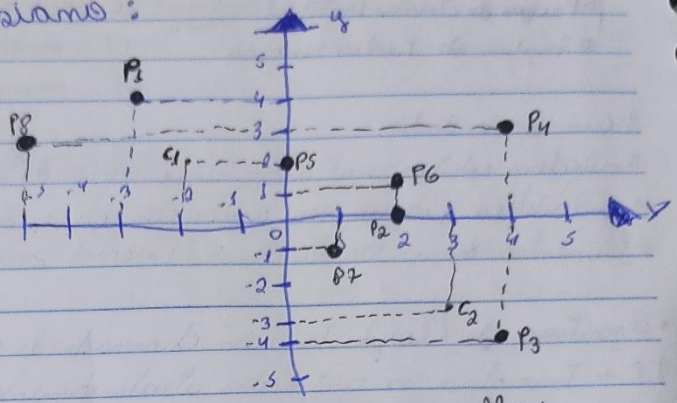
Homework sobre agrupamento usando o algoritmo K-means

1) Usando a distância Euclidiana, realize o agrupamento dos 8 vetores bidimensionais abaixo a partir do algoritmo K-means. Com o mesmo, encontre K = 2 centróides (médias) para representar estes dados e mostre estas centróides em um gráfico Cartesiano:

X = [-3, 4
2, 0
4, -4
4, 3
0, 2
2, 1
1, -1
-5, 3]

1. Usando a distância Euclidiana, realize o agrupamento dos 8 vetores bidimensionais abaixo a partir do algoritmo K-means. Com o mesmo, encontre K=2 centróides para representar estes dados e mostre estes centroides em um gráfico Cartesiano:

$X = [-3, 4 \Rightarrow P_1$
$2, 0 \Rightarrow P_2$
$4, -4 \Rightarrow P_3$
$4, 3 \Rightarrow P_4$
$0, 2 \Rightarrow P_5$
$2, 1 \Rightarrow P_6$
$1, -1 \Rightarrow P_7$
$-5, 3] \Rightarrow P_8$



Como nas literaturas os centroides iniciais são escolhidos aleatóriamente, Vou escolher dois pontos aleatórios para representar os centroides. $C_1(-2,2)$; $C_2(+3,-3)$

Agora faremos um agrupamento inicial para determinar quais pontos estão mais próximos dos centroides.

Para C_1

$C_1 \Rightarrow P_1 = 2.23 \Rightarrow C_1$
$C_1 \Rightarrow P_2 = 4.47$
$C_1 \Rightarrow P_3 = 8.48$
$C_1 \Rightarrow P_4 = 6.08$
$C_1 \Rightarrow P_5 = 2.00 \Rightarrow C_1$
$C_1 \Rightarrow P_6 = 4.12 \Rightarrow C_1$
$C_1 \Rightarrow P_7 = 4.24$
$C_1 \Rightarrow P_8 = 3.16 \Rightarrow C_1$

Para C_2

$C_2 \Rightarrow P_1 = 8,21$
$C_2 \Rightarrow P_2 = 3.16 \Rightarrow C_2$
$C_2 \Rightarrow P_3 = 9.49 \Rightarrow C_2$
$C_2 \Rightarrow P_4 = 6.08 \Rightarrow C_2$
$C_2 \Rightarrow P_5 = 5.83$
$C_2 \Rightarrow P_6 = 4.12$
$C_2 \Rightarrow P_7 = 2.82 \Rightarrow C_2$
$C_2 \Rightarrow P_8 = 10$

Agrupados os dados, vamos calcular os novos centroides a partir da média dos pontos dentro de cada cluster.

Para C_1	Para C_2
(-3, 4) P1	(2,0) P2
(0,2) P5	(4,-4) P3
(2, 1) P6	(4, 3) P4
(-5, 3) P8	(1,-1) P7
média x = (-3+0+2-5)/4	media x = (2+4+4+1)/4
média y = (4+2+1+3)/4	media y = (0+(-4)+3-1)/4
Novo centroide C_1 = (-1.5, 2.5)	Novo C_2 = (2.75, -0.5)

#Agora temos que recalcular as distâncias com base nesses novos centroides. Para verificar se terá mudança na proximidade dos pontos em relação a esses novos centroides.

