**感知器算法实验报告**

1. **问题描述**

编写感知器算法程序，求下列模式分类的解向量：

ω1: {(0 0 0)T , (1 0 0)T , (1 0 1)T , (1 1 0)T}

ω2: {(0 0 1)T , (0 1 1)T , (0 1 0)T , (1 1 1)T}

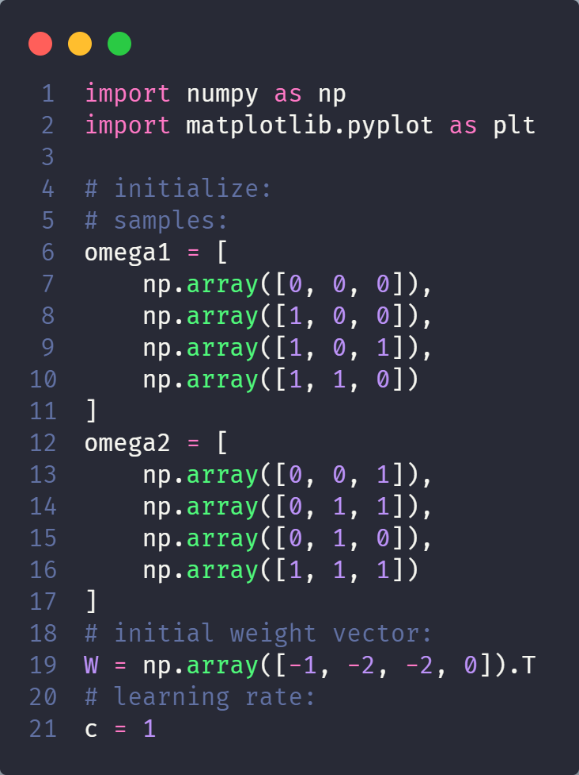
设w(1)=(-1 -2 -2 0)T

1. **算法描述 & 程序实现**

感知器算法是通过对已知类别的训练样本集的学习，寻找一个满足判别函数的权向量W。

**算法的具体步骤如下：**

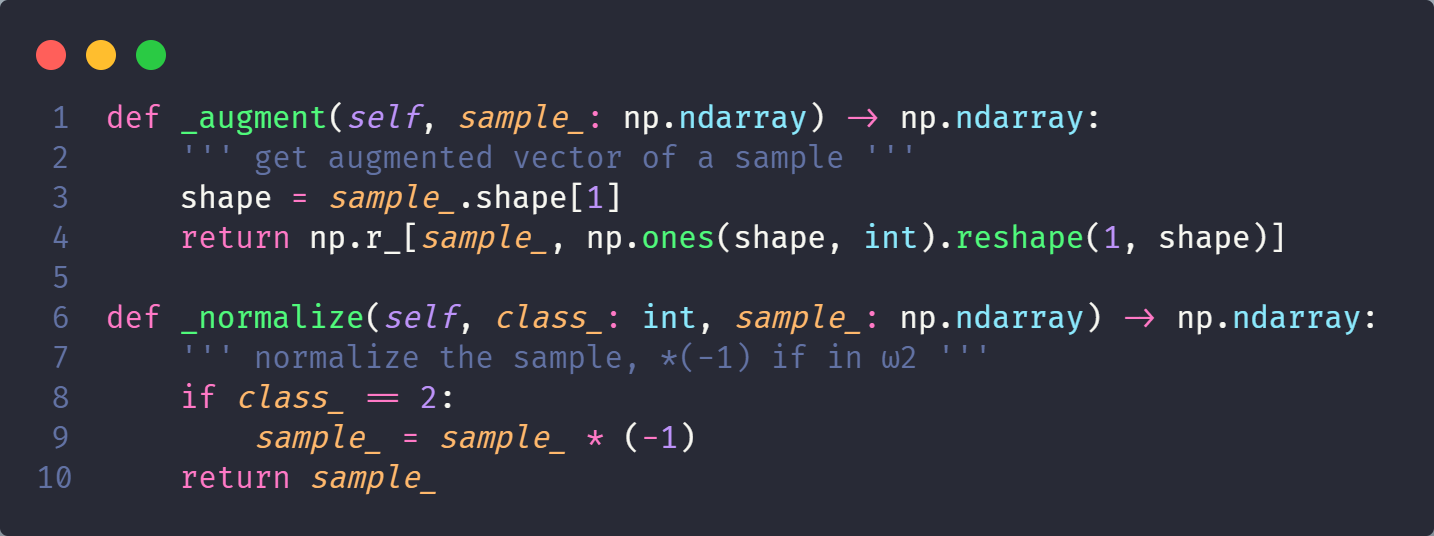
1. 库的导入、初始值的设定（包括已知类别的样本、初始权向量、矫正增量系数）：



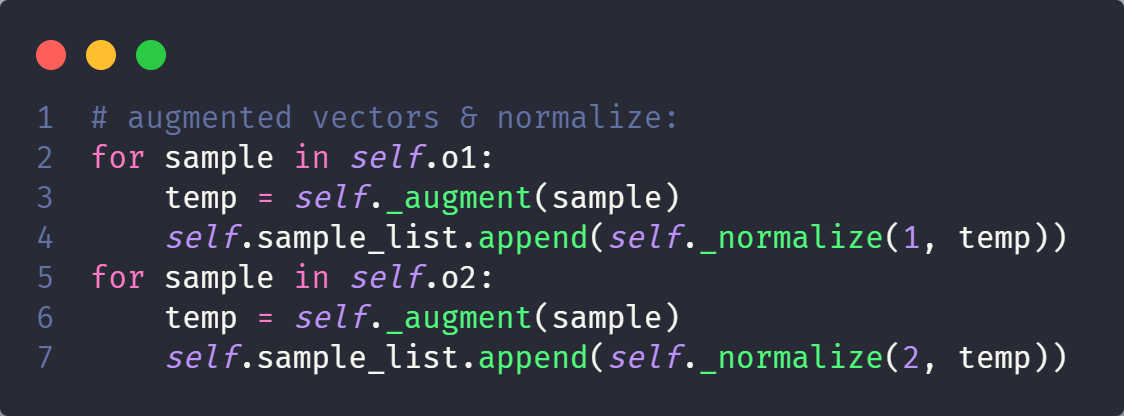
1. 将算法封装为一个类，各步骤作为类内函数实现，最后集合在一个solve()函数中，用于外部调用。类的初始化，即传递参数和声明类内成员变量：



