1. （必填）自己提出的问题的理解（罗列全部）：
2. 提出的问题1：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

讨论后的理解： 口袋算法（Pocket Algorithm）可以用来求线性不可分模型的 “次优解”，

其算法思想是在迭代改变参数的过程中寻求尽可能少犯错误的解，如果找到则更新参数。缺点是两个参数在一次比较中就需要遍历两遍数据集。

提出的问题2：感知机一定收敛嘛，收敛又是否是算法的最优策略

讨论后的理解：线性可分的条件下感知机算法一定收敛，具体证明见李航书41页。然而，由于感知机算法不是为了产生大间隔，而是为了尽可能少的犯错误，这导致了感知机算法在接近收敛时容易过拟合，因此存在噪声时有必要采取“早停”策略以增强模型的泛化效果。

1. （必填）别人提出的问题的理解（选择几个问题罗列，并给出理解）：
2. 问题3：感知机与SVM的区别是什么？

自己的理解：1）普通的感知器不能产生大间隔(margin)，而SVM可以

2）带margin的感知器可以通过两种手段产生大间隔：一种是早停，另一种是加正则化项。

3）加入了L2正则化项的带 margin的感知器就是SVM。

问题4：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： 1）1/||w||不影响-y(w,x+b)正负的判断，即不影响学习算法的中间过程。因为感知机学习算法是误分类驱动的，这里需要注意的是所谓的“误分类驱动”指的是我们只需要判断-y(wx+b）的正负来判断分类的正确与否，而1/||w||并不影响正负值的判断。所以1/||w||对感知机学习算法的中间过程可有可无；

2）1/||w||不影响感知机学习算法的最终结果。因为感知机学习算法最终的终止条件是所有的输入都被正确分类，即不存在误分类的点。则此时损失函数为0. 对应于-y（wx+b）/||w||，即分母为0.则可以看出1/||w||对最终结果也无影响。

 提出的问题5：感知机对偶形式相对于普通形式的优点？

讨论后的理解：对偶形式在模型训练之前就将所有训练数据间的点积计算好，生成一个矩阵，这样在对偶形式进行预测时便可以直接按照索引在矩阵中找寻对应的点积结果，这样一来对偶形式无论在参数优化还是模型预测时，其所需的计算资源都要比原始形式要小。

1. （必填）读书计划

1、本周完成的内容章节：李航书第二章

2、下周计划：自由阅读李航书第三章，复习KNN算法

四、（选做）读书摘要及理解或伪代码的具体实现（读书摘要、伪代码的具体实现代码等可以写到这个部分）

1、读书摘要及理解（选做）

1)感知机是根据输入实例的特征向量α对其进行二类分类的线性分类模型:

f(z)= sign(w· x+b)，

感知机模型对应于输入空间(特征空间）中的分离超平面w·x＋b=0.

2)感知机学习的策略是极小化损失函数:

minL(w,b)=

损失函数对应于误分类点到分离超平面的总距离。

3)感知机学习算法是基于随机梯度下降法的对损失函数的最优化算法，有原始形式和对偶形式。算法简单且易于实现。原始形式中，首先任意选取一个超平面，然后用梯度下降法不断极小化目标函数。在这个过程中一次随机选取一个误分类点使其梯度下降。

4)当训练数据集线性可分时，感知机学习算法是收敛的。感知机算法在训练数据

集上的误分类次数k满足不等式:

当训练数据集线性可分时，感知机学习算法存在无穷多个解，其解由于不同的初

值或不同的迭代顺序而可能有所不同。

2.代码实现

import pandas as pd

import numpy as np

from sklearn.datasets import load\_iris

import matplotlib.pyplot as plt

#

# load data

iris = load\_iris()

#这里不借助DataFrame也可以，直接把X, y = iris.data, iris.target

df = pd.DataFrame(iris.data, columns=iris.feature\_names)

df['label'] = iris.target

# print(df.columns)

#'sepal length (cm)', 'sepal width (cm)', 'petal length (cm)','petal width (cm)', 'label' 原来的列名

df.columns = ['sepal length', 'sepal width', 'petal length', 'petal width', 'label']

plt.figure(figsize=(15, 8))

plt.subplot(131)

plt.scatter(df[:50]['sepal length'], df[:50]['sepal width'], label='0')

plt.scatter(df[50:100]['sepal length'], df[50:100]['sepal width'], label='1')

plt.xlabel('sepal length')

plt.ylabel('sepal width')

plt.title('original data')

plt.legend()

data = np.array(df.iloc[:100, [0, 1, -1]]) #这里不能用df[:100][0, 1, -1]

X, y = data[:,:-1], data[:,-1]

y = np.array([1 if i == 1 else -1 for i in y])

# 数据线性可分，二分类数据

# 此处为一元一次线性方程

class Model:

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

self.w = np.ones(len(data[0]) - 1, dtype=np.float32)

self.b = 0

self.l\_rate = 0.1

# self.data = data

def sign(self, x, w, b):

y = np.dot(x, w) + b

return y

# 随机梯度下降法

def fit(self, X\_train, y\_train):

is\_wrong = False

while not is\_wrong:

wrong\_count = 0

for d in range(len(X\_train)):

X = X\_train[d]

y = y\_train[d]

if y \* self.sign(X, self.w, self.b) <= 0:

self.w = self.w + self.l\_rate \* np.dot(y, X)

self.b = self.b + self.l\_rate \* y

wrong\_count += 1

if wrong\_count == 0:

is\_wrong = True

return 'Perceptron Model!'

perceptron = Model()

perceptron.fit(X, y)

x\_points = np.linspace(4, 7, 10)

y\_ = -(perceptron.w[0]\*x\_points + perceptron.b)/perceptron.w[1]

plt.subplot(132)

plt.plot(x\_points, y\_)

plt.plot(data[:50, 0], data[:50, 1], 'bo', color='blue', label='0') #用scatter也可以

plt.plot(data[50:100, 0], data[50:100, 1], 'bo', color='orange', label='1')

plt.xlabel('sepal length')

plt.ylabel('sepal width')

plt.title('perceptron byhand')

plt.legend()

from sklearn.linear\_model import Perceptron

clf = Perceptron(fit\_intercept=True, max\_iter=1000, shuffle=False) #fit\_intercept表示是否保留截距

clf.fit(X, y)

# Weights assigned to the features.

print(clf.coef\_) #二维array

# 截距 Constants in decision function.

print(clf.intercept\_)

x\_ponits = np.arange(4, 8)

y\_ = -(clf.coef\_[0][0]\*x\_ponits + clf.intercept\_)/clf.coef\_[0][1]

plt.subplot(133)

plt.plot(x\_ponits, y\_)

plt.plot(data[:50, 0], data[:50, 1], 'bo', color='blue', label='0')

plt.plot(data[50:100, 0], data[50:100, 1], 'bo', color='orange', label='1')

plt.xlabel('sepal length')

plt.ylabel('sepal width')

plt.title('perceptron by sklearn')

plt.legend()

plt.subplots\_adjust(top=0.92, bottom=0.08, left=0.10, right=0.95, hspace=0.25,

wspace=0.35) #调整子图间距

plt.savefig("demo.jpg")

plt.show()