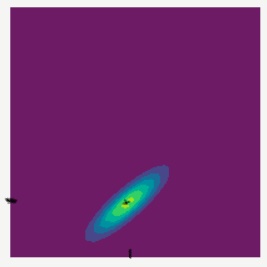
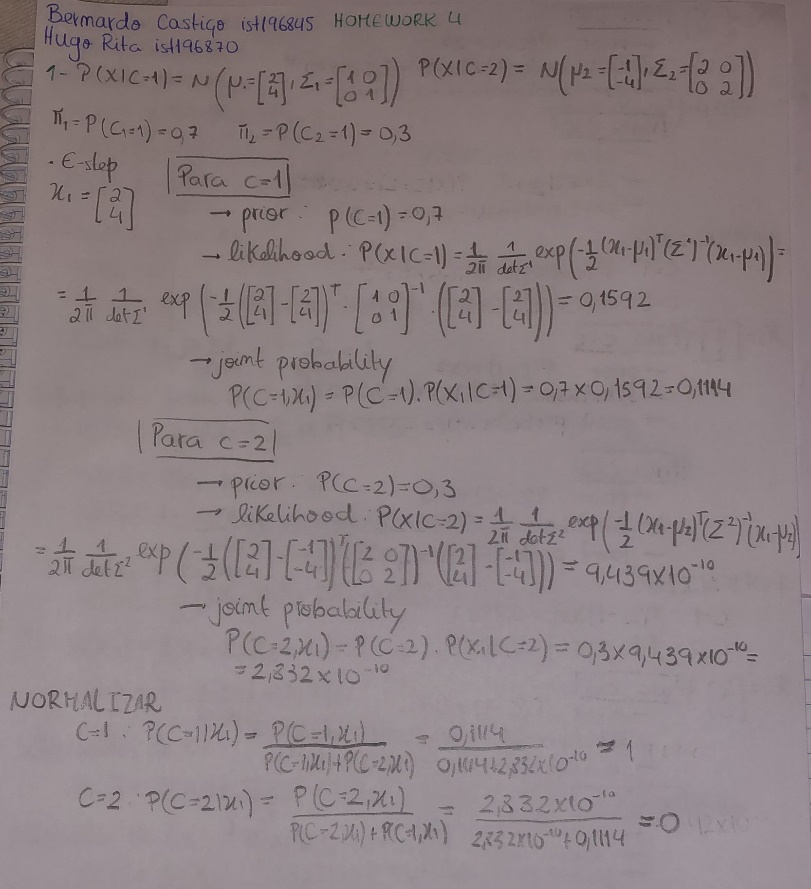
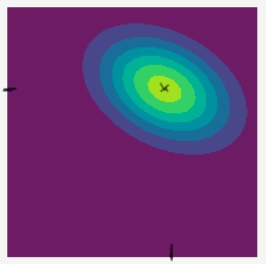
**I. Pen-and-paper**