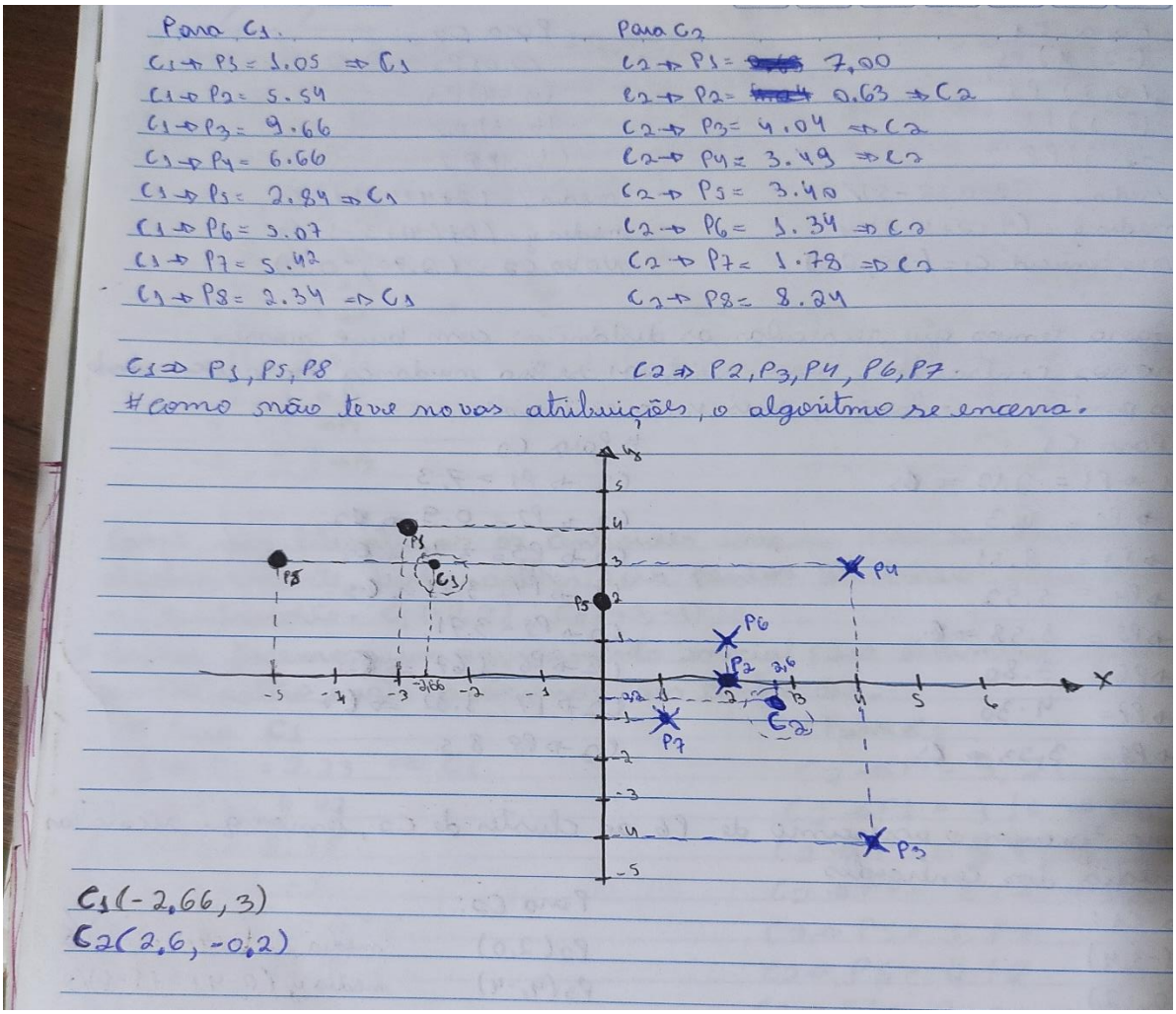
# Para C_1	# Para C_2
$C_1 \rightarrow P1 = 2.12 \Rightarrow C_1$	$C_2 \rightarrow P1 = 7.3$
$C_1 \rightarrow P2 = 4.3$	$C_2 \rightarrow P2 = 0.9 \Rightarrow C_2$
$C_1 \rightarrow P3 = 8.51$	$C_2 \rightarrow P3 = 3.71 \Rightarrow C_2$
$C_1 \rightarrow P4 = 5.52$	$C_2 \rightarrow P4 = 3.71 \Rightarrow C_2$
$C_1 \rightarrow P5 = 1.58 \Rightarrow C_1$	$C_2 \rightarrow P5 = 3.71$
$C_1 \rightarrow P6 = 3.80$	$C_2 \rightarrow P6 = 1.67 \Rightarrow C_2$
$C_1 \rightarrow P7 = 4.30$	$C_2 \rightarrow P7 = 1.82 \Rightarrow C_2$
$C_1 \rightarrow P8 = 3.53 \Rightarrow C_1$	$C_2 \rightarrow P8 = 8.5$

como tivemos o acrescimo de P6 ao cluster de C_2, temos que recalcular a média dos centroides