1. 对样本进行预处理，即写为增广向量的形式并进行规范化处理：



在solve()函数中完成对样本的遍历，调用以上两函数进行处理：

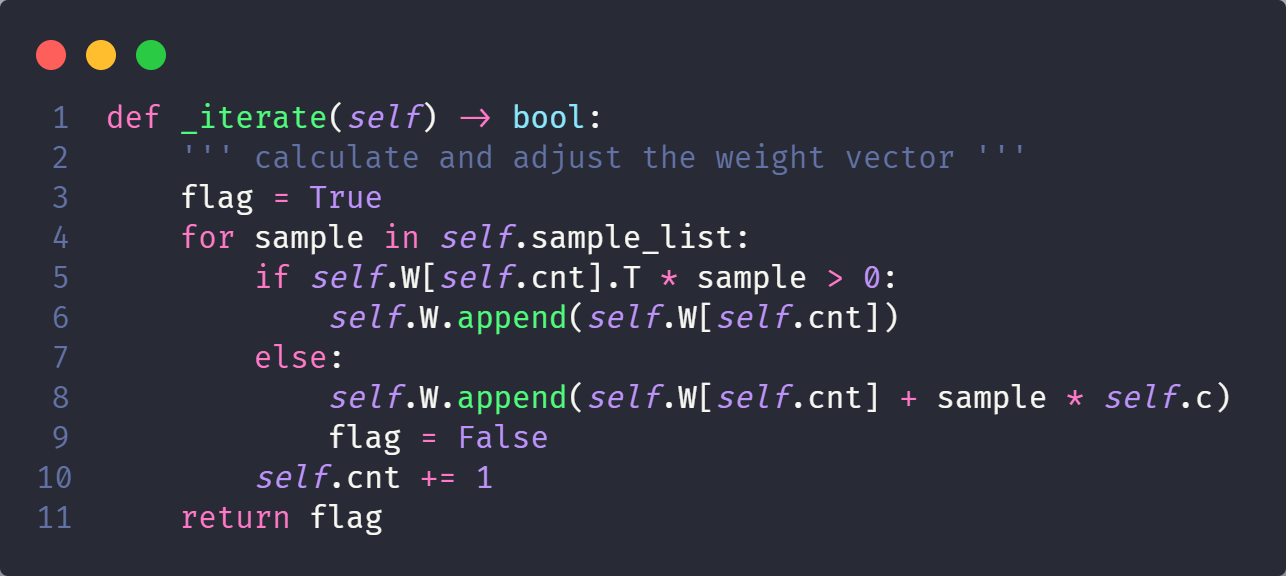


1. 进行迭代，遍历所有样本，计算判别函数WTX，根据结果的正负来调整权向量：

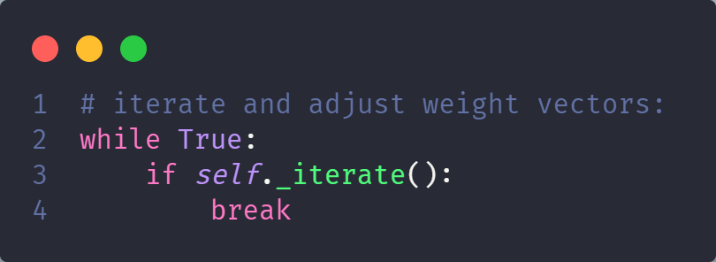
，则权向量不变；

，则。

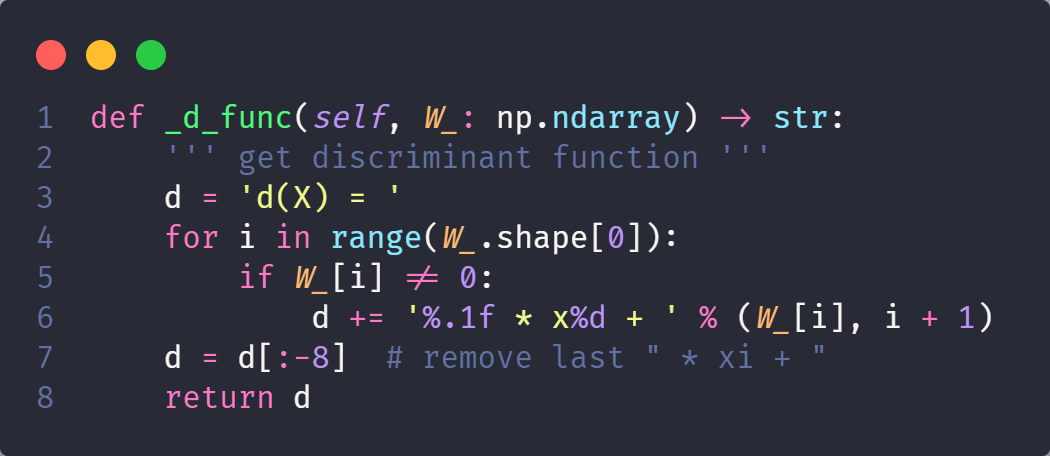
当一轮中没有错判，即判别函数计算结果均大于0时，算法结束：



在solve()函数中进行是否结束的判断：



1. 根据权向量计算结果，打印判别函数，便于画图：



1. 算法结束，画图展示结果：



（其中部分内容需要根据打印的计算结果手动调整参数来得到图像）

1. **计算过程**

以下是程序在控制台的输出，展示了每一轮迭代中的权向量的变化以及最后得到的判别函数：

W1: [[-1 -2 -2 0]] T

W2: [[-1 -2 -2 1]] T

W3: [[ 0 -2 -2 2]] T

W4: [[ 1 -2 -1 3]] T

W5: [[ 1 -2 -1 3]] T

W6: [[ 1 -2 -2 2]] T

W7: [[ 1 -2 -2 2]] T

W8: [[ 1 -3 -2 1]] T

W9: [[ 1 -3 -2 1]] T

W10: [[ 1 -3 -2 1]] T

