的输入 x，感知器的输出为：

  $y = \text{sign}(wx + b)$，

  其中 sign 是符号函数，如果 $wx + b$ 大于等于零，则输出为 +1，否则为 -1。

3. 目标函数：

- 批量处理感知器算法的目标是最小化误分类样本的数量。

- 定义误分类样本集合为 M，其包含所有被错误分类的样本。
- 目标函数可以表示为：

  $E(w, b) = -\Sigma [wx + b]$，其中 $[wx + b]$ 表示对样本 x 的预测结果。

4. 参数更新：
   - 使用梯度下降法来更新参数 w 和 b。
   - 参数的更新公式为：

     $w(t+1) = w(t) + \eta \Sigma [y(t) - sign(wx(t) + b(t))]x(t)$，其中 η 是学习率，t 表示迭代次数。

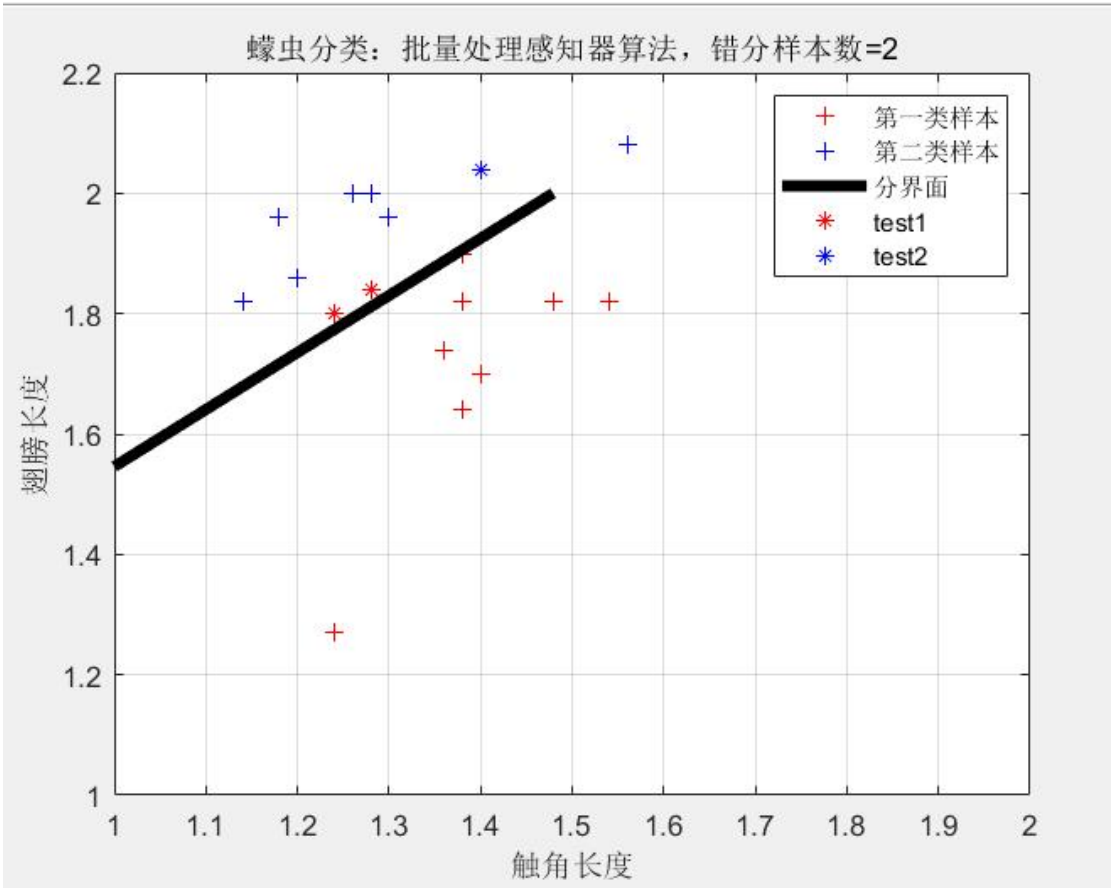
     $b(t+1) = b(t) + \eta \Sigma [y(t) - sign(wx(t) + b(t))]$
5. 算法步骤：
   - 初始化权向量 w 和阈值 b。
   - 对于每个样本 x：
     - 计算预测结果 $y = sign(wx + b)$。
     - 如果样本被错误分类（即 y 与真实标签不一致），则更新参数 w 和 b。
   - 重复上述步骤，直到所有样本都被正确分类或达到最大迭代次数。
6. 分类边界：
   - 分类边界是由权向量 w 和阈值 b 决定的超平面。
   - 如果一个样本点位于超平面的一侧，则将其分类为正例；如果位于另一侧，则分类为负例。

