实验二 线性判别与非线性判别

一、 实验目的

掌握几何分类法的基本思想。通过线性判别函数，非线性判别函数基本概念和方法的学习,掌握感知器算法，掌握前馈神经网络与反向传播算法，了解多模式分类方法与概念。

二、 实验内容

1、Iris 数据集以鸢尾花的特征作为数量来源。该数据集由 3 种不同类型的鸢尾花的 50 个样本数据构成。使用感知器算法对 3 种类型的鸢尾花两两分类。

2、使用 BP 算法对 Iris 数据集中的 3 类样本分类。

3、产生以(0, 0)，(1, 1)，(1, 0)，(0, 1)为中心的数据样本，其中(0,0)和(1,1)为一类，(1,0)和(0,1)为一类，使用 BP 算法对这两类数据点分类。

三、 实验原理

1. 感知器算法

（1） 感知器模型

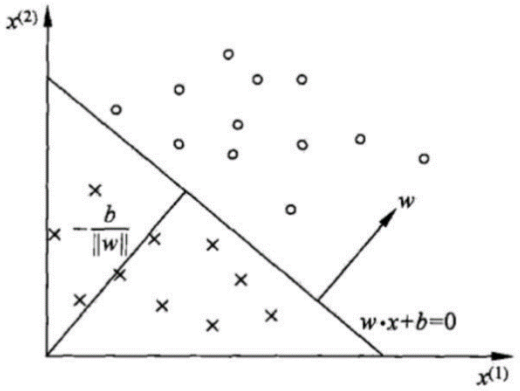
感知器是一种线性分类的二分类模型，输入为实例的特征向量，输出为实例的类别，分别用 1 和 −1 表示。感知机将输入空间（特征空间）中的实例划分为正负两类分离的超平面，旨在求出将训练集进行线性划分的超平面，为此，导入基于误分类的损失函数，利用梯度下降法对损失函数进行极小化，求得最优解。假设输入空间是$\mathrm{x} \in R^n$，输出空间是$\gamma = (+1, -1)$，由输入空间到输出空间的映射，也即感知器的函数公式如下：

$$f(x) = sign(w \cdot x + b)$$

其中，w和b称为感知机模型参数，$\mathrm{w} \in R^n$叫做权值或者权值向量，$b \in R$叫做偏置，w · x表示二者的内积，$sign$是符号函数，即：

$$sign(x) = \begin{cases} +1, & x \geq 0 \\ -1, & x < 0 \end{cases}$$

感知机的几何解释：线性方程$w \cdot x + b = 0$对应特征空间R^n中的一个超平面S，其中w是超平面的法向量，b是超平面的截距，该超平面将特征空间分为两个部分，将特征向量分为正负两类，如下图所示：



（2） 学习策略

假设训练数据集是线性可分的，感知机的学习目标就是找到能够将正负实例点完全分开的超平面，即确定感知机模型参数w和b，需定义损失函数并最小化损失函数。感知器$sign(w \cdot x + b)$的损失函数定义如下，其中M为误分类点的集合。

$$L(w,b) = -\sum_{x \in M} y_i(w \cdot x_i + b)$$

感知器算法是使得损失函数$L(w,b)$极小化的最优化问题，可以使用随机梯度下降法来进行最优化。在极小化目标函数的过程中，并不是一次使M中所有误分类的点梯度下降，而是每次随机一个误分类的点进行梯度下降。具体步骤为:

a) 假设误分类点的集合为M，那么损失函数$L(w,b)$的梯度为:

$$\nabla_w L(w,b) = -\sum_{x \in M} y_i x_i$$

$$\nabla_b L(w,b) = -\sum_{x \in M} y_i$$

b) 随机选取一个误分类的点(x_i, y_i)，对w, b进行更新:

$$w \leftarrow w + \eta y_i x_i$$

$$b \leftarrow b + \eta y_i$$

式中$\eta(0 < \eta \leq 1)$是步长，又称为学习率。通过迭代可以使损失函数不断减小，直到为 0。当训练数据集线性可分的时候，感知机学习算法是收敛的，并且存在无穷多个解，解会随着不同的初值或不同的迭代顺序不同而有所不同。