Para C_1:	Para C_2:	
P3(-3,4)	P2(2,0)	média x = (5+4+4+2+1)/5
P5(0, 2)	P3(4,-4)	media y (0-4+3+1-1)/5
P8(-5,3)	P4(4,3)	C_2 = (2.6, -0.2)
média x = (-3+0-5)/3	P6(2,1)	
média y = (4+2+3)/3	P7(1,-1)	
Novo centroide C_1 = (-2.66, 3)		

Agora temos que recalcular as distâncias com base nesses novos centroides para verificar se é adicionado algum novo ponto a algum cluster.

SÃO DOMINGOS



2) Um problema sério do uso do KNN na prática é quando o conjunto de treino é muito grande, e fazer a busca pelo vizinho mais próximo, ou mesmo armazenar o conjunto de treino em memória RAM pode ser proibitivo. Uma solução para isso é usar, antes do KNN, um algoritmo de agrupamento como o K-means (https://en.wikipedia.org/wiki/K-means_clustering) para diminuir o conjunto de dados que o KNN irá manipular. Note que o KNN é um algoritmo de aprendizado supervisionado. Já algoritmos de agrupamento como o K-means são tipicamente "não-supervisionados" (não exigem rótulo para cada exemplo). Mas neste problema usaremos o K-means de forma independente para cada classe (rótulo) do conjunto de treino: executaremos um K-means apenas para os exemplos com rótulo "0" e outro K-means para os exemplos com rótulo "1". Usando esses dois agrupamentos, converta seu conjunto de treino em um novo conjunto com apenas dois vetores (chamados "centróides"), um representando a classe "0" e outro a classe "1". Depois o KNN deve ser usado com base neste conjunto de centróides para classificar o seu conjunto de teste.

Observe que o K-means é um algoritmo iterativo. Ele é implementado no Matlab / Octave como kmeans e no Scikit-learn: https://scikit-

learn.org/stable/modules/generated/sklearn.cluster.KMeans.html. Mas nós não precisaremos usar estas implementações pois assumiremos que a distância é Euclideana e queremos apenas 1 centróide. Daí esta centróide pode simplesmente ser obtida como a média aritmética dos vetores com os quais estamos lidando, e não precisamos de iterações extras do K-means.

Por exemplo, assuma que há 2 classes em um dado conjunto de treino com 6 exemplos no total:
1000,0,2,0
0,1,-4,0
2,2,2,0
-6000,-4,-1,1
0,-1,-4,1
-3000,-6,-1,1
O vetor de entrada x possui 3 elementos, e os organizando para as classes 0 e 1, e retirando a média aritmética (o que equivale a executar o K-means neste caso),encontra-se as centróides desejadas. No Octave isso pode ser feito com os comandos (note que não usamos o label "0" e "1" ao compor X0 e X1):

```
X0=[1000,0,2
  0,1,-4
  2,2,2]
X1=[-6000,-4,-1
  0,-1,-4
  -3000,-6,-1]
centroide_0 = mean(X0)
centroide_1 = mean(X1)
```

que gera:

```
centroide_0 = [   334     1     0]
centroide_1 = [  -3000.0000     -3.6667     -2.0000]
```

Daí você comporia o equivalente a um novo conjunto de treino para o KNN com estes dois vetores:
334, 1, 0, 0
-3000.0000, -3.6667, -2.0000, 1
e classificaria seu conjunto de teste usando KNN com K=1 para estes dois vetores.

Observação: como você é curioso, caso queira aprender a usar o Kmeans no Octave (Matlab e Python é parecido), segue um exemplo abaixo (mas não é requerido o executar):

Carregue primeiro o pacote que contém o kmeans

pkg load statistics

e execute os comandos:

```
K=1;
[~,centroide_0] = kmeans (X0, K, 'distance','sqeuclidean')
[~,centroide_1] = kmeans (X1, K, 'distance','sqeuclidean')
```

2°) Dados o conjunto de treino e teste.

set_train

X	Y	label
3000	0	0
3000	-3	0
-1000	2	0
1000	0	0
-6000	-2	1
0	0	1
-3000	-5	1
-2000	-6	1

set_test

X	Y	label
-4000	-2	0
4000	3	0
0	0	1
-3000	-1	1
-5000	-6	1

Normalizando os dados usando o min-max scaling

$$D = \frac{dado - min}{max - min}$$

set_train

X	Y	label
1	0.75	0
1	0.375	0
0.55	1	0
0.77	0.75	0
0	0.5	1
0.66	0.75	1
0.33	0.125	1
0.44	0	1

set_test

	X	Y	label
P_1	0.22	0.5	0
P_2	1.11	1.125	0
P_3	0.66	0.75	1
P_4	0.33	0.625	1
P_5	0.11	0	1