批量处理感知器算法通过迭代更新参数来找到一个能够将两类样本分开的超平面。它具有简单、高效的特点，并且在线性可分的情况下能够收敛到最优解。然而，在线性不可分的情况下，该算法可能无法收敛或者收敛到局部最优解。

下面是程序的步骤：

1. 定义数据：定义了一个矩阵 A，其中包含两类样本的触角长度和翅膀长度数据。
2. 数据处理：将数据分成两个矩阵 x1 和 x2，分别表示两类样本的触角长度和翅膀长度数据。同时，定义了每个类别的样本数量 n1 和 n2。
3. 绘制样本散点图：绘制了两类样本的散点图，用红色加号表示第一类样本，用蓝色加号表示第二类样本。
4. 训练批量处理感知器分类器：使用 FUNBatchPerceptron 函数训练批量处理感知器分类器。该函数会返回权向量 A 和迭代次数 LOOP。注释中还提供了另一种训练方法 FUNSinglePerceptron 的调用方式。
5. 计算归一化权向量和阈值权：根据权向量 A 计算归一化权向量 w 和阈值权 w0。
6. 绘制分界线：根据归一化权向量 w 和阈值权 w0，绘制分类的分界线。
7. 测试样本分类：定义了两个测试样本 b1 和 b2，并使用训练得到的权向量和阈值对其进行分类。将分类结果以红色星号和蓝色星号绘制在样本散点图上。
8. 错误测试：计算分类错误的样本数 Ne1 和 Ne2，并求得总的错误样本数 Ne。
9. 绘制分类结果图：在标题中显示分类结果和错误样本数。

