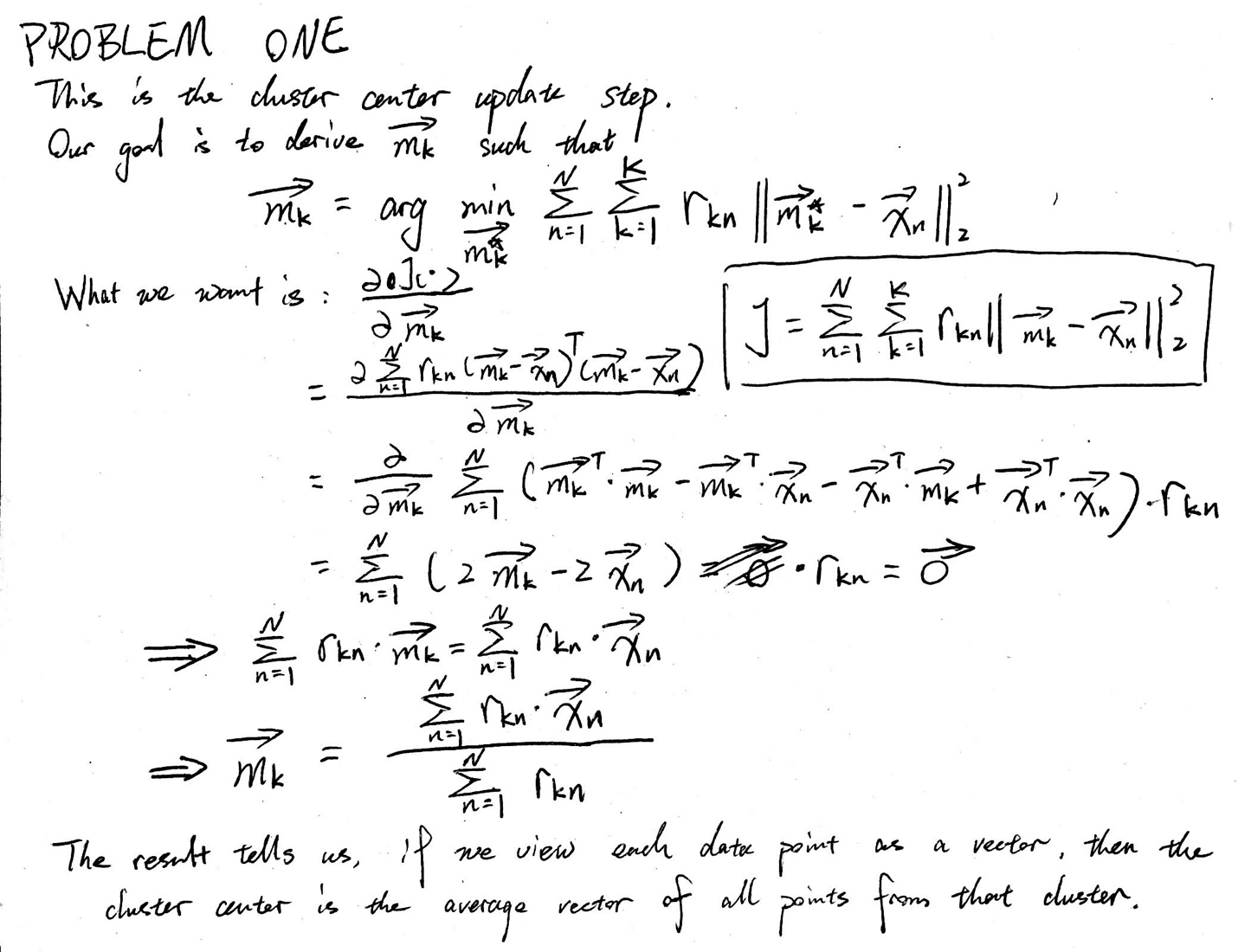
**COEN 240 MACHINE LEARNING**

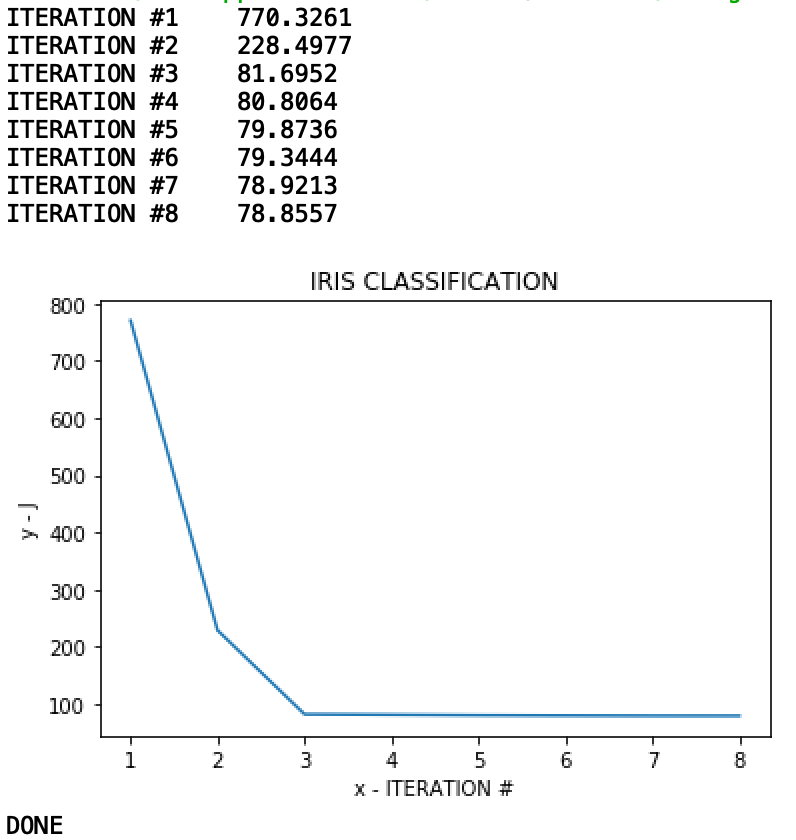
**HOMEWROK TWO**

**NAME: BOSEN YANG STUDENT ID: 1589880**

PROBLEM ONE



PROBLEM TWO:

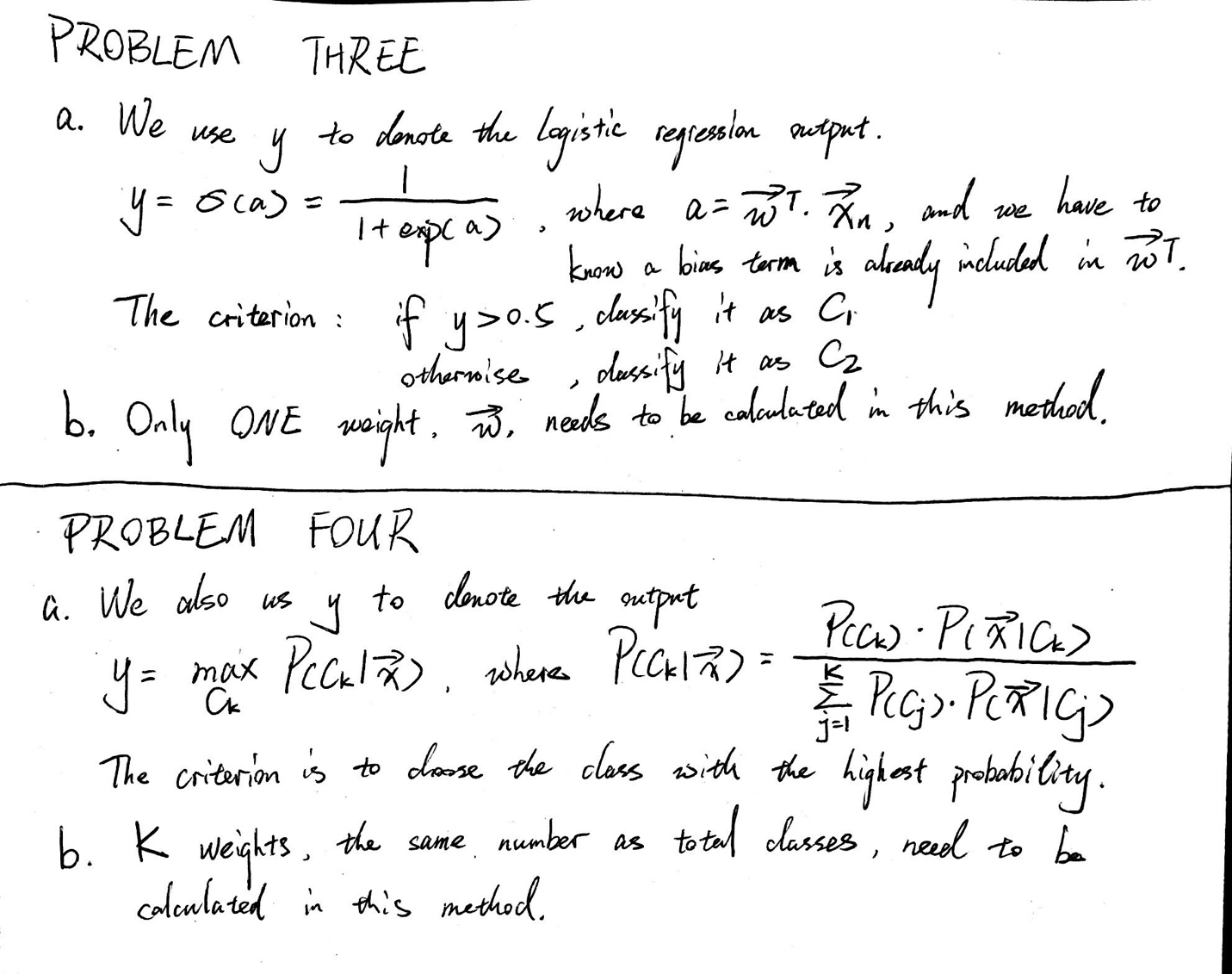


This is a typical result of our algorithm. After several repeats, it is found that the best our algorithm can do is to minimize the error function to some figure around 78.85. We can see our model converges really fast, normally taking 6 to 14 iterations, which means the model can quickly approximate the real cluster centers We can also find that for a majority of iteration, the algorithms is just adjusting the cluster center around its final position for better accuracy.