**Answer 1**

Cluster 1

****

-3.4176

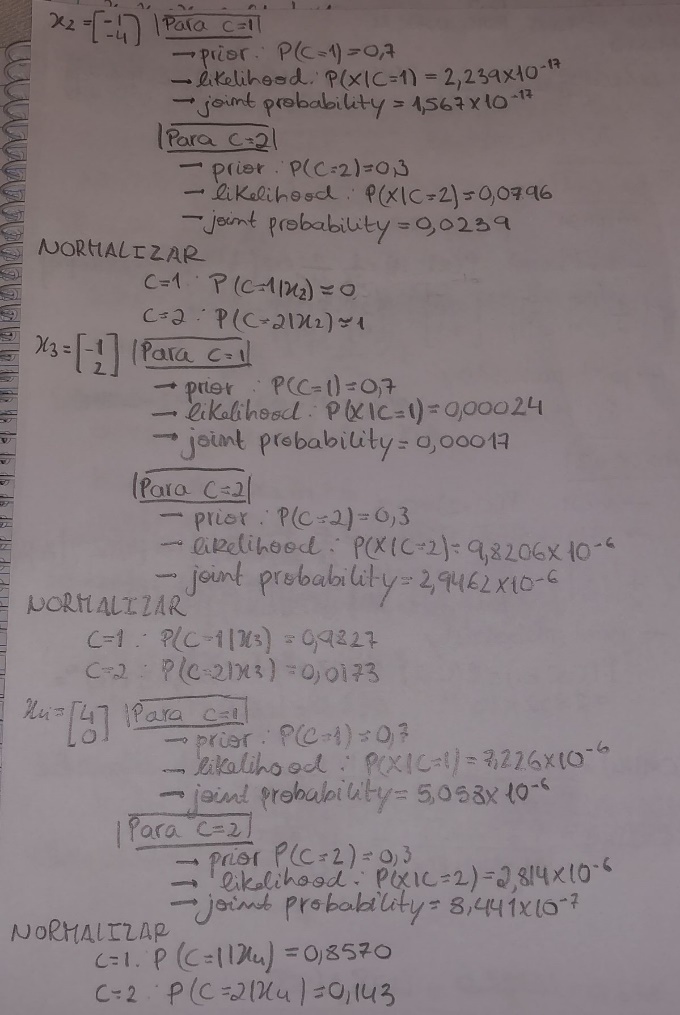
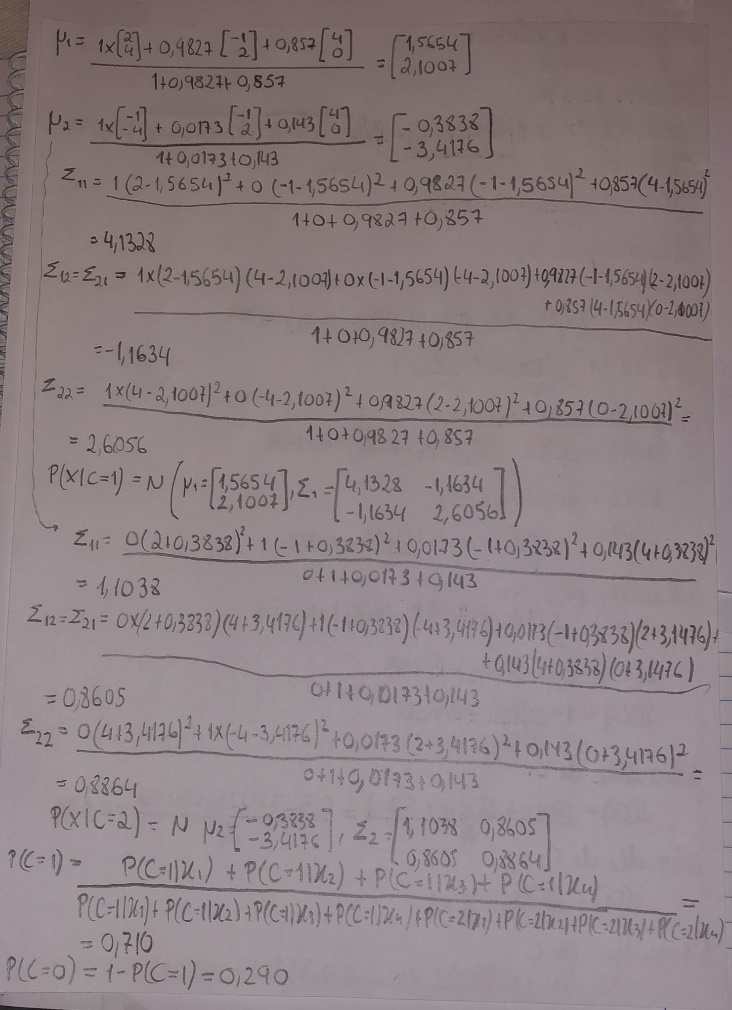
-0.3837