在本实验中，批量处理感知器算法是有效的，错分样本数为 2，可以有效区分出第一类样本和第二类样本。

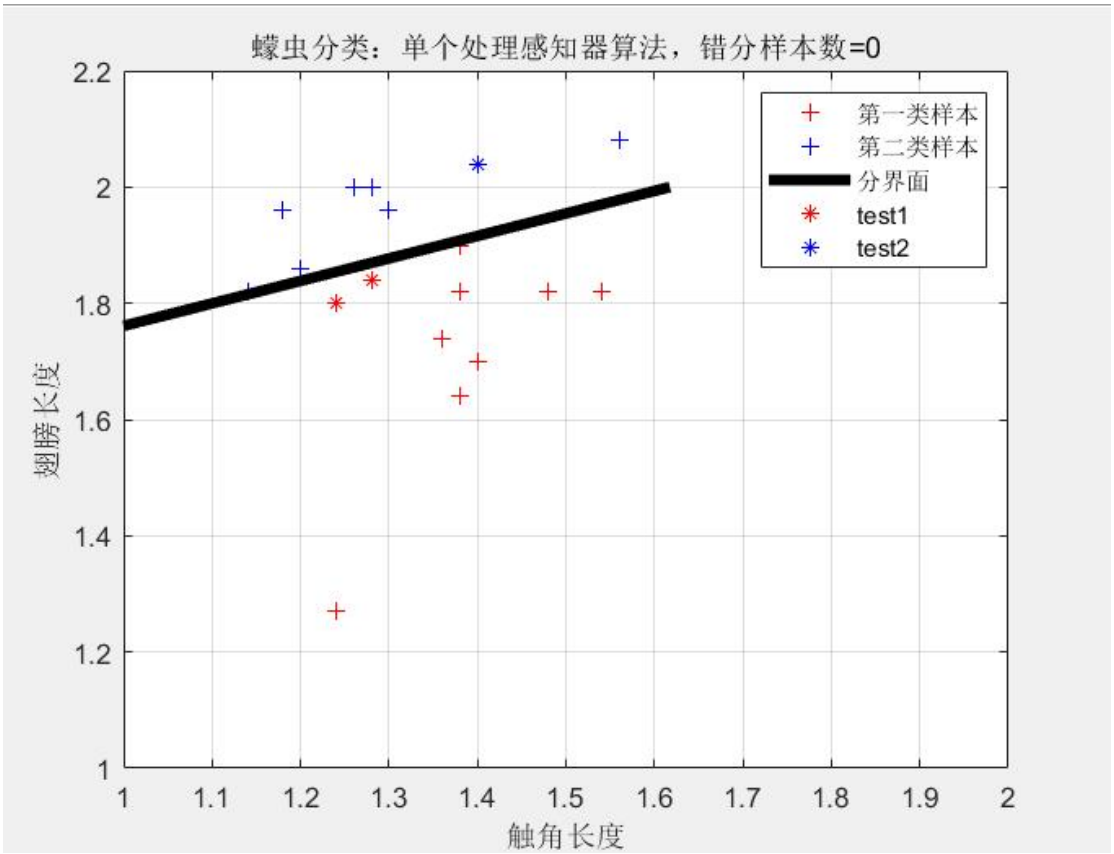


蠓虫分类：单个处理感知器算法

单个处理感知器算法只能处理单个样本，即每次只输入一个样本，计算并更新权重参数。这种算法只能进行有限的模式识别任务，因为它没有考虑到样本之间的关系，无法处理大规模数据。

相比之下，批量处理感知器算法可以同时处理多个样本。在每次迭代中，它会将所有样本的误差累加起来，并根据总误差来更新权重参数。

总之，批量处理感知器算法和单个处理感知器算法的主要区别在于处理数据的方式不同。在本实验中，单个处理感知器算法错分样本数为 0，是最精确的，在数据比较少时比较有优势。



5、源程序

Fisher 分类器

```
mu1=[-2,-2]'
sigma1=[1 0;0 1];
n1=500;
c1=mvnrnd(mu1,sigma1,n1);
mu2=[2,2]'
sigma2=[1 0;0 4];
n2=500;
c2=mvnrnd(mu2,sigma2,n2);
figure;
plot(c1(:,1),c1(:,2),'r+');grid on;hold on;
plot(c2(:,1),c2(:,2),'b+');grid on;hold on;
title('两类训练样本');
m1=mean(c1);
m2=mean(c2);
S1=0;
S2=0;
for i=1:n1
S1=S1+(c1(i,:)-m1)'*(c1(i,:)-m1);
end
```

```
for i=1:n2
S2=S2+(c2(i,:)-m2)'*(c2(i,:)-m2);
end
S1
S2
Sw=S1+S2;
W=inv(Sw)*(m1'-m2');
W=W./norm(W);
disp(['归一化权向量=[',num2str(W'),']'])
m11=W'*m1';
m22=W'*m2';
m11
m22
W0=-(m11+m22)/2;
disp(['阈值权=',num2str(W0)])
a=W(1);
b=W(2);
c=W0;
syms x y
f=a*x+b*y+c;
h=fimplicit(f,[-10 10 -10 10]);
set(h,'color','black','LineWidth',4);
line([m1(1) m2(1)],[m1(2) m2(2)],'color','yellow','LineWidth',4);
legend('第一类样本','第二类样本','分界面','均值连线');
title(['Fisher 分类器']);
NT1=10000;
NT2=10000;
test1=mvnrnd(mu1,sigma1,NT1);
test2=mvnrnd(mu2,sigma2,NT2);
Pw1=NT1/(NT1+NT2);
Pw2=NT2/(NT1+NT2);
Ne1=0;
for k=1:NT1
x=test1(k,:);
gx=W'*x'+W0;
if gx>=0
XLabel=1;
else
Ne1=Ne1+1;
XLabel=-1;
end
end
Pe1=Ne1/NT1;
Ne2=0;
```

```
for k=1:NT2
x=test2(k,:);
gx=W'*x'+W0;
if gx<0
XLabel=-1;
else
XLabel=1;
Ne2=Ne2+1;
end
end
Pe2=Ne2/NT2;
Pe=Pe1*Pw1+Pe2*Pw2;
disp(['第 1 类样本错分为第 2 类的错误率=',num2str(Pe1)]);
disp(['第 2 类样本错分为第 1 类的错误率=',num2str(Pe2)]);
disp(['平均错误率=',num2str(Pe)]);
```