2. BP 算法

基本 BP 算法包括两个方面：信号的前向传播和误差的反向传播。即计算实际输出时按从输入到输出的方向进行，而权值和阈值的修正从输出到输入的方向进行。

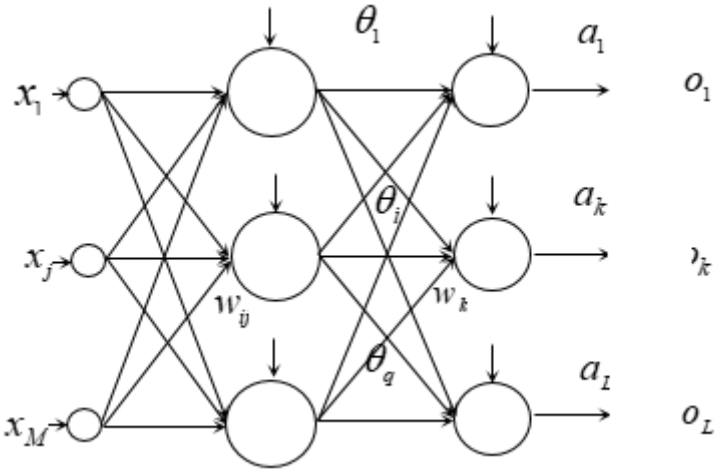


图 1　BP 网络结构

图中：x_j 表示输入层第 j 个节点的输入，j=1,…,M;

w_{ij} 表示隐含层第 i 个节点到输入层第 j 个节点之间的权值;

θ_i 表示隐含层第 i 个节点的阈值;

$\phi(x)$ 表示隐含层的激励函数;

w_{ki} 表示输出层第 k 个节点到隐含层第 i 个节点之间的权值，i=1,…,q;

a_k 表示输出层第 k 个节点的阈值,k=1,…,L;

$\psi(x)$ 表示输出层的激励函数;

o_k 表示输出层第 k 个节点的输出。

（1）信号的前向传播过程

隐含层第 i 个节点的输入 neti:

$$net_i = \sum_{j=1}^{M} w_{ij} x_j + \theta_i$$

隐含层第 i 个节点的输出 yi:

$$y_i = \phi(net_i) = \phi(\sum_{j=1}^{M} w_{ij} x_j + \theta_i)$$

输出层第 k 个节点的输入 netk:

$$net_k = \sum_{i=1}^{q} w_{ki} y_i + a_k = \sum_{i=1}^{q} w_{ki} \phi(\sum_{j=1}^{M} w_{ij} x_j + \theta_i) + a_k$$

输出层第 k 个节点的输出 ok:

$$o_k = \psi(net_k) = \psi(\sum_{i=1}^{q} w_{ki} y_i + a_k) = \psi\left(\sum_{i=1}^{q} w_{ki} \phi(\sum_{j=1}^{M} w_{ij} x_j + \theta_i) + a_k\right)$$

（2）误差的反向传播过程

误差的反向传播，即首先由输出层开始逐层计算各层神经元的输出误差，然后根据误差梯度下降法来调节各层的权值和阈值，使修改后的网络的最终输出能接近期望值。

对于每一个样本 p 的二次型误差准则函数为 Ep:

$$E_p = \frac{1}{2} \sum_{k=1}^{L} (T_k - o_k)^2$$

系统对 P 个训练样本的总误差准则函数为:

$$E = \frac{1}{2} \sum_{p=1}^{P} \sum_{k=1}^{L} (T_k^p - o_k^p)^2$$

根据误差梯度下降法依次修正输出层权值的修正量Δwki，输出层阈值的修正量Δak，隐含层权值的修正量Δwij，隐含层阈值的修正量$\Delta\theta_i$。

$$\Delta w_{ki} = -\eta \frac{\partial E}{\partial w_{ki}}; \quad \Delta a_k = -\eta \frac{\partial E}{\partial a_k}; \quad \Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}; \quad \Delta \theta_i = -\eta \frac{\partial E}{\partial \theta_i}$$

输出层权值调整公式:

$$\Delta w_{ki} = -\eta \frac{\partial E}{\partial w_{ki}} = -\eta \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial w_{ki}} = -\eta \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial net_k} \frac{\partial net_k}{\partial w_{ki}}$$

输出层阈值调整公式:

$$\Delta a_k = -\eta \frac{\partial E}{\partial a_k} = -\eta \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial a_k} = -\eta \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial net_k} \frac{\partial net_k}{\partial a_k}$$