Cluster 2

1,5654

2,1007

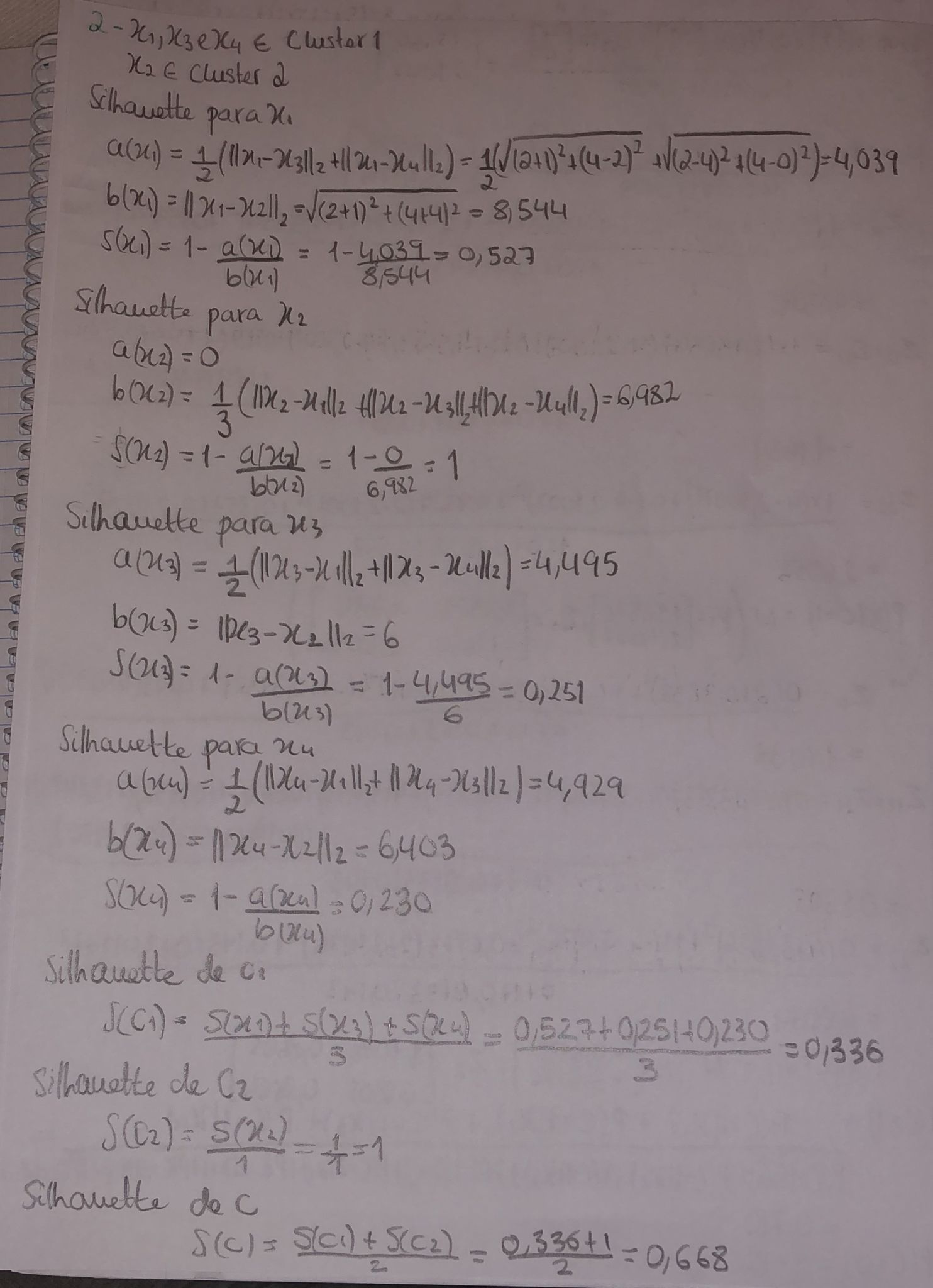
1

** **

3

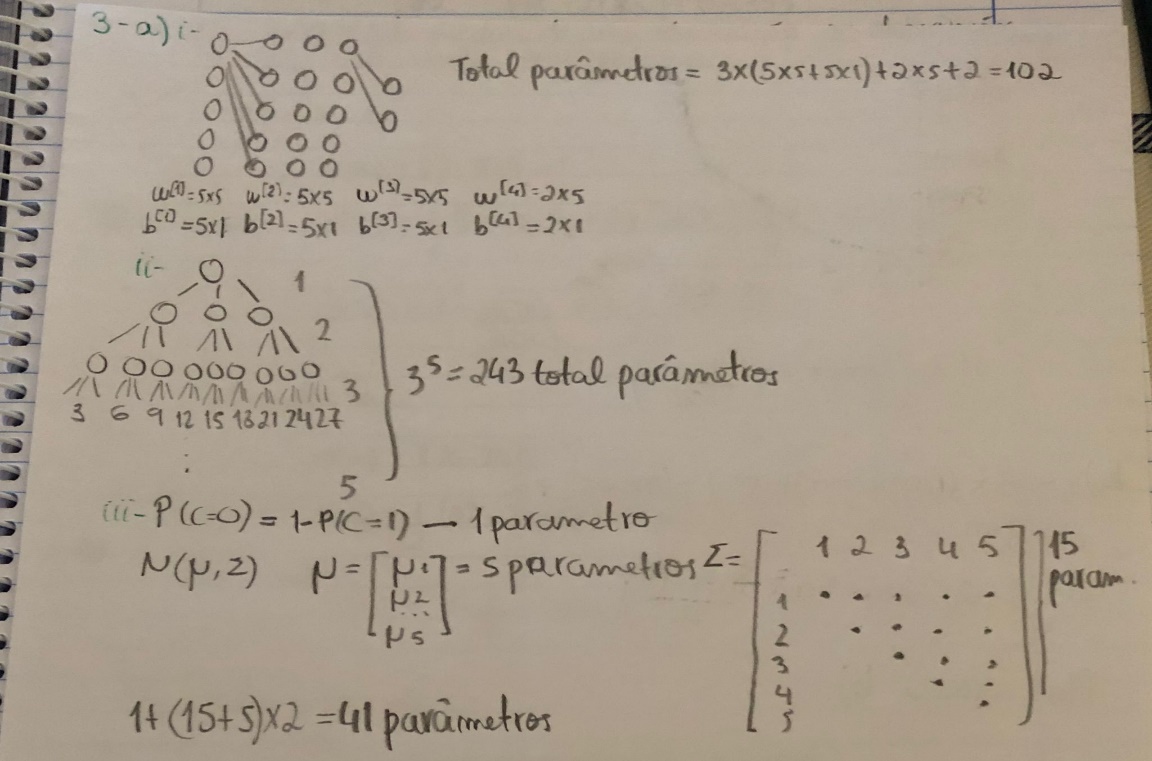
2

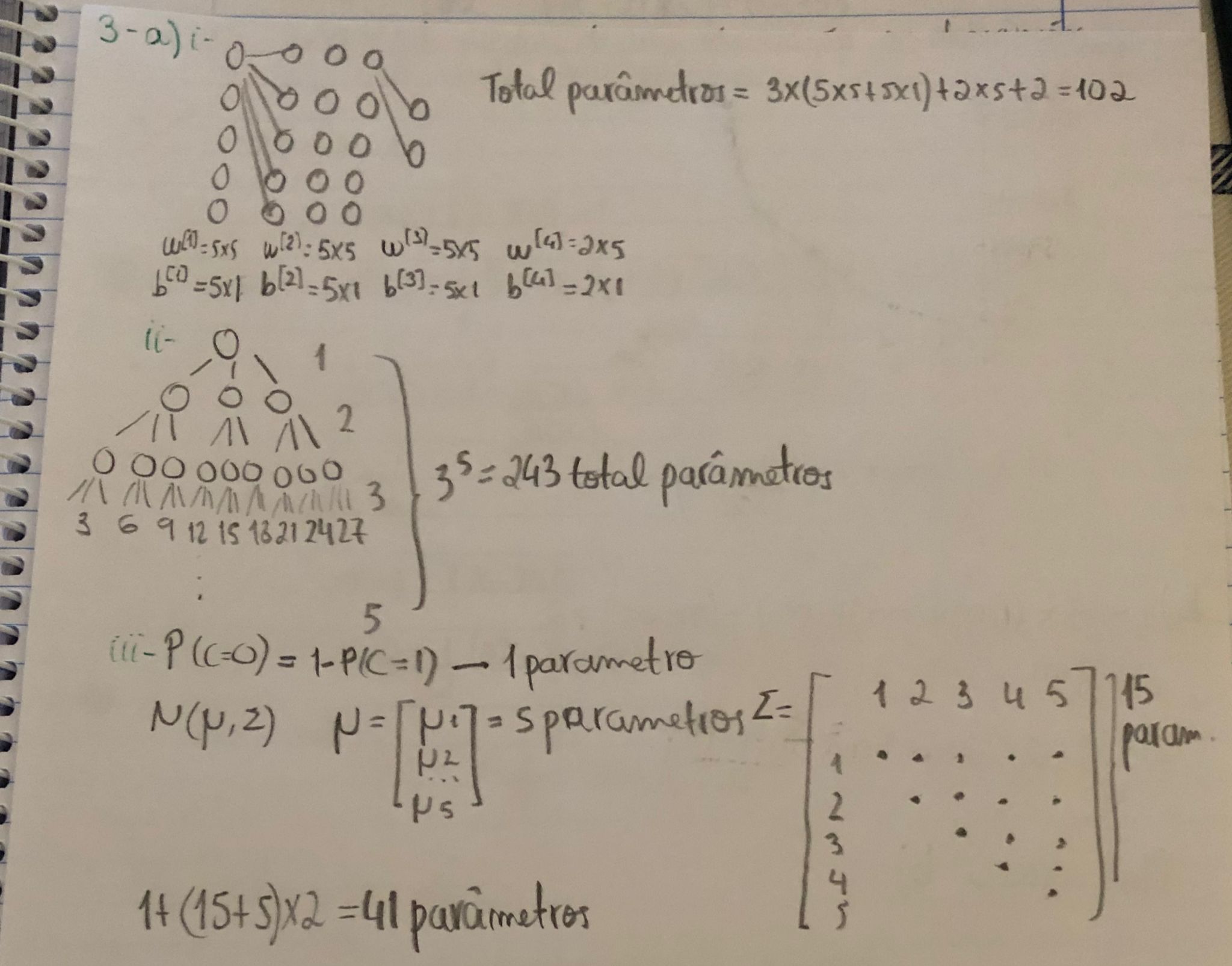