蠓虫分类：Fisher 准则算法

```
clc;clear all;close all;
A=[1.24 1.36 1.38 1.38 1.38 1.40 1.48 1.54 1.56 1.14 1.18 1.20 1.26 1.28
1.30
1.27 1.74 1.64 1.82 1.90 1.70 1.82 1.82 2.08 1.82 1.96 1.86 2.0 2.0 1.96]
c1=A(:,1:8);
n1=8;
c2=A(:,9:15);
n2=7;
figure;
plot(c1(1,:),c1(2,:),'r+');grid on;hold on;
plot(c2(1,:),c2(2,:),'b.');grid on;hold on;
m1=mean(c1');
m2=mean(c2');
S1=0;S2=0;
for i=1:n1
S1=S1+(c1(:,i)-m1)'*(c1(:,i)-m1);
end
for i=1:n2
S2=S2+(c2(:,i)-m2)'*(c2(:,i)-m2);
end
S1
S2
Sw=S1+S2;
W=inv(Sw)*(m1'-m2');
W=W./norm(W);
disp(['归一化权向量=[',num2str(W'),']'])
```

```
m11=W'*m1';
m22=W'*m2';
m11
m22
w0=-(m11+m22)/2;
disp(['阈值权=',num2str(w0)])
a=W(1);b=W(2);c=w0;
syms x y;
f=a*x+b*y+c;
h=fimplicit(f,[1 2 1 2]);
set(h,'color','cyan','LineWidth',4);
%line([m1(1) m2(1)],[m1(2) m2(2)],'color','yellow','LineWidth',4)
B1=[1.24 1.28
1.8 1.84];
B2=[1.0
2.04];
plot(B1(1,:),B1(2,:),'r*');grid on;hold on;
plot(B2(1,:),B2(2,:),'b*');grid on;hold on;
xlabel('触角长度');ylabel('翅膀长度');
legend('第 1 类训练样本','第 2 类训练样本','分界面','第 1 类测试样本','第 2 类测试样本')
Ne1=0;
for k=1:2
x=B1(:,k);
gx=W'*x+w0;
if gx>0
XLabel=1;
else
XLabel=-1;
Ne1=Ne1+1;
end
end

Ne2=0;
for k=1:1
x=B2(k,:);
gx=W'*x+w0;
if gx>0
XLabel=1;
Ne2=Ne2+1;
else
XLabel=-1;
end
end
```

```
Ne=Ne1+Ne2;
title(['蠓虫分类，FISHER 准则算法，错分样本数=',num2str(Ne)])
```

