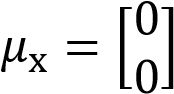
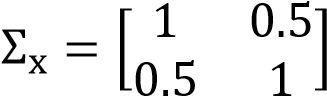
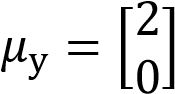
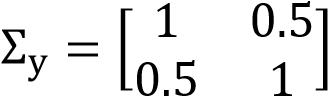
HW #5 Due: 5/16/2022

資工碩一 呂昀星 110598056

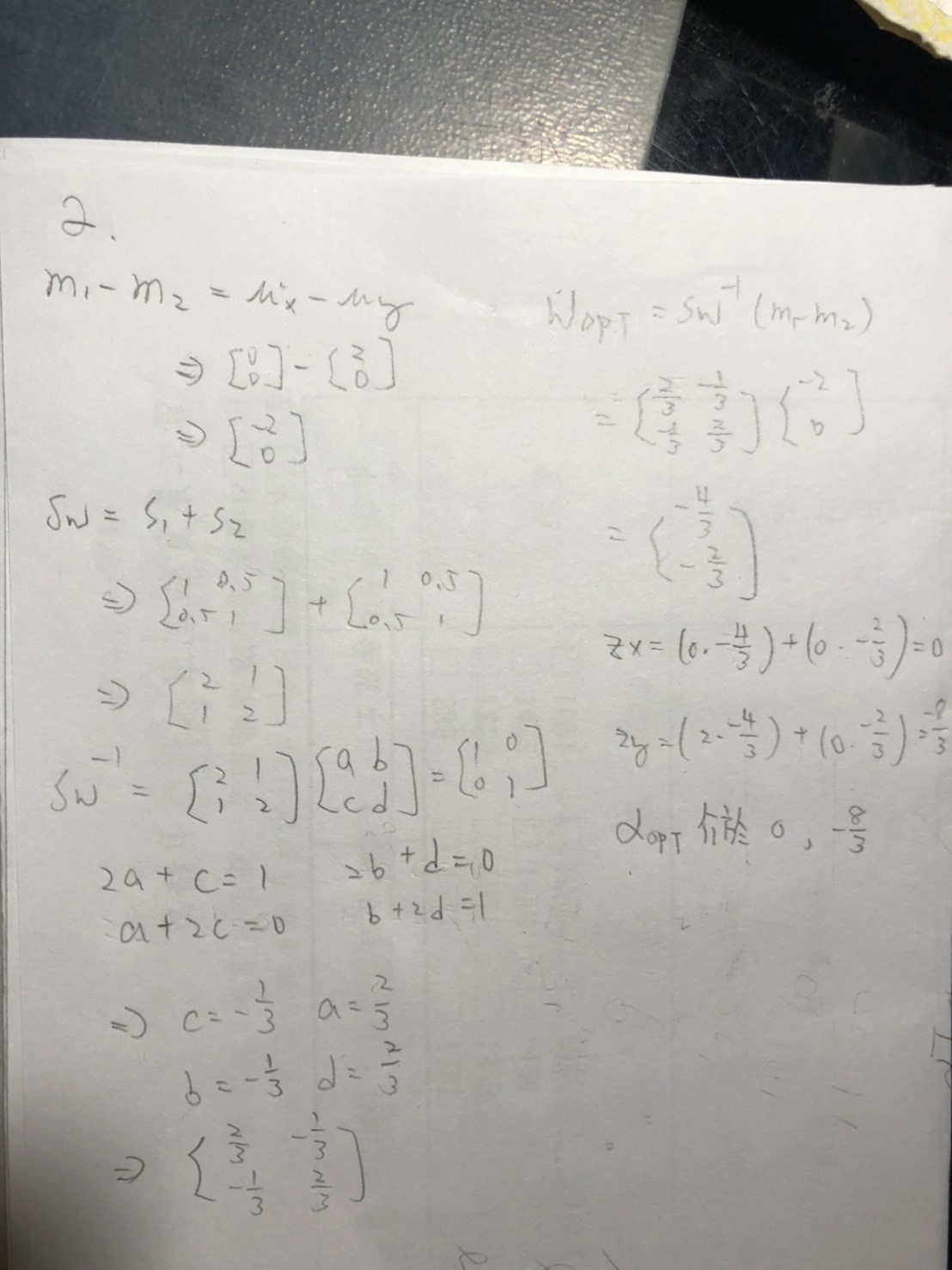
1. If we want to use the approach given on the lecture notes to perform dimensionality reduction based on ICA, can we directly pick independent components with larger energy? Explain.