隐含层权值调整公式：

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta \frac{\partial E}{\partial net_i} \frac{\partial net_i}{\partial w_{ij}} = -\eta \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial net_i} \frac{\partial net_i}{\partial w_{ij}}$$

隐含层阈值调整公式：

$$\Delta \theta_i = -\eta \frac{\partial E}{\partial \theta_i} = -\eta \frac{\partial E}{\partial net_i} \frac{\partial net_i}{\partial \theta_i} = -\eta \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial net_i} \frac{\partial net_i}{\partial \theta_i}$$

又因为：

$$\frac{\partial E}{\partial o_k} = -\sum_{p=1}^{P} \sum_{k=1}^{L} (T_k{}^p - o_k{}^p)$$

$$\frac{\partial net_k}{\partial w_{ki}} = y_i \quad , \quad \frac{\partial net_k}{\partial a_k} = 1 \quad , \quad \frac{\partial net_i}{\partial w_{ij}} = x_j \quad , \quad \frac{\partial net_i}{\partial \theta_i} = 1$$

$$\frac{\partial E}{\partial y_i} = -\sum_{p=1}^{P} \sum_{k=1}^{L} (T_k^p - o_k^p) \cdot \psi'(net_k) \cdot w_{ki}$$

$$\frac{\partial y_i}{\partial net_i} = \phi'(net_i)$$

$$\frac{\partial o_k}{\partial net_k} = \psi'(net_k)$$

所以最后得到以下公式：

$$\Delta w_{ki} = \eta \sum_{p=1}^{P} \sum_{k=1}^{L} (T_k{}^p - o_k{}^p) \cdot \psi'\left(net_k\right) \cdot y_i$$

$$\Delta a_k = \eta \sum_{p=1}^{P} \sum_{k=1}^{L} (T_k{}^p - o_k{}^p) \cdot \psi'\left(net_k\right)$$

$$\Delta w_{ij} = \eta \sum_{p=1}^{P} \sum_{k=1}^{L} (T_k^p - o_k^p) \cdot \psi'\left(net_k\right) \cdot w_{ki} \cdot \phi'(net_i) \cdot x_j$$

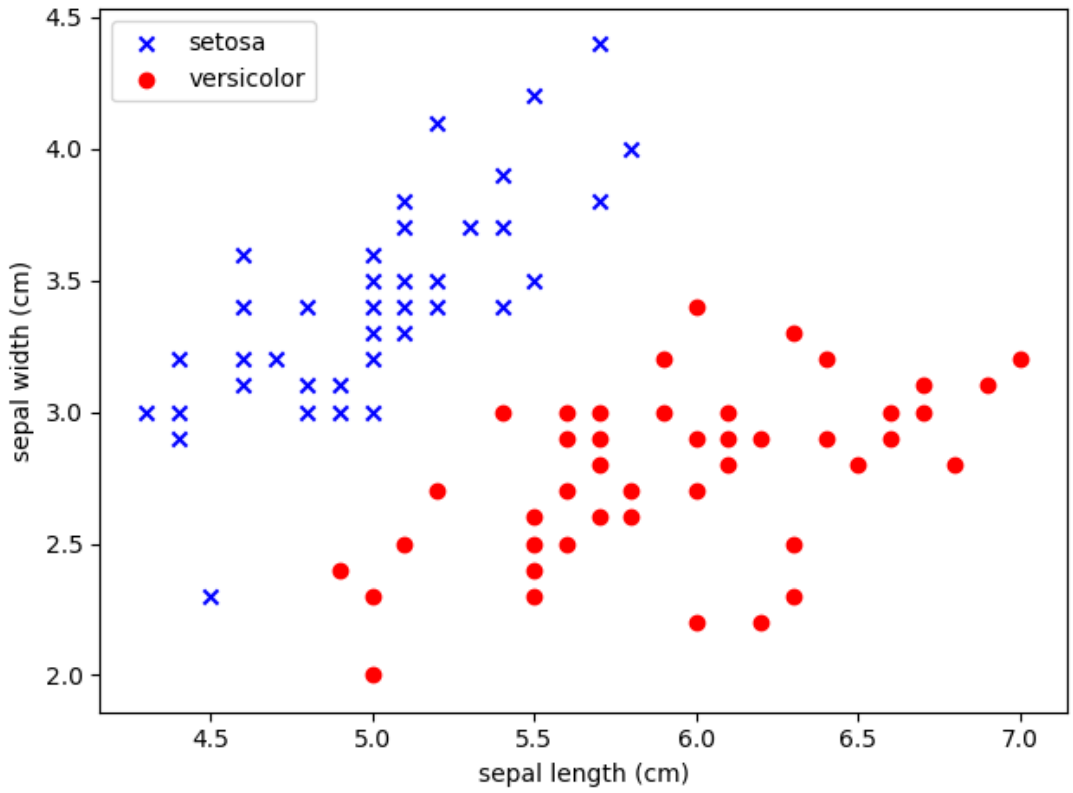
$$\Delta\theta_i = \eta \sum_{p=1}^{P} \sum_{k=1}^{L} (T_k^p - o_k^p) \cdot \psi'(net_k) \cdot w_{ki} \cdot \phi'(net_i)$$