**Answer 2**

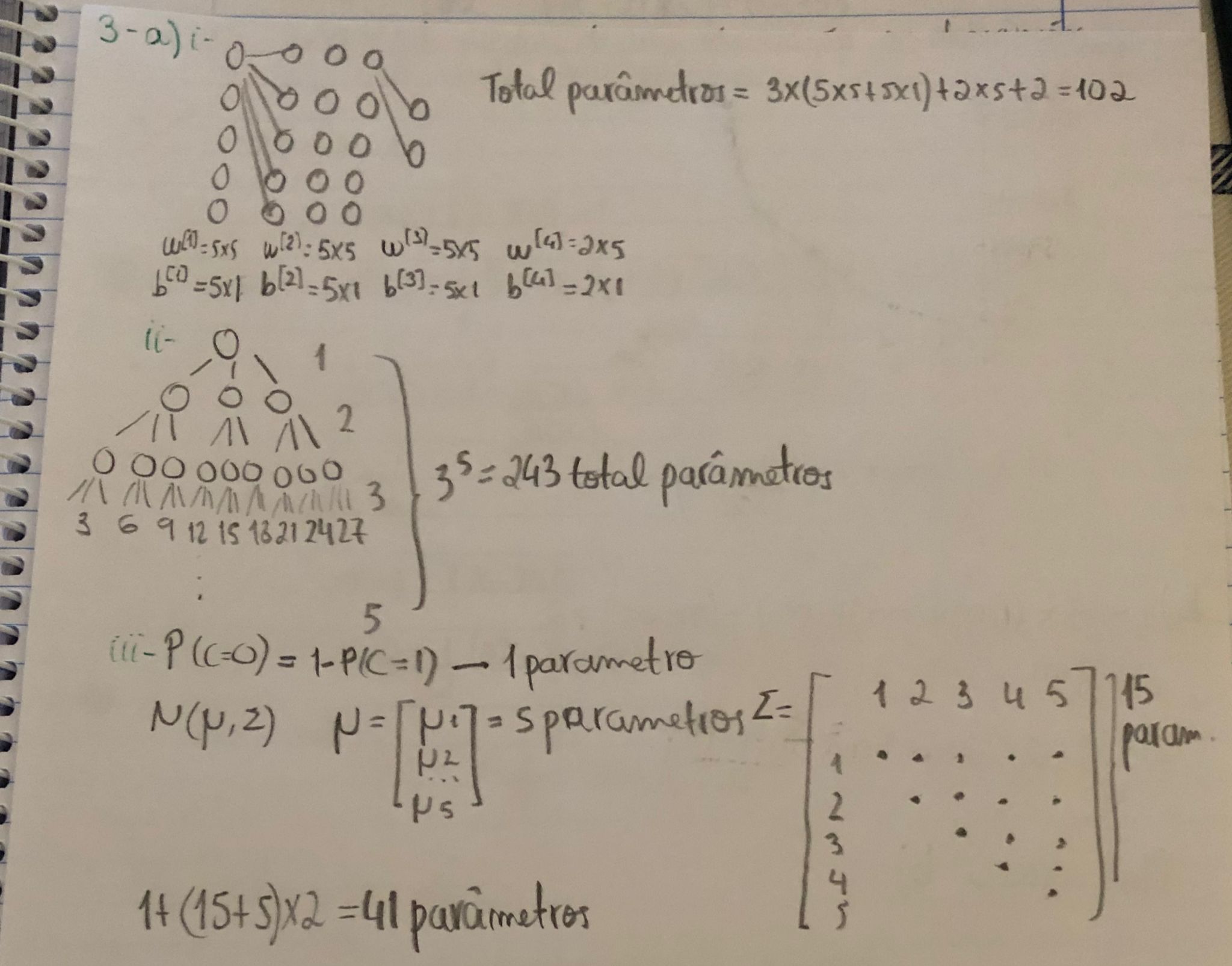
****

**Answer 3-a**

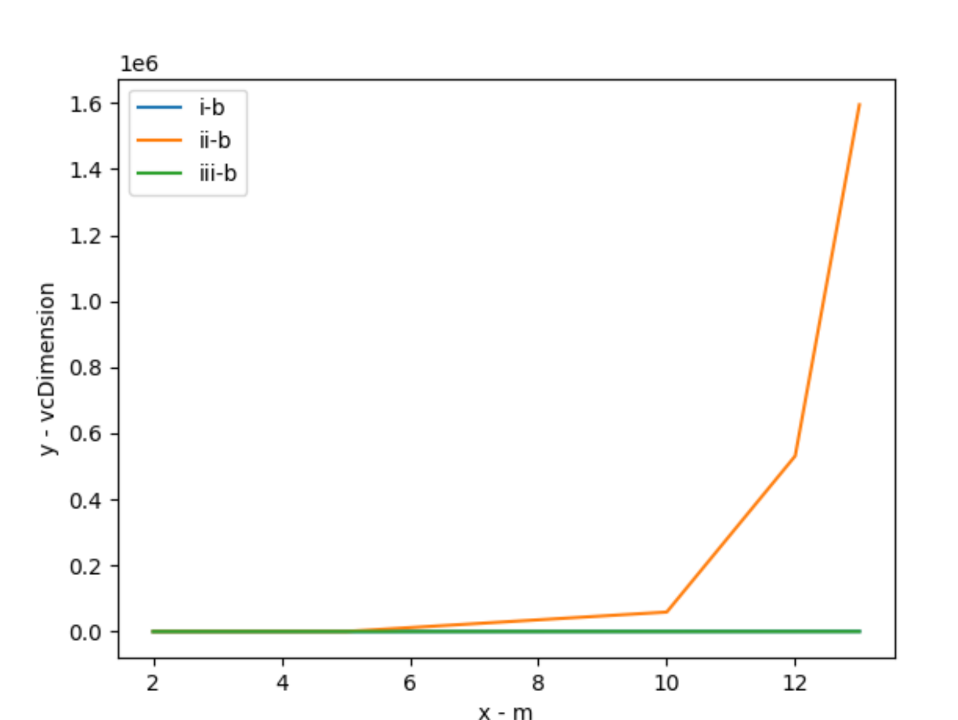
**i-**

****

**ii- **

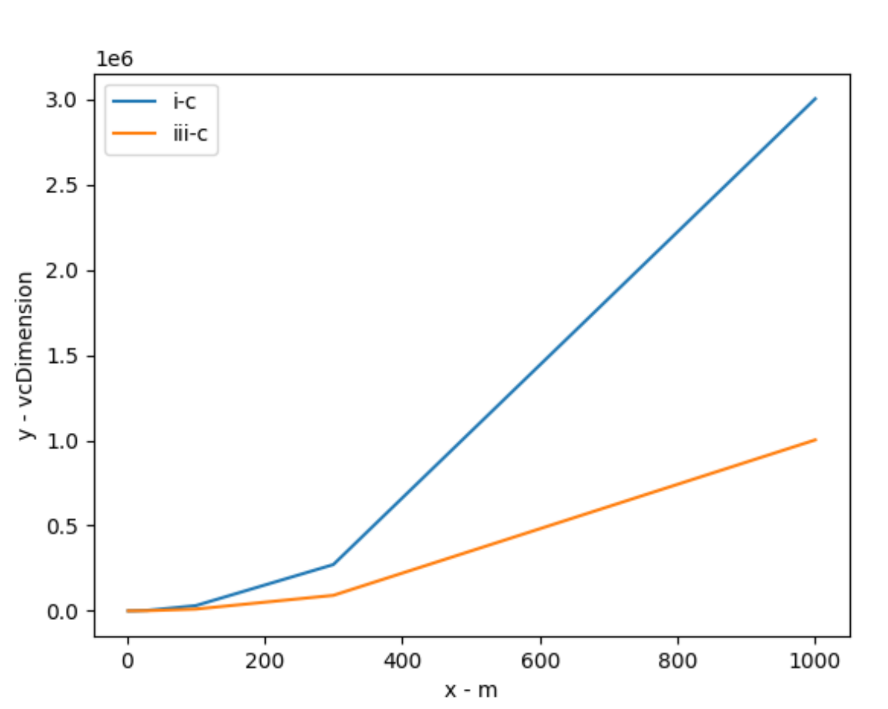
**iii-**

**Answer 3-b**

****

Pela observação do gráfico, verifica-se que a variação da vcDimension da decision tree(ii), aumenta exponencialmente, e verifica-se um aumento abrupto a partir de data dimensionality = 10 face à MLP com 3 hidden layers e ao Bayesian Classifier com uma multivariate Gaussian likelihood.

