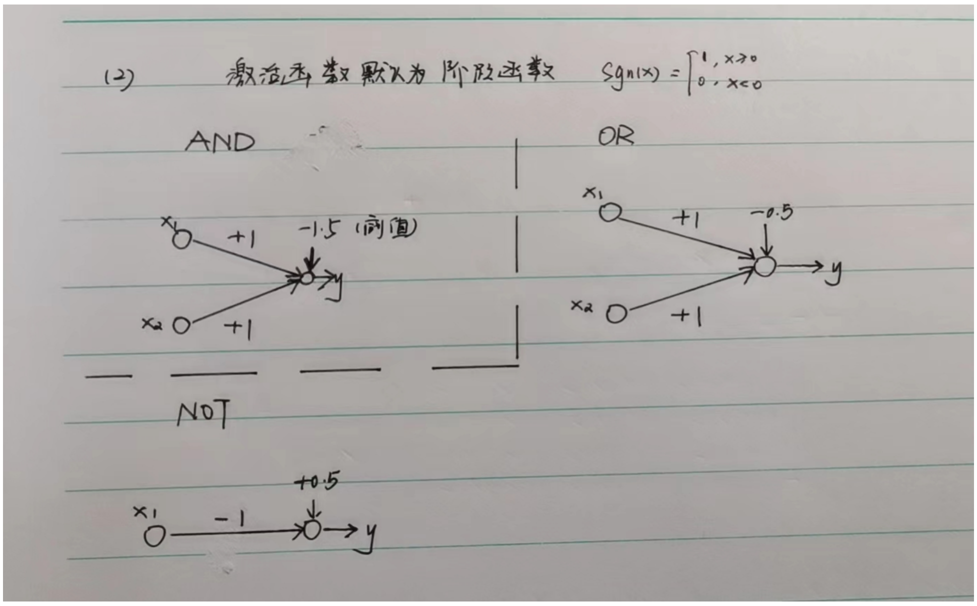
# 机器学习 Perceptron 作业

**姓名： 张泽群 学号： 19049100002 班级：1班**

Perceptron 3-1

答：（1）二元逻辑函数AND、NOT、OR是线性可分的，而XOR是非线性可分的

（2）

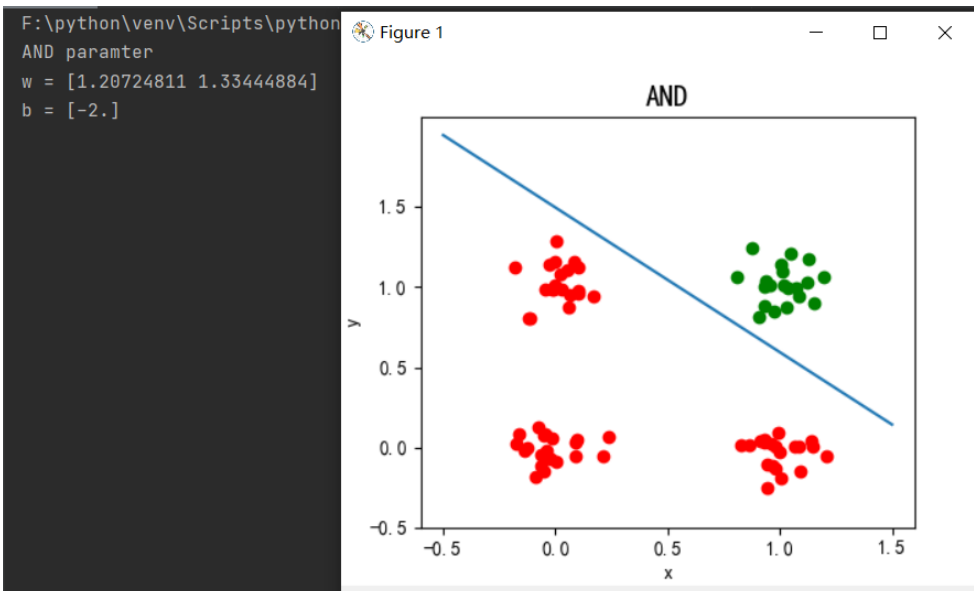


```python
# Perceptron 3-2


# -*- coding:utf-8 -*-

from sklearn.linear_model import Perceptron
from matplotlib.pylab import plt, multivariate_normal
import numpy as np

def pltshow(data,labels,title):
    int_labels=map(int,list(labels))
    colors=['r','g']
    x,y = data.T
    for index,label in enumerate(int_labels):
        plt.scatter(x[index],y[index],color=colors[label])
    plt.xlabel('x')
    plt.ylabel("y")
    my_x_ticks = np.arange(-0.5, 2, 0.5)
    my_y_ticks = np.arange(-0.5, 2, 0.5)
    plt.xticks(my_x_ticks)
    plt.yticks(my_y_ticks)
    plt.title(title, fontsize=15)
    plt.show()

def plot_line(w,b,x_lim):
    w = w.reshape(-1)
    Lx = np.linspace(x_lim[0],x_lim[1])
    Ly = -(w[0] *Lx + b) / w[1]
    plt.plot(Lx,Ly)

def create_and(sample_num,cov,mean):
    # AND 数据集
    posdata1 = np.random.multivariate_normal(mean[0], cov, sample_num)  # 正数据集
    posd = posdata1
    negdata1 = np.random.multivariate_normal(mean[1], cov, sample_num)  # 负数据集
    negdata2 = np.random.multivariate_normal(mean[2], cov, sample_num)
    negdata3 = np.random.multivariate_normal(mean[3], cov, sample_num)
    negd = np.vstack((negdata1, negdata2, negdata3))

    pos_num = posd.shape[0]  # 行数
    neg_num = negd.shape[0]
    labels = np.ones((pos_num + neg_num, 1))
    labels[pos_num:] = 0
    train_data = np.vstack((posd, negd))
    DataMat = np.array(train_data, dtype='float32')
    Labels = np.array(labels.reshape(-1))
    return DataMat,Labels

def create_or(sample_num,cov,mean):
    # AND 数据集
```