三、实验过程

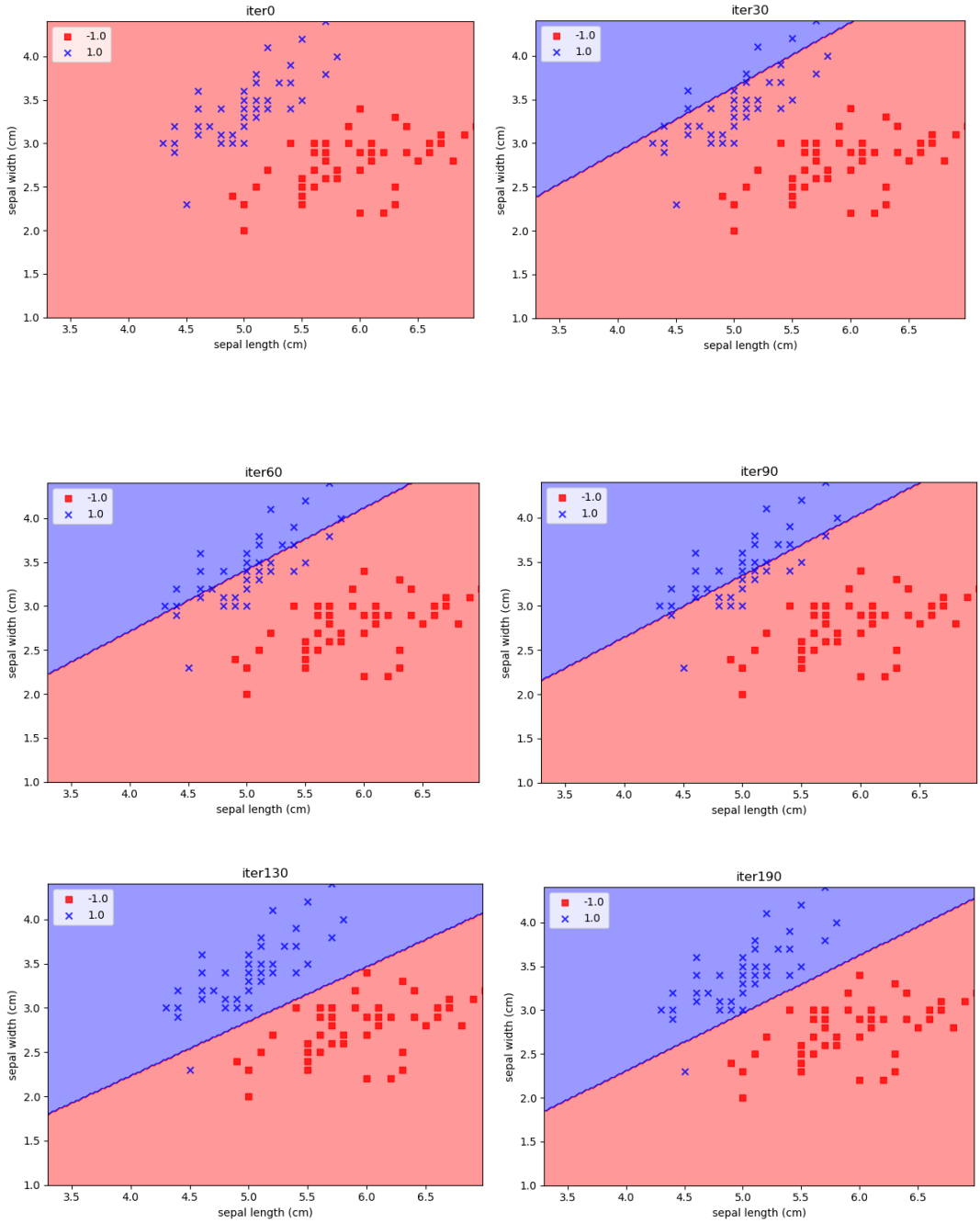
1. 用感知器算法对 Iris 鸢尾花数据进行两两分类

（1）数据线性可分

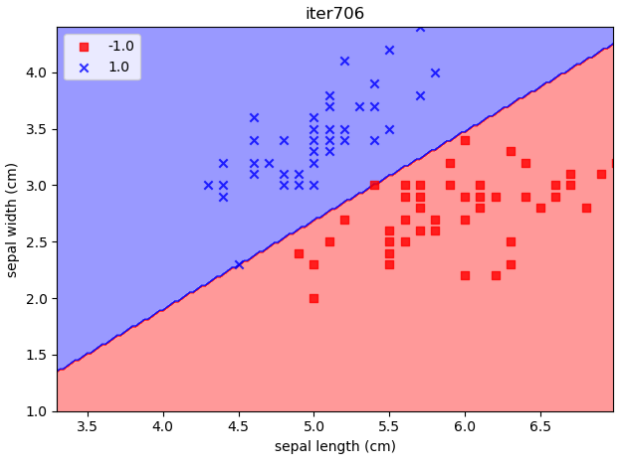
选取“setosa”和“versicolor”两类进行分类，为了达到线性可分地要求并更清晰地展示，可选取两个特征组成感知器的输入特征向量，这里选取“sepal length (cm)”和“sepal width (cm)”这两个特征；训练数据集选取 “setosa”和“versicolor”类的两个特征的全部数据，即总共 100 个训练样本，由于两个特征值情况下，训练数据有可能重复，所以对训练数据进去去重，最后的两类用于训练的数据分别为 38 和 44 个。用 matplotlib 库画出其中两种花的两个特征的关系：