Yes. To use ICA for dimensionality reduction, a simple method is to use components with larger energy (during whiting) as features

1. Suppose we have two classes of data. The samples in the first class are from  where x1 and x2 are jointly Gaussian random variables (r.v.) with

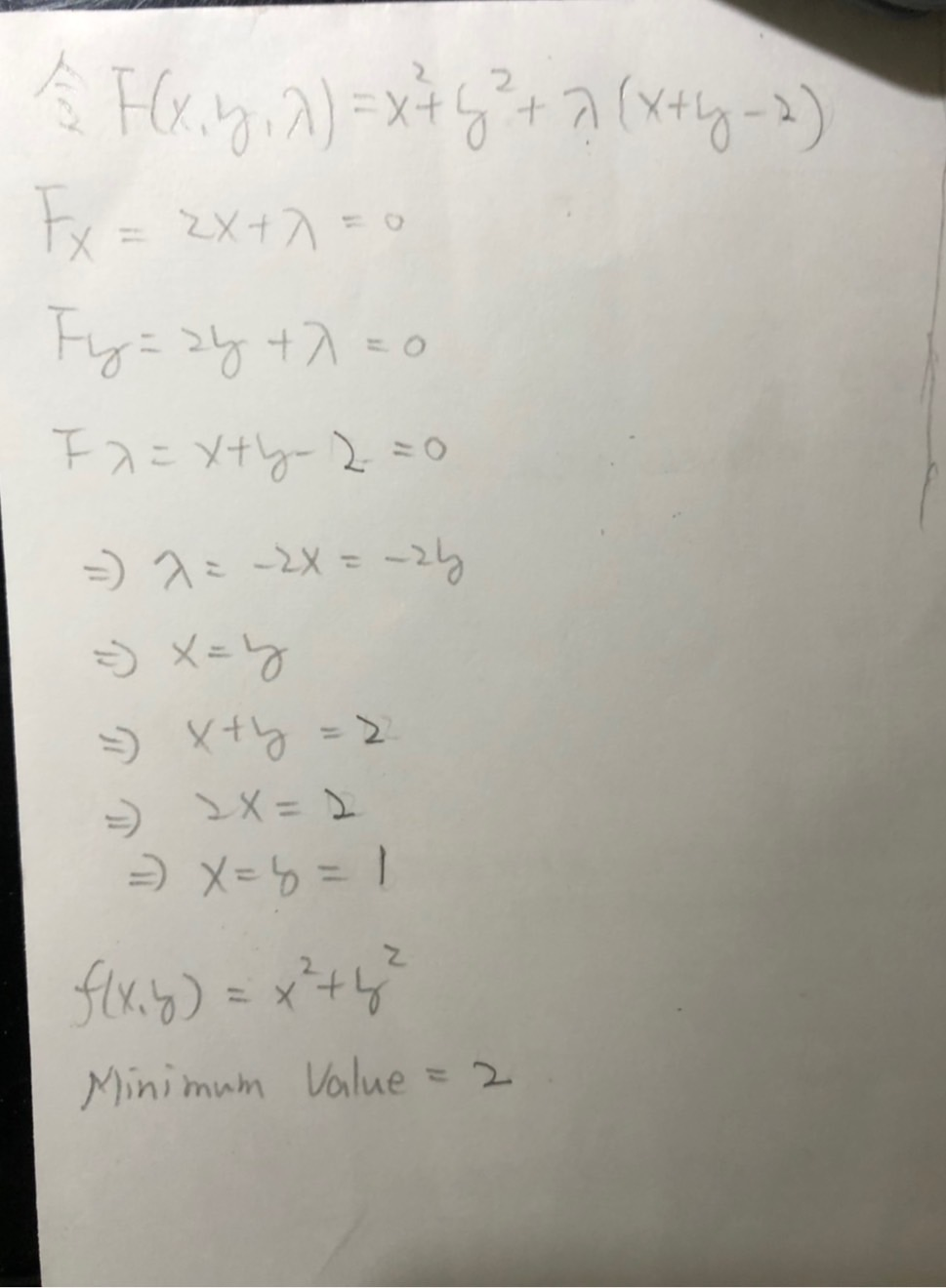
 and . The samples in the second class are from  where y1 and y2 are jointly Gaussian r.v. with  and .