**Answer 3-c**

****

Pela observação do gráfico, verifica-se que um maior aumento da vcDimension no MLP Classifier com 3 hidden layes do que no Bayesian Classifier com uma multivariate Gaussian likelihood a partir da data dimensionality = 100.

**II. Programming and critical analysis**

**Answer 4**

**Uma imagem com texto

Descrição gerada automaticamente**

**a)**

Pelos resultados acima apresentados verifica-se que no algoritmo kMeans, k = 3 apresenta uma melhor ECR value que k = 2 para a nossa data.

Sendo o ECR a media dos pontos mal classificados concluimos que ao adicionarmos um novo cluster vai existir uma maior margem para classificação de pontos, e portanto menos pontos mal classificados, assim é natural que o ECR seja mais pequeno para k = 3.

**b)**

Pelos resultados acima apresentados verifica-se que no algoritmo kMeans, k = 2 apresenta uma melhor silhouette que k = 3 para a nossa data. Isto deve-se ao facto de para k = 2 os clusters serem mais compactos e estarem mais separados entre si.

**Answer 5**

**Answer 6**

**Uma imagem com texto

Descrição gerada automaticamente**

No exercício 5 verifica-se que a silhouette do algoritmo kMeans com k = 2 e apenas selecionando as 2 melhores features da nossa data segundo a mutual information é bastante boa, isto significa que os cluster são compactos e estão afastados entre si, algo que se pode verificar pelo gráfico apresentado na resposta 5.

Quanto ao ECR value verifica-se que este é melhor do que para k = 2 usando toda a data, mas pior do que para k = 3 usando toda a data.

Sendo o ECR a media dos pontos mal classificados concluimos que ao adicionarmos um novo cluster vai existir uma maior margem para classificação de pontos, e portanto menos pontos mal classificados, assim é natural que o ECR seja mais pequeno do que para k = 2. Dado que no exercicio 5 apenas se selecionam as duas mlehores features, a nossa data torna-se mais imprecisa o que leva um maior grupo de pontos mal classificado, portanto é normal que o ECR value do exercício 5 seja maior que o de k = 3.

**III. APPENDIX**

Paste your programming code here using Consolas 9pt or 10pt.

Use **highlighting** or colored text to facilitate the analysis by your faculty hosts.

# Grupo 117 Aprendizagem HomeWork 4

# Bernardo Castico ist196845

# Hugo Rita ist196870

import pandas as pd

import sklearn

from scipy.io import arff

from sklearn.cluster import KMeans

import numpy as np

from sklearn.metrics import silhouette\_score

from sklearn.feature\_selection import SelectKBest

from sklearn.feature\_selection import mutual\_info\_classif

import matplotlib.pyplot as plt

def getDataToMatrix(lines):

    realLines = []

    data = []

    toDelete = []

    for i in range(len(lines)):

        if i > 11:

            realLines += [lines[i]]

    for i in range(len(realLines)):

        for j in range(len(realLines[i])):

            if realLines[i][j] == "benign\n":

                realLines[i][j] = "benign"

            elif realLines[i][j] == "malignant\n":

                realLines[i][j] = "malignant"

            elif realLines[i][j] == '?':

                toDelete += [i]

            else:

                realLines[i][j] = int(realLines[i][j])

    for i in range(len(realLines)):

        if i not in toDelete:

            data += [realLines[i]]

    return data

def splitData(list):

    a = []

    b = []

    for i in list:

        a.append(i[:-1])

        b.append(i[-1])

    return [a,b]

def main():

    data, res2 = [],[]

    cluster02, cluster12, cluster03, cluster13, cluster23, cluster05, cluster15, cluster25 = 0,0,0,0,0,0,0,0

    Benign02, malignant02, Benign12, malignant12, Benign03, malignant03, Benign13, malignant13, Benign23, malignant23 = 0,0,0,0,0,0,0,0,0,0