设置感知器学习率为 0.01，迭代次数为 200 次，权值初始化为 0，每隔 10 次输出 w 和 b，并画出线性决策面：



可以看到随着训练迭代次数的增加，w 和 b 不断进行更新、线性决策面不断进行调整，图中的正确分类区域逐渐增大，最终得到一个能够对绝大多数样本进行分类的分离超平面，但是由于迭代次数较少，仍有 1 个点未能正确分类，如果设置更大的迭代次数，这个问题将会被解决，最后得到一个能对全部样本点进行正确分类的分离超平面。这种情况下，迭代到第 706 次时，感知器收敛，训练结束，如下图：

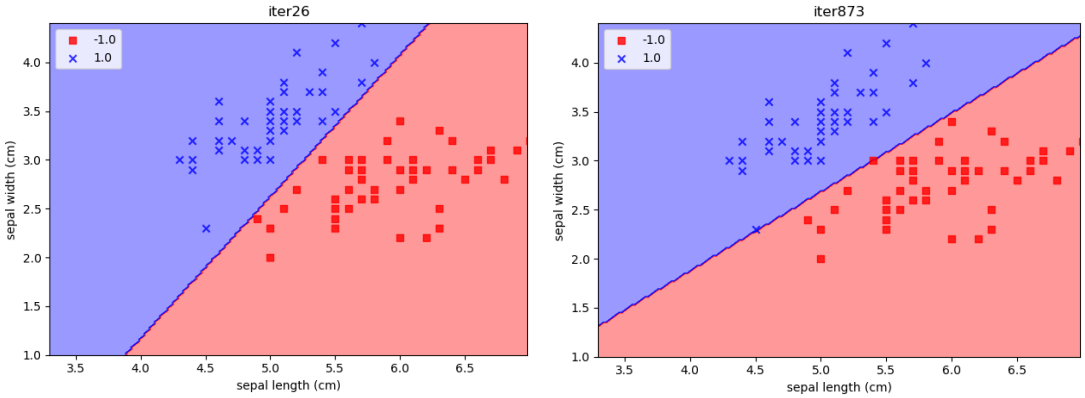


在以上实验的基础上，记录不同学习率下，感知器训练达到收敛所需的迭代次数，本次测试中 w 和 b 在 0 到 1 之间随机初始化（标准正态分布），运行 500 次，结果如下表：