There is one more thing we should be aware of. The random initialization of cluster center can possibly lead to different convergence results. Sometimes we might not get result as good as the graph show above. For example, the following graphs are examples of convergence to a not so good local optimum.



PROBLEM THREE & FOUR



**ATTACHMENTS**

**PROBLEM TWO CODE**

**﻿**

﻿import numpy as np

import pandas as pd

import random

import sys

import matplotlib.pyplot as plt

def initializeCenter(X, centers):

attr\_max = np.zeros((NUM\_COLUMN, 1))

attr\_min = np.zeros((NUM\_COLUMN, 1))

for i in range(NUM\_COLUMN):

attr\_val = X[:, i]

attr\_max[i] = np.amax(attr\_val)

attr\_min[i] = np.amin(attr\_val)

for center in centers:

for i in range(X.shape[1]):

center[i][0] = random.uniform(attr\_min[i], attr\_max[i])

# X is a list of lists, where each component list represents a cluster

def assignCenter(X, centers):

new\_X = []

for i in range(NUM\_CENTER):

new\_X.append([])

M = 0

for cluster in X:

for point in cluster:

point = point.reshape(NUM\_COLUMN,1)

index, min\_distance = findCorresCenter(point, centers)

new\_X[index].append(point)

M = M + min\_distance

return new\_X, M

def findCorresCenter(point, centers):

index = 0

min\_distance = np.linalg.norm(point-centers[0])\*\*2

for i in range(len(centers)):

temp = np.linalg.norm(point-centers[i])\*\*2

if (temp < min\_distance):

min\_distance = temp

index = i

return index, min\_distance

def calculateCenter(X, centers):

new\_centers = []

for i in range(NUM\_CENTER):

cluster = X[i]

cluster\_center = np.zeros((NUM\_COLUMN, 1))

if (len(cluster) == 0):

new\_centers.append(centers[i])

continue

for i in range(NUM\_COLUMN):

cluster = np.array(cluster)

cluster = cluster.reshape(cluster.shape[0], NUM\_COLUMN)

attr\_val = cluster[:, i]

cluster\_center[i][0] = np.mean(attr\_val)

new\_centers.append(cluster\_center)

return new\_centers

# READ FROM ORIGINAL XLS FILE INTO NUMPY ARRAY

file\_path\_xls = "/Users/bosen/Library/Mobile Documents/com~apple~CloudDocs/Portal/COEN 240/Assignment/HW02/Iris.xls"

file\_path\_csv = "/Users/bosen/Library/Mobile Documents/com~apple~CloudDocs/Portal/COEN 240/Assignment/HW02/Iris.csv"

iris\_xls = pd.read\_excel(file\_path\_xls)

iris\_xls.to\_csv(file\_path\_csv, index = None, header=False)

iris\_raw = np.genfromtxt(file\_path\_csv, delimiter=',')[:, 1:]

del(iris\_xls)

NUM\_ROW = iris\_raw.shape[0]

NUM\_COLUMN = iris\_raw.shape[1]-1

X = iris\_raw[:, :NUM\_COLUMN]

t = iris\_raw[:, NUM\_COLUMN].reshape(NUM\_ROW,1)

# DEFINE HYPERPARAMETERS AND INITIALIZE CENTERS

M = sys.float\_info.max

Ms = []

EPSILON = 10\*\*(-5)

NUM\_CENTER = 3

NUM\_ITERATION = 0

centers = []

for i in range(NUM\_CENTER):

center = np.zeros((NUM\_COLUMN, 1))

centers.append(center)

initializeCenter(X, centers)

X = np.array([X])

# ALTERNATES BETWEEN ASSIGNMENT AND CLUSTER-CENTER UPDATE

while(True):

NUM\_ITERATION = NUM\_ITERATION+1

X, new\_M = assignCenter(X, centers) # THE FORM OF X

if (M - new\_M < EPSILON):

break

M = new\_M

Ms.append(M)

print("ITERATION #%d\t%.4f" % (NUM\_ITERATION, M))

centers = calculateCenter(X, centers)

# PLOTTING

plt.plot(range(1, len(Ms)+1, 1), Ms)

plt.xlabel('x - ITERATION #')

plt.ylabel('y - J')

plt.title('IRIS CLASSIFICATION')

plt.show()

print("DONE")