    Benign05, malignant05, Benign15, malignant15, Benign25, malignant25 = 0,0,0,0,0,0

    xCluster0, yCluster0, xCluster1, yCluster1, xCluster2, yCluster2 = [],[],[],[],[],[]

    with open("HW3-breast.txt") as f:

        lines = f.readlines()

    for line in lines:

        tmp = line.split(',')

        res2.append(tmp)

    data = getDataToMatrix(res2)

    trainDataSplit = splitData(data)

    kMeans2 = KMeans(n\_clusters=2, random\_state=0).fit(trainDataSplit[0])

    kMeans3 = KMeans(n\_clusters=3, random\_state=0).fit(trainDataSplit[0])

    kLabels2 = kMeans2.labels\_

    kLabels3 = kMeans3.labels\_

    for i in range(len(kLabels2)):

        if kLabels2[i] == 0:

            cluster02 += 1

            if trainDataSplit[1][i] == 'malignant':

                malignant02 += 1

            else:

                Benign02 += 1

        elif kLabels2[i] == 1:

            cluster12 += 1

            if trainDataSplit[1][i] == 'malignant':

                malignant12 += 1

            else:

                Benign12 += 1

        if kLabels3[i] == 0:

            cluster03 += 1

            if trainDataSplit[1][i] == 'malignant':

                malignant03 += 1

            else:

                Benign03 += 1

        elif kLabels3[i] == 1:

            cluster13 += 1

            if trainDataSplit[1][i] == 'malignant':

                malignant13 += 1

            else:

                Benign13 += 1

        elif kLabels3[i] == 2:

            cluster23 += 1

            if trainDataSplit[1][i] == 'malignant':

                malignant23 += 1

            else:

                Benign23 += 1

    ECR2 = 0.5\*((cluster02-max(Benign02,malignant02)) + (cluster12-max(Benign12, malignant12)))

    ECR3 = (1/3)\*((cluster03-max(Benign03,malignant03)) + (cluster13-max(Benign13, malignant13))+ (cluster23-max(Benign23, malignant23)))

    print("ECR K = 2")

    print(ECR2)

    print("ECR k = 3")

    print(ECR3)

    print("Silhouette K = 2")

    print(silhouette\_score(trainDataSplit[0], kLabels2))

    print("Silhouette K = 3")

    print(silhouette\_score(trainDataSplit[0], kLabels3))

    #EX5

    decision = SelectKBest(mutual\_info\_classif, k=2).fit(trainDataSplit[0], trainDataSplit[1])

    decisionTrainData = decision.transform(trainDataSplit[0])

    kMeans3Ex5 = KMeans(n\_clusters=3, random\_state=0).fit(decisionTrainData)

    kLabelsEx5 = kMeans3Ex5.labels\_

    for i in range(len(kLabelsEx5)):

        if kLabelsEx5[i] == 0:

            cluster05 += 1

            if trainDataSplit[1][i] == 'malignant':

                malignant05 += 1

            else:

                Benign05 += 1

        elif kLabelsEx5[i] == 1:

            cluster15 += 1

            if trainDataSplit[1][i] == 'malignant':

                malignant15 += 1

            else:

                Benign15 += 1

        elif kLabelsEx5[i] == 2:

            cluster25 += 1

            if trainDataSplit[1][i] == 'malignant':

                malignant25 += 1

            else:

                Benign25 += 1

    ECR5 = (1/3)\*((cluster05-max(Benign05,malignant05)) + (cluster15-max(Benign15, malignant15))+ (cluster25-max(Benign25, malignant25)))

    print("ECR Ex5")

    print(ECR5)

    print("Silhouette Ex5")

    print(silhouette\_score(decisionTrainData, kLabelsEx5))

    for i in range(len(kLabelsEx5)):

        if kLabelsEx5[i] == 0:

            xCluster0 += [decisionTrainData[i][0]]

            yCluster0 += [decisionTrainData[i][1]]

        elif kLabelsEx5[i] == 1:

            xCluster1 += [decisionTrainData[i][0]]

            yCluster1 += [decisionTrainData[i][1]]

        else:

            xCluster2 += [decisionTrainData[i][0]]

            yCluster2 += [decisionTrainData[i][1]]

    plt.scatter(xCluster0, yCluster0, label="Cluster 0")

    plt.scatter(xCluster1, yCluster1, label="Cluster 1")

    plt.scatter(xCluster2, yCluster2, label="Cluster 2")

    plt.xlabel('x - BestFeature1')

    plt.ylabel('y - BestFeature2')

    plt.title('Cluster solution with k=3 and 2 K best features')

    # show a legend on the plot

    plt.legend()

    plt.show()

main()

**END**