学习率	达到收敛时的迭代次数
0.001	1267
0.01	624
0.1	619
0.9	715

可以看到学习率选取较小时（比如 0.001）或较大时（比如 0.9）感知器达到收敛所需的迭代次数更大，因此选择一个适合的学习率是至关重要的。

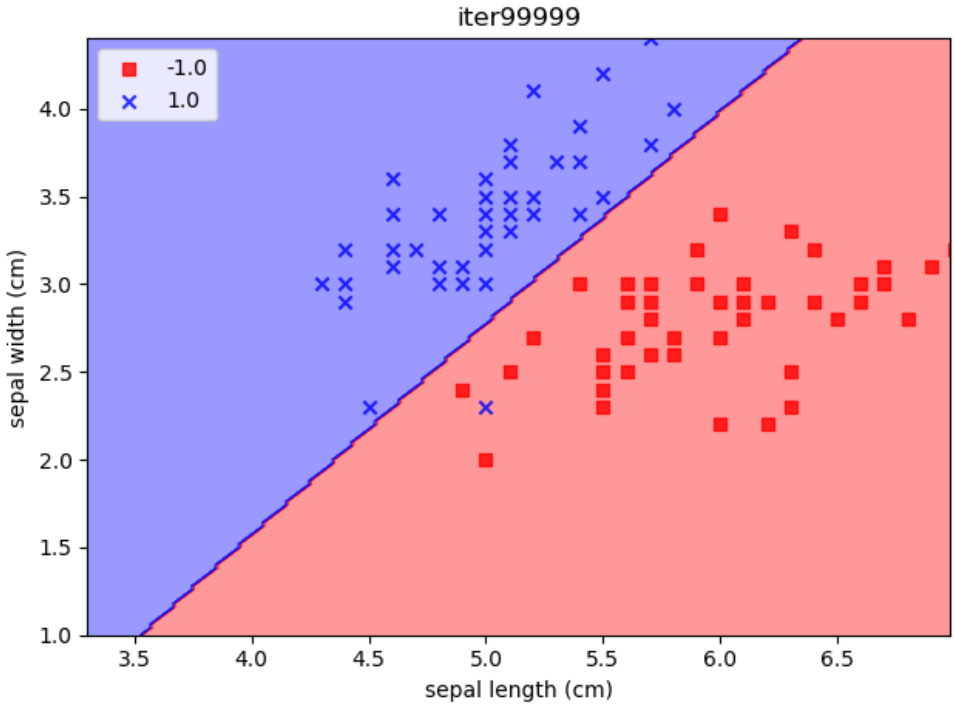
同样可以看到参数的初始值对感知器的训练结果和迭代次数有着较大的影响，下图是学习率为 0.01 时，两次不同初始化情况下的训练结果：



（2）数据线性不可分：

由于感知器算法是一种二分类线性模型，只能对线性可分的数据进行分类，如果数据线性不可分，感知器的训练无论迭代多少次都不会收敛，也就是找到一个分离超

平面。可以在以上数据中改变一个点使得数据线性不可分，然后设置迭代次数阈值为100000，观察感知器的训练是否能在此范围内达到收敛，结果是否定的，如下图：



2. 用 BP 算法对 Iris 鸢尾花数据进行分类

由于 Iris 数据集包含三类样本，可以使用 one-hot 编码，用[1,0,0]表示第一类（setosa），[0,1,0]表示第二类（versicolor），[0,0,1]表示第三类（virginica），输出神经元为 3 个。由于每个样本有 4 个特征，分别是：'sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)'，因此网络的输入神经元为 4 个，隐藏层层数和神经元个数可以自定义。下面实验采用每层 2 个单元的前向神经网络来进行，评价指标为 Accuracy，即正确分类的样本数占总分类样本数的比值。