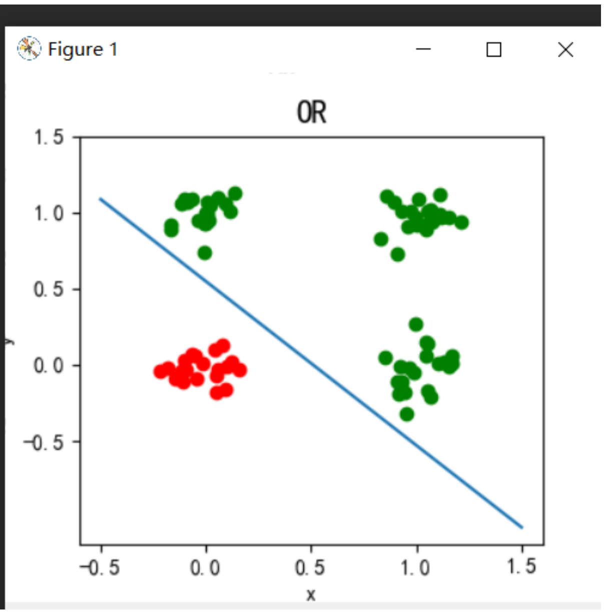
```python
    negdata1 = np.random.multivariate_normal(mean[3], cov, sample_num)  # 负数据集
    negd = negdata1
    posdata1 = np.random.multivariate_normal(mean[1], cov, sample_num)  # 正数据集
    posdata2 = np.random.multivariate_normal(mean[2], cov, sample_num)
    posdata3 = np.random.multivariate_normal(mean[0], cov, sample_num)
    posd = np.vstack((posdata1, posdata2, posdata3))

    pos_num = posd.shape[0]  # 行数
    neg_num = negd.shape[0]
    labels = np.ones((pos_num + neg_num, 1))
    labels[pos_num:] = 0
    train_data = np.vstack((posd, negd))
    DataMat = np.array(train_data, dtype='float32')
    Labels = np.array(labels.reshape(-1))
    return DataMat,Labels


if __name__ == '__main__':
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False

    sample_num=20
    sigma=0.01 # 样本偏移度
    cov = sigma * np.identity(2)
    mean = [[1, 1], [1, 0], [0, 1], [0, 0]]

    # AND
    DataMat, Labels = create_and(sample_num,cov,mean)

    p1 = Perceptron(max_iter=30, shuffle=False)
    p1.fit(DataMat, Labels)
    pre = p1.predict(DataMat)
    w1 = p1.coef_[0]
    b1 = p1.intercept_
    plot_line(w1,b1,[-0.5,1.5])
    print('AND paramter')
    print('w =',w1)
    print('b =',b1)
    pltshow(DataMat,pre,title="AND")

    # OR
    DataMat, Labels = create_or(sample_num, cov, mean)

    p2 = Perceptron(max_iter=30, shuffle=False)
    p2.fit(DataMat, Labels)
    pre = p2.predict(DataMat)
    w2 = p2.coef_[0]
    b2 = p2.intercept_
    plot_line(w2,b2,[-0.5,1.5])
    print('OR paramter')
    print('w =',w2)
```

```
print('b =',b2)
pltshow(DataMat,pre,title="OR")

# NOT
sample_num=20
sigma=0.01 # 样本偏移度
cov = sigma * np.identity(1)
mean = [[0],[1]]

negd = np.random.multivariate_normal(mean[0], cov, sample_num)  # 负数据集
posd = np.random.multivariate_normal(mean[1], cov, sample_num)  # 正数据集
pos_num = posd.shape[0]  # 行数
neg_num = negd.shape[0]
labels = np.ones((pos_num + neg_num, 1))
labels[pos_num:] = 0
train_data = np.vstack((posd, negd))
DataMat = np.array(train_data, dtype='float32')
Labels = np.array(labels.reshape(-1))

p3 = Perceptron(max_iter=30, shuffle=False)
p3.fit(DataMat, Labels)
pre = p3.predict(DataMat)
w3 = p3.coef_[0]
b3 = p3.intercept_

print('NOT paramter')
print('w =',w3)
print('b =',b3)
```

运行结果：



```
OR paramter
w = [1.9670655  1.82723979]
b = [-1.]
```




```
NOT paramter
w = [1.62737115]
b = [-1.]
```

从运行结果可见得到的perceptron模型的结果(参数归一化后)与问题3.1所写的模型相近，都可以完成AND\OR\NOT的功能。

Perceptron 3-3

(1). 如果采用一对多策略，至少需要用多少个二分类器来完成?