W11: [[ 1 -3 -2 1]] T

W12: [[ 2 -3 -1 2]] T

W13: [[ 2 -3 -1 2]] T

W14: [[ 2 -3 -2 1]] T

W15: [[ 2 -3 -2 1]] T

W16: [[ 2 -3 -2 1]] T

W17: [[ 2 -3 -2 1]] T

W18: [[ 2 -3 -2 1]] T

W19: [[ 2 -3 -2 1]] T

W20: [[ 2 -3 -2 1]] T

W21: [[ 3 -2 -2 2]] T

W22: [[ 3 -2 -3 1]] T

W23: [[ 3 -2 -3 1]] T

W24: [[ 3 -2 -3 1]] T

W25: [[ 3 -2 -3 1]] T

W26: [[ 3 -2 -3 1]] T

W27: [[ 3 -2 -3 1]] T

W28: [[ 3 -2 -3 1]] T

W29: [[ 3 -2 -3 1]] T

W30: [[ 3 -2 -3 1]] T

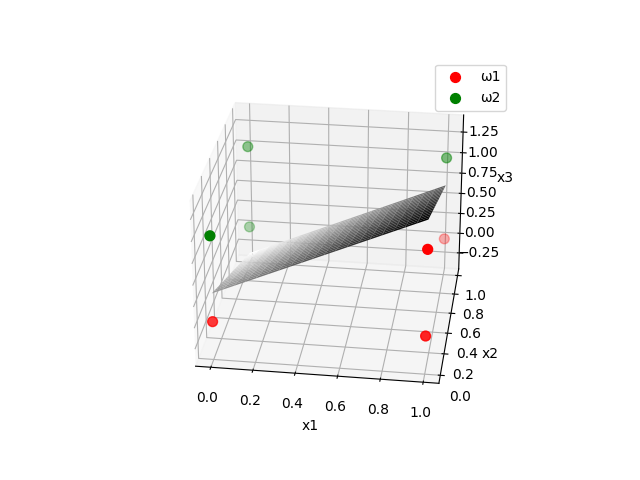
W31: [[ 3 -2 -3 1]] T

W32: [[ 3 -2 -3 1]] T

W33: [[ 3 -2 -3 1]] T

d(X) = 3.0 \* x1 + -2.0 \* x2 + -3.0 \* x3 + 1.0

1. **结果分析**



从结果图来看，计算得到的判别平面能够将两类样本（分别以红色、绿色标注）很好地分开。