（1） 数据划分对结果的影响

设置隐藏层数为 2，隐藏学习率为 0.1, epochs 为 10000，将全部数据 shuffle 后划分为训练集和测试集，ratio 表示训练数据所占比例，在每个 ratio 下的实验都进行 500 次，实验结果如下：

ratio	训练集	测试集	Accuracy
0.2	30	120	0.93929
0.4	60	90	0.94524
0.6	90	60	0.94356

0.8	120	30	0.94406

可以看到在这种设置下，随着 ratio 的增大，训练数据的量增加，Accuracy 的值随着训练数据量的增加而有一定的提高，但是 ratio 超过 0.4 之后，再增加训练数据的量对 Accuracy 没有影响。

（2） 隐藏层数对结果的影响：

设置划分 ratio 为 0.6，学习率为 0.1, epochs 为 10000，将隐藏层数设置为 2、4、6、8，分别测试 500 次，得到实验结果如下：

隐藏层数	Accuracy
2	0.94799
4	0.94629
6	0.94356
8	0.94510

可以看到隐藏层数的增加并未提升最后的 Accuracy 值。

（3） 训练 epochs 数目对结果的影响：

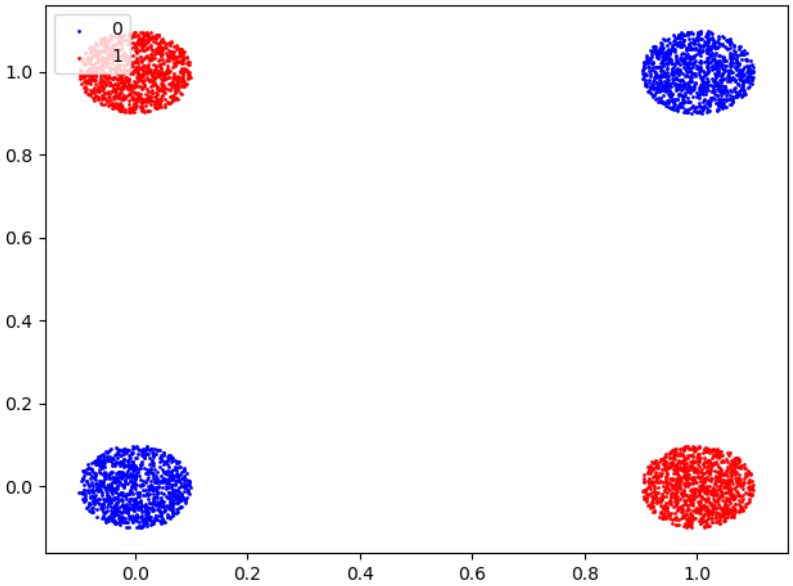
设置划分 ratio 为 0.6，学习率为 0.1，隐藏层数设置为 2，隐层单元数为 2，训练中的 epoch 数目分别设为 100、200、500、1000、5000、10000，分别测试 500 次，得到实验结果如下：

epoch 数目	Accuracy
100	0.64979
200	0.68033
500	0.69383
1000	0.73703
5000	0.93123
10000	0.94799

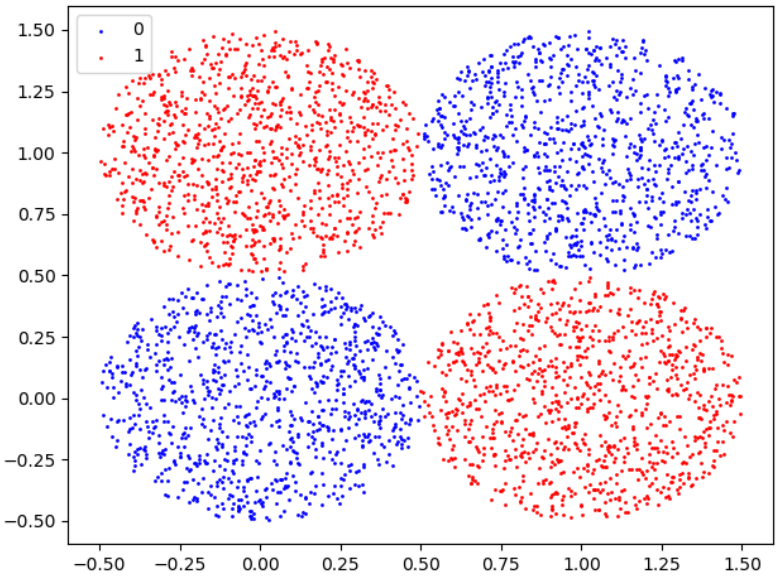
可以看到训练 epoch 的数目显著影响了最后的结果，因此在训练时需要找到一个适合的 epoch 数目，保证网络能够充分地训练。

