1. **自己提出的问题的理解：**

**问题1：**p132，图7.6中0-1损失函数在间隔大于0的部分是什么样的。

**讨论后的理解：**0-1损失函数即为“判断正确”的话损失函数的值为0，“判断错误”的话损失函数的值为1，所以在图中左边的部分为一条恒等于1 的直线，右边的部分则是和坐标轴重合的直线。

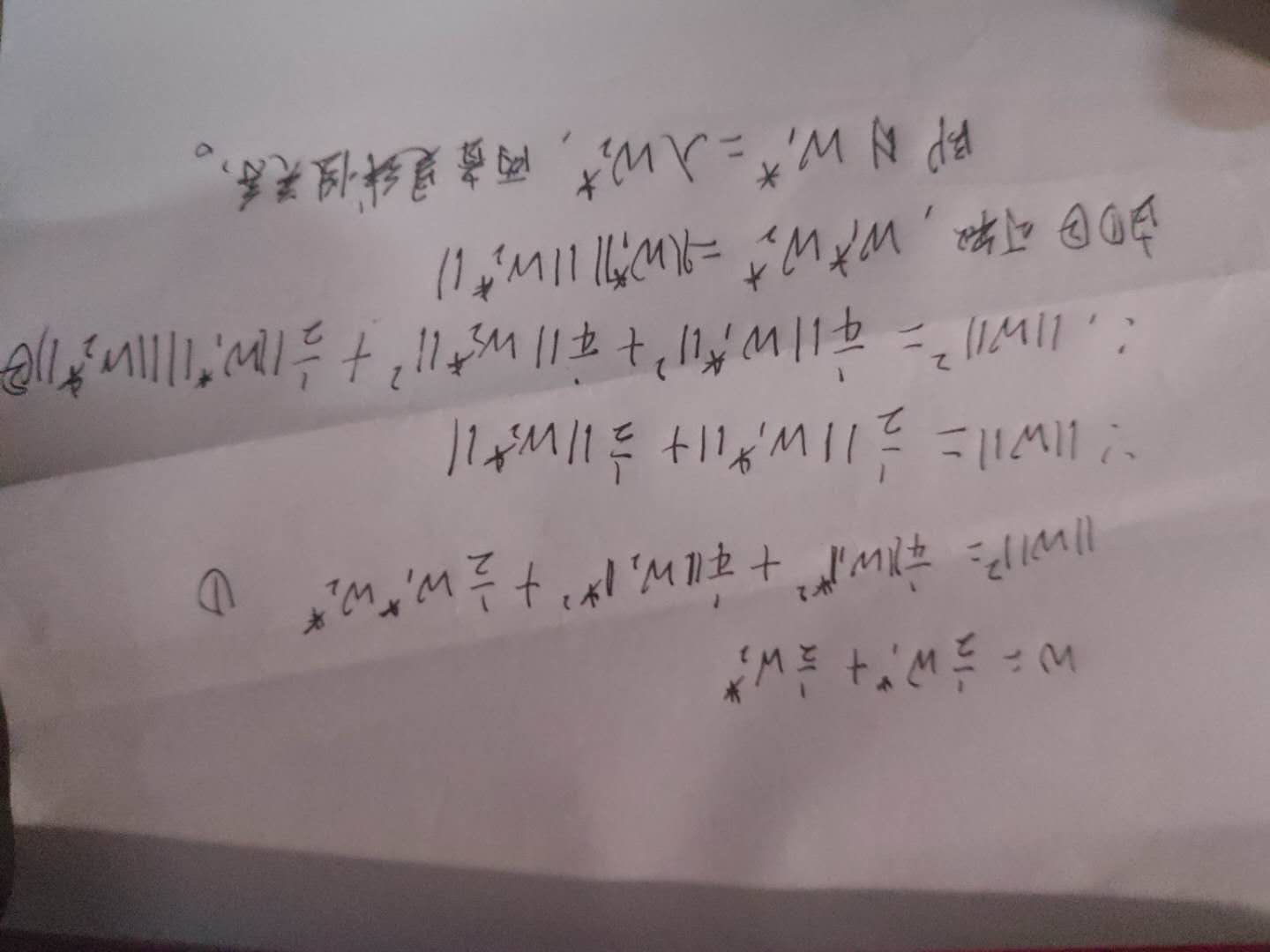
**问题2：**p131，（7.58）为什么式中要有一个正则化项？

**讨论后的理解：**和本章开始的时候的线性支持向量机相同，因为函数距离在求分类超平面的时候会出现只要等比例增加w和b就能在增大函数距离的同时而不改变超平面，这里也是类似，如果仅仅使用前面的损失函数，也会出现同样的问题，所以可以在后面增加一个w的l2范数来避免这种情况。