Find the 𝒘𝑜𝑝𝑡 (with normalization) and optimal decision value 𝛼𝑜𝑝𝑡 based on the concept of the linear discriminant analysis (LDA).



1. Use the Lagrange multiplier method to find the solution of the following:

Minimize 𝑓(𝑥, 𝑦) = 𝑥2 + 𝑦2 subject to 𝑥 + 𝑦 = 2.



1. Use the PCA approach to reduce the feature dimension of the Iris data set from 4 to 3. As usual, take 70% of the samples as training set and perform PCA. Use 5-NN to classify the test set and then report the average accuracy after 10 trials.

import numpy as np

from sklearn import decomposition

from sklearn import datasets

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

total=0

for i in range(10):

iris = datasets.load\_iris()

data\_x = iris.data

data\_y= iris.target

pca = decomposition.PCA(n\_components=3)

pca.fit(data\_x)

data\_x= pca.transform(data\_x)

data\_y= np.choose(data\_y, [1, 2, 0]).astype(float)

x\_train,x\_test,y\_train,y\_test=train\_test\_split(data\_x,data\_y,test\_size=0.3)

knn=KNeighborsClassifier(n\_neighbors=5)

knn.fit(x\_train,y\_train)

acc=knn.score(x\_test,y\_test)

print('Acc=%.2f'%acc)

total+=acc

avg=total/10

print('avg acc=%.2f'%avg)

1. We mention the gradient descent algorithm in the lecture. You are asked to write a gradient descent program to find the minimum of the following cost function

𝐽(𝑥, 𝑦) = 𝑥 + 𝑦 − 100(𝑥2 + 𝑦2 − 1)2.

To have a unique answer, use 𝑥0 = 𝑦0 = 1 and 𝜂 = 0.005. (a) Iterate the program 1,000 times and print out the final values of (*x*, *y*). (b) Is your answer close to the result obtained from the Lagrange multiplier (the example in the lecture notes)? (c) If the analytic form of the 𝐽(x, y) is unknown and only input/output pairs are available (like a black box), how do you estimate the derivatives to complete the computation?

1. Overflow (-∞, -∞)
2. Not even close
3. To observe every iterator if it goes to the right direction

from numpy import \*

from numpy.linalg import norm

def f(x, y):

return x+y-100.0\*((x\*\*2+y\*\*2-1)\*\*2)

def dfdx(x, y):

return 1-400.0\*(x)\*(x\*\*2+y\*\*2-1)

def dfdy(x, y):

return 1-400.0\*(y)\*(x\*\*2+y\*\*2-1)

def gradf(x, y):

return array([dfdx(x, y), dfdy(x, y)])

def grad\_descent2(f, gradf, init\_t, alpha):

EPS = 1e-5

prev\_t = init\_t-10\*EPS

t = init\_t.copy()

max\_iter = 1000

iter = 0

while norm(t - prev\_t) > EPS and iter < max\_iter:

prev\_t = t.copy()

t -= alpha\*gradf(t[0], t[1])

print( t, f(t[0], t[1]), gradf(t[0], t[1]))

iter += 1

return t

grad\_descent2(f, gradf, array([1.0, 1.0]), 0.005)