3. BP 算法解决异或问题

（1）在异或问题中，4 个异或点分别为(0, 0)，(1, 1)，(1, 0)，(0, 1)，设定以这 4 个点为中心，r=0.1 为半径，随机均匀生成 N=1000 个数据点，其中(0,0)和(1,1)周围点为一类，(1,0)和(0,1)周围点为一类，生成数据如下图：



以 r=0.5 为半径，随机生成 N=1000 个数据点：

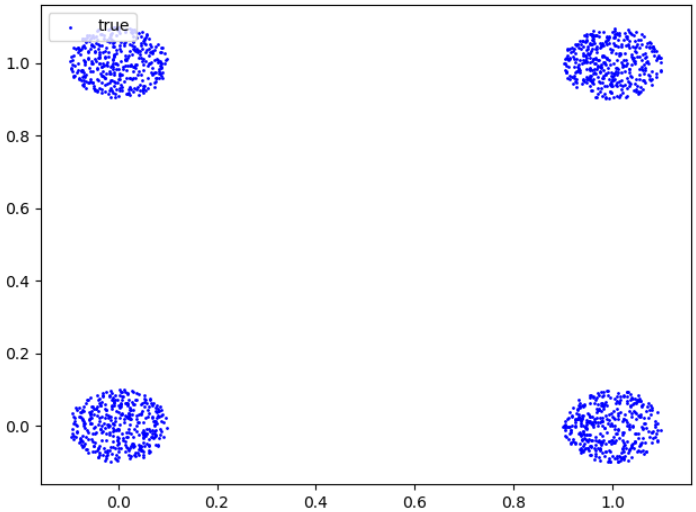


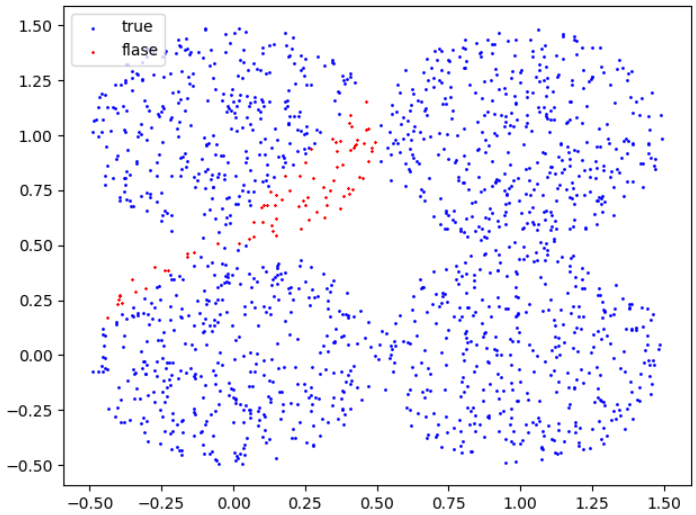
（2）半径 r 的值的选取对结果的影响：

设置随机生成数据点数 N=1000，训练集与测试集的划分比例为 6：4（即 ratio=0.4），网络结构为三层前向网络（输入层单元数为 2，一个包含 2 个单元的隐层，输出层单元数为 1），learning_rate=0.1, epochs=10000，跑 500 次求得最后的平均 Accuracy 值，实验结果如下：

半径 r	Accuracy
0.5	0.948455
0.4	0.9828625
0.3	0.9978225
0.2	0.99946
0.1	1.0

r=0.1 和 r=0.5 时，测试集上的分类结果如下（其中蓝色点为正确分类点，红色点为错误分类点）：





可以看出，随着随机生成数据点的半径的减小，对该数据集进行分类的难度变小，分类准确率逐渐增大，到 r=0.1 时，Accuracy=1，网络已经能对所有测试样本正确分类。

四、 实验结论

通过该次实验比较了线性分类器和非线性分类器，在 Iris 数据集上线性分类器和非线性分类器的性能差距并不是很大。而对异或数据线性分类器是无法完成分类任务的，非线性分类器则可以比较好地解决该问题。通过此次实验，进一步加深了对线性判别函数，非线性判别函数基本概念和方法、感知器算法，前馈神经网络与反向传播算法的理解。