1. **别人提出的问题的理解：**

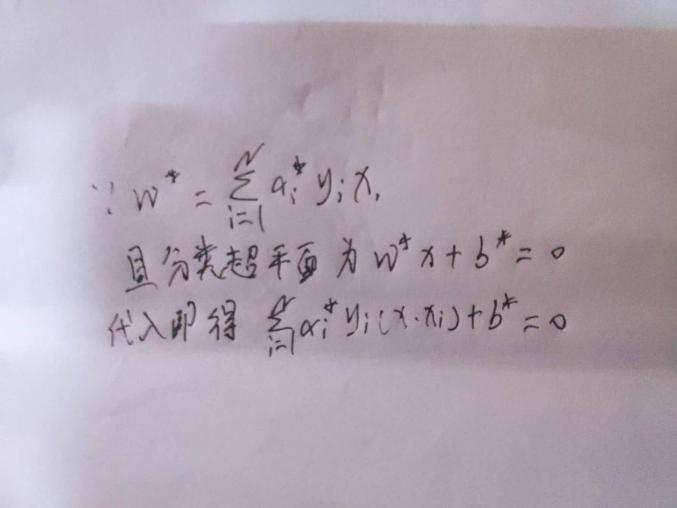
**问题1：**p117页，为什么从而有w\*1=lamda w\*2

**讨论后的理解：**



**问题2：**p122页，分离超平面为什么可以写成这个式子？

**讨论后的理解：**

****

书中的该式子上面的式子只是在证明b的值，在这个结论中省略了b的展开。

**问题3：**P115页，几何间隔是实例点到超平面的距离，什么叫带符号的距离？

**讨论后的理解：**带负号的距离是指当样本点被正确分类的时候最后的结果是正的，当样本点被错误分类的时候最后的结果是负的，所以可以说成是样本点与分类超平面之间的距离是带符号的。

1. **读书计划**

1、本周完成的内容章节：《统计学习方法》前两节

2、下周计划：《统计学习方法》第七章剩余部分

1. **读书摘要及理解或伪代码的具体实现**
2. **读书摘要：**

1、欧式空间：欧几里得空间就是在对现实空间的规则抽象和推广（从n<=3推广到有限n维空间）

2、函数间隔：y\*(wx+b)

函数间隔表示分类预测的正确性和确信度，但是最大化函数间隔并不可行，因为如果要最大化它的话，如果将w和b都增大，这个时候函数间隔变大，但是超平面却没有变。

3、支持向量

原始最优化问题及对偶最优化问题中将训练数据集中对应于α＞0的样本点的实例称为支持向量，因为支持向量是离分类超平面最近的向量，所以只有支持向量对分类超平面的值有作用。

1. **代码实现**

import pandas as pd

import numpy as np

import csv

import codecs

header = ['text', 'label']

data = [

("苹果","1"),

("桃子","1"),

("西瓜","1"),

("火龙果","1"),

("芒果","1"),

("香蕉","1"),

("电路板","0"),

("电脑","0"),

("机箱","0"),

("主机","0"),

("显示器","0"),

("适配器","0"),

("充电器","0"),

("手机","0"),

]

with open('train.csv', 'w', encoding='utf-8') as f:

writer = csv.writer(f)

writer.writerow(header)

writer.writerows(data)

def get\_data(path):

dataset = pd.read\_csv(path)

dataset = np.array(dataset)

train = dataset[:,0:dataset.shape[1]-1]

labels = dataset[:,-1]

return train,labels

train,labels = get\_data("train.csv")

class NaiveBayes(object):

def \_\_init\_\_(self,lamda,train,labels):

self.label\_map = {1:"水果",0:"机器"}

self.lamda = lamda

self.train = train

self.labels = set(labels)

#label\_prob\_pre : 每个类别的先验概率，即Pr(cj)

labels = list(self.labels)

label\_set = set(labels)

self.label\_prob\_pre = {}

for label in label\_set:

self.label\_prob\_pre[label] = labels.count(label)/float(len(labels))

#probs\_wc:Pr(w|c)

word\_list = [word for sen in self.train for word in sen[0]]

self.words\_set = set(word\_list)

self.probs\_wc = {}

for c in labels:

for word in self.words\_set:

self.probs\_wc[(word,c)]=self.get\_prob\_wc(word,c)

def get\_prob\_wc(self,word,c):

#求Pr(w|c)

num\_word\_c = 0

num\_c = 0

v = len(self.words\_set)

for i,data in enumerate(train):

label = labels[i]

if label == c:

num\_c += len(data[0])

num\_word\_c += data[0].count(word)

# print(num\_c)

return (num\_word\_c+self.lamda)/float(self.lamda\*v+num\_c)

#def get\_multi\_wc(self):

def classify(self,d):

mul\_dc\_sum = 0 #分母上对每个c，求每个word的乘积

for label in self.labels:

mul\_dc\_tmp = 1

for w in d:

mul\_dc\_tmp \*=self.probs\_wc[(w,label)]

mul\_dc\_sum +=mul\_dc\_tmp \* self.label\_prob\_pre[label]

results = []

for label in self.labels:

mul\_dc = 1

for w in d:

mul\_dc \*= self.probs\_wc[(w,label)]

results.append(self.label\_prob\_pre[label] \* mul\_dc)

print(results)

res\_c = np.argmax(results)

return self.label\_map[res\_c]

nb = NaiveBayes(0.5,train,labels)

print(nb.classify("电器"))

print(nb.classify("香瓜"))

运行结果：

[0.0057988165680473375, 0.00019223375624759708]

机器

[0.0001183431952662722, 0.0017301038062283738]

水果