1. **自己提出的问题的理解：**

问题1：感知机的普通形式和对偶形式有什么区别？换句话说为什么要有对偶模式？我感觉对偶形式不就是批次训练么？

讨论后的理解：对偶形式是与普通方式相对的另一种方式，也算是对原始方式的一种优化：样本点的特征向量以内积的形式存在于感知机对偶形式的训练算法中，凡是涉及到矩阵，向量内积的运算量就非常大，此时如果事先计算好所有的内积，以后碰到需要这个值的时候直接去矩阵里面寻找这个值就好了，大大减小了时间复杂度。

问题2：P44页中为什么说更新次数越多，对学习结果造成的影响越大？

讨论后的理解：对于一个离分割线比较远的点，他和分割线的距离越大，也就说明分割线对他的分类约有自信，也就是说这个点被错误分类的可能性就越小，此时分割线不需要根据这个点作改变。而相反，对于离分割线比较近的点，分割线对它的分类正确的自信非常小，也就是说这个点更有可能是一个被错误分类的点，此时分割线就越容易被这个点所“调整”，所以更新次数越多，对学习成果造成的影响也就越大。

1. **别人提出的问题的理解：**

问题1：感知机与SVM的区别是什么？

自己的理解：

感知机采用的是函数间隔，而SVM使用的是几何间隔。

SVM希望能够把边界点到分割线的距离最小化，而感知机则没有这么严格，它只要求被错误分类的点的数目能够达到0即可。

2、问题2：感知机学习中，在定义损失函数时，为什么可以直接忽略w的L2范数的倒数从而得到该损失函数？

自己的理解：首先，感知机的优化原理是把所有错误分类的点全部调整到正确分类，这在损失函数上表现出来就是所有的损失函数都能够不小于0，而因为w的范数是一个绝对的正数，所以它不会对损失函数的正负产生影响，所以去掉也就没有什么关系了。

3、问题3：普通感知机只能处理线性可分问题，那线性不可分的模型怎么处理

自己的理解：普通感知机模型可以使用的一个先决条件就是需要数据是线性可分的，如果不是线性可分的就会出现永远都会有点使得损失函数小于0。而对于线性不可分的数据，可以使用口袋算法，这个算法的改进非常简单，就是将目前最好的分割线握在自己手里，然后如果有更好的参数w(t+1)，可以使得当前被错误分类的点更少， 那么将原参数w(t)换成新的参数w(t+1)，最后设置一个迭代次数即可。

1. 读书计划

1、本周完成的内容章节：《统计学习方法》第2章

2、下周计划：《统计学习方法》第3章

1. 读书摘要及理解或伪代码的具体实现

**(1)读书摘要：**

2.1 感知机模型

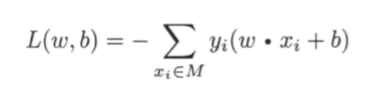
y=f(wx+b)

是一个二分类模型，和支持向量机类似，将空间中的点分类成为+-样例。

2.2 感知机学习策略

首先前提条件是数据集的线性可分性。

学习策略是使用所有分类错误的数据，最小化损失函数



正确分类的数据对模型的优化不会做贡献。

2.3 感知机学习算法

问题转化为最小化损失函数，使用梯度下降法。

当前理解的感知机和支持向量机的区别是：感知机约束条件仅仅是数据线性可分，而支持向量机则在此基础上需要满足约束条件，以使得最终得到的分界面是唯一的。

P44页中之所以说实例点更新次数越多，意味着它距离分离超平面越近，我的理解是因为wb得到最后的一个固定的值，学习率也是一个固定的值，更新的次数越多，说明它最终累加的次数越多，所以x和y都会相对较小。另一个理解是因为离得很近，所以每次更新的值就会非常小，所以更新次数多，两个等价。至于为什么越难正确分类也就对学习结果影响最大，我的理解是这样的：容易分类的数据比较“普通”，也就是具有非常多相似的数据，整个数据的分布主要由这些“普通”数据决定，所以最终学习得到的分界线是一个“差不多”的一个结果，而离分界线近的则因为分类特征不够明显，所以为了能够将这些数据很好的分类，感知机需要一点一点调整自己，尽可能的将这些数据也能正确分类，所以这类数据对最终的结果影响要多。

1. **代码实现**

w=0.0

b=0.0

datas=[(1,1),

(2,2),

(3,3),

(4,4),

(4,5),

(0,1),

(0,0.5),

(0.3,0.5),

(3,5),

(1.2,2.3)]

while True:

    f=0

    for item in datas:

        if (w\*item[0]+b)\*item[1]<=0:

            f=1

            w+=item[0]\*item[1]

            b+=item[1]

    if f==0:

        break

print(w,b)

**输出：W=1.0 b=1.0**