蠓虫分类：批量处理感知器算法

```
A=[1.24 1.36 1.38 1.38 1.38 1.40 1.48 1.54 1.56 1.14 1.18 1.20 1.26 1.28
1.30
1.27 1.74 1.64 1.82 1.90 1.70 1.82 1.82 2.08 1.82 1.96 1.86 2.0 2.0 1.96];


x1=A(: ,1:8);
n1=8;
x2=A(: ,9:15);
n2=7;

figure;
plot(x1(1,:),x1(2,:),'r+');grid on;hold on;
plot(x2(1,:),x2(2,:),'b+');grid on;hold on;
title('两类训练样本');

A0=[0 0 0]';
rho=1;

[A,LOOP]=FUNBatchPerceptron(x1',x2',rho,A0)
%[A,LOOP]=FUNSinglePerceptron(x1',x2',rho,A0)

w=A(2:3);
y0=A(1);
w0=A(1)/norm(w);

w=w./norm(w);
disp(['归一化权向量=[',num2str(w'),']'])
disp(['阈值权=',num2str(w0)])

%分界线
a=w(1);
b=w(2);
c=w0;

syms x y
f=a*x+b*y+c;

h=fimplicit(f,[1 2 1 2]);
```

```
set(h,'color','black','LineWidth',4);
%line([m1(1) m2(1)],[m1(2) m2(2)],'color','yellow','LineWidth',4);
xlabel('触角长度');ylabel('翅膀长度');
title('蠓虫分类-批处理感知器算法');

%测试样本分类
b1=[1.24 1.28
1.8 1.84 ];
b2=[1.4
2.04];
plot(b1(1,:),b1(2,:),'r*');grid on;hold on;
plot(b2(1,:),b2(2,:),'b*');grid on;hold on;
xlabel('触角长度');
ylabel('翅膀长度');
legend('第一类样本','第二类样本','分界面','test1','test2');

%错误测试
Ne1=0;
for k=1:2
x=b1(:,k);
gx=w'*x+w0;
if gx>=0
XLabel=1;
else
XLabel=-1;
Ne1=Ne1+1;
end
end
Ne1

Ne2=0;
for k=1:1
x=b2( :, k);
gx=w'*x+w0;
if gx<0
XLabel=-1;
else
XLabel=1;
Ne2=Ne2+1;
end
end

Ne2
```

```
Ne=Ne1+Ne2
title(['蠓虫分类：批量处理感知器算法，错分样本数=',num2str(Ne)]);
```

蠓虫分类：单个处理感知器算法

```
A=[1.24 1.36 1.38 1.38 1.38 1.40 1.48 1.54 1.56 1.14 1.18 1.20 1.26 1.28
1.30
1.27 1.74 1.64 1.82 1.90 1.70 1.82 1.82 2.08 1.82 1.96 1.86 2.0 2.0 1.96];


x1=A(: ,1:8);
n1=8;
x2=A(: ,9:15);
n2=7;

figure;
plot(x1(1,:),x1(2,:),'r+');grid on;hold on;
plot(x2(1,:),x2(2,:),'b+');grid on;hold on;
title('两类训练样本');

A0=[0 0 0]';
rho=1;

%[A,LOOP]=FUNBatchPerceptron(x1',x2',rho,A0)
[A,LOOP]=FUNSinglePerceptron(x1',x2',rho,A0)

w=A(2:3);
y0=A(1);
w0=A(1)/norm(w);

w=w./norm(w);
disp(['归一化权向量=[',num2str(w'),']'])
disp(['阈值权=',num2str(w0)])

%分界线
a=w(1);
b=w(2);
c=w0;

syms x y
f=a*x+b*y+c;

h=fimplicit(f,[1 2 1 2]);
```

```
set(h,'color','black','LineWidth',4);
%line([m1(1) m2(1)],[m1(2) m2(2)],'color','yellow','LineWidth',4);
xlabel('触角长度');ylabel('翅膀长度');
title('蠓虫分类-单个处理感知器算法');

%测试样本分类
b1=[1.24 1.28
1.8 1.84 ];
b2=[1.4
2.04];
plot(b1(1,:),b1(2,:),'r*');grid on;hold on;
plot(b2(1,:),b2(2,:),'b*');grid on;hold on;
xlabel('触角长度');
ylabel('翅膀长度');
legend('第一类样本','第二类样本','分界面','test1','test2');

%错误测试
Ne1=0;
for k=1:2
x=b1(:,k);
gx=w'*x+w0;
if gx>=0
XLabel=1;
else
XLabel=-1;
Ne1=Ne1+1;
end
end
Ne1

Ne2=0;
for k=1:1
x=b2( :, k);
gx=w'*x+w0;
if gx<0
XLabel=-1;
else
XLabel=1;
Ne2=Ne2+1;
end
end

Ne2
```

```
Ne=Ne1+Ne2
title(['蠓虫分类：单个处理感知器算法，错分样本数=',num2str(Ne)]);
```