答：一对多策略每次将一个类的样例作为正例、所有其他类的样例作为反例来训练N个分类器。共有11个类别，则有11个二分类器。

(2). 所对应的编码矩阵是怎样的?

答：**从上到下对应** 0、1、2...10 **，从做到右对应**f1、f2、f3...f10

$$
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
$$

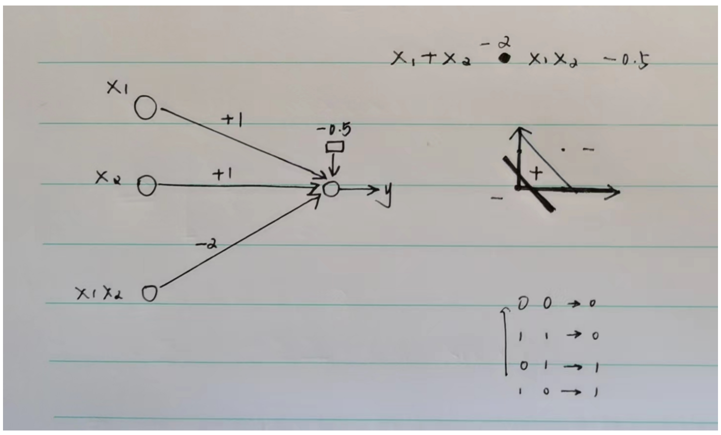
(3). 对于一个测试样本，怎样通过这些二分类器的判别结果，给出这个样本属于哪一类的决策？

答：将待测样本提交给所有的分类器预测，得到N个 (11) 分类结果。若仅有一个分类器预测为正类，则对应类别为最终的分类结果。但如果有多个分类器预测为正类，选择置信度最大的类别作为最终的分类结果。

Perceptron 3-4

两元逻辑函数XOR不是一个线性可分问题，因此该函数无法用Perceptron实现。请编程Perceptron学习算法，学习出这个XOR函数的广义Perceptron实现。

**模型：**



```python
# -*- coding:utf-8 -*-

from sklearn.linear_model import Perceptron
from matplotlib.pylab import plt, multivariate_normal
import numpy as np

def pltshow(data, labels, title):
    int_labels=map(int, list(labels))
    colors=['r', 'g']
    x, y, z = data.T
    for index, label in enumerate(int_labels):
        plt.scatter(x[index], y[index], color=colors[label])

    plt.xlabel('x')
    plt.ylabel("y")
    my_x_ticks = np.arange(-0.5, 2, 0.5)
    my_y_ticks = np.arange(-0.5, 2, 0.5)
    plt.xticks(my_x_ticks)
    plt.yticks(my_y_ticks)
    plt.title(title, fontsize=15)
    plt.show()

def plot_line(w,b,x_lim):
    w = w.reshape(-1)
    Lx = np.linspace(x_lim[0],x_lim[1])
    Ly = -(w[0] *Lx + b) / w[1]
    plt.plot(Lx,Ly)

def create_xor(sample_num,cov,mean):
    # XOR 数据集
    posdata1 = np.random.multivariate_normal(mean[1], cov, sample_num)  # 正数据集
    posdata2 = np.random.multivariate_normal(mean[2], cov, sample_num)
    posd = np.vstack((posdata1, posdata2))
    negdata1 = np.random.multivariate_normal(mean[0], cov, sample_num)  # 负数据集
    negdata2 = np.random.multivariate_normal(mean[3], cov, sample_num)
    negd = np.vstack((negdata1, negdata2))

    pos_num = posd.shape[0]  # 行数
    neg_num = negd.shape[0]
    labels = np.ones((pos_num + neg_num, 1))
    labels[pos_num:] = 0
    train_data = np.vstack((posd, negd))
    DataMat = np.array(train_data, dtype='float32')
    Labels = np.array(labels.reshape(-1))
    return DataMat,Labels

if __name__ == '__main__':
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
```

```python
sample_num=20
sigma=0.01 # 样本偏移度
cov = sigma * np.identity(3)
# 加入一维增广 x1*x2
mean = [[1, 1, 1], [1, 0, 0], [0, 1, 0], [0, 0, 0]]
# XOR
DataMat, Labels = create_xor(sample_num,cov,mean)

p1 = Perceptron(max_iter=100, shuffle=False)
p1.fit(DataMat, Labels)
pre = p1.predict(DataMat)
w1 = p1.coef_[0]
b1 = p1.intercept_
print('XOR paramter')
print('w =',w1)
print('b =',b1)
pltshow(DataMat,pre,title="XOR")
```

**运行结果：**