### 实验二 感知器模型实现并进行分类

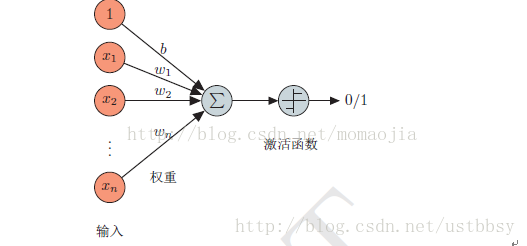
感知器，也可翻译为感知机，是Frank Rosenblatt在1957年就职于Cornell航空实验室(Cornell Aeronautical Laboratory)时所发明的一种人工神经网络。它可以被视为一种最简单形式的前馈式人工神经网络，是一种二元线性分类器。感知器是生物神经细胞的简单抽象，神经细胞结构大致可分为：树突、突触、细胞体及轴突。单个神经细胞可被视为一种只有两种状态的机器——激动时为‘是’，而未激动时为‘否’。神经细胞的状态取决于从其它的神经细胞收到的输入信号量，及突触的强度（抑制或加强）。当信号量总和超过了某个阈值时，细胞体就会激动，产生电脉冲。电脉冲沿着轴突并通过突触传递到其它神经元。为了模拟神经细胞行为，与之对应的感知机基础概念被提出，如权量（突触）、偏置（阈值）及激活函数（细胞体）。

**实验目的：**加深对感知器模型的理解，能够使用感知器模型解决简单的分类问题

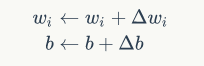
**实验原理：**参考课本对感知器模型的讲解

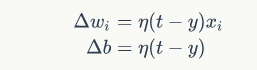
**实验内容：**根据感知器的相关知识，使用Python语言实现一个简单的感知器模型，该模型能够实现简单的二分类任务（与或非运算）。

感知器模型如下所示：



权重更新规则如下所示：





其中，t为label，y是预测，t-y=偏差，η为学习率，w为权重，b为偏置，本次实验需要使用到的包为numpy，可以使用pip安装。简单示例如下：

import numpy as np

#定义激活函数为阶跃函数

def acti\_fun(x):

return 1 if x > 0 else 0

#创建感知器类

class Perception(object):

#初始化权重

def \_\_init\_\_(self):

self.weights = np.random.random()

self.bias = 0

#定义训练函数，包括训练次数iter,学习率rate

def train(self,input\_vecs,labels,iter,rate):

for i in range(iter):

for input\_vec,label in zip(input\_vecs,labels):

output = acti\_fun(sum(np.array(input\_vec) \* self.weights) + self.bias)

bias = label – output#此处可以加判断语句使程序提前停止训练

#更新权重

self.weights += rate\*bias\*np.array(input\_vec)

self.bias += rate\*bias

return self.weights,self.bias

#定义预测函数

def predict(self,input\_data):

input\_data = np.array(input\_data)

pred = []

for each in input\_data:

pred\_each = acti\_fun(sum(np.array(each)\*

np.array(self.weights)) + self.bias)

pred.append(pred\_each)

return pred

#测试

if \_\_name\_\_=='\_\_main\_\_':

#训练数据

input\_vecs = [[1,1],[1,0],[0,1],[0,0]]

labels = [1,0,0,0]

#测试数据

inputdata = [[-1,0],[1,1]] #可以单个[1,0]，也可以多个[[-1,0],[1,1]]

#开始训练和测试

p = Perception()

p.train(input\_vecs,labels,50,0.1)

predict = p.predict(inputdata)

print(predict)

**实验要求：**

1. 基于与或非真值表构建训练数据，将构建好的训练数据存储到列表中。
2. 创建网络。
3. 二分类任务测试。将测试数据输入到创建并训练好的网络中，网络的输出是与该测试数据与或非运算最为接近的数据。