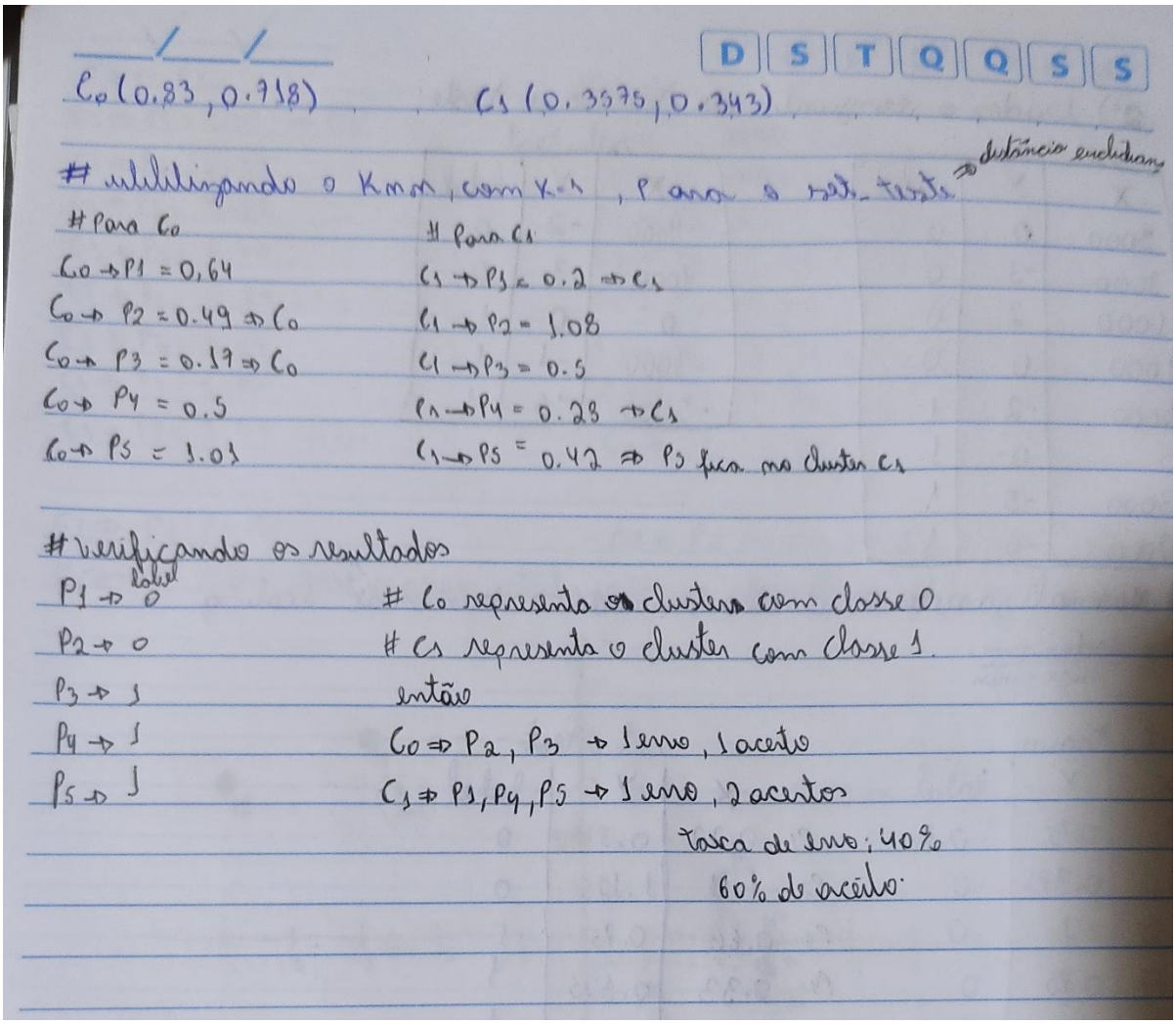
1° Passo dividir o conjunto de treino em label 0 e label 1 e encontrar seus respectivos Centroides.

label 0

X = [1, 0.75
1, 0.375 ⇒ $Centroide_0 = (0.83, 0.718)$
0.55, 1
0.77, 0.75]

label 1

X_1 [0, 0.5
0.66, 0.75 ⇒ $centroide_1$ (0.375, 0.
0.33, 0.125
0.44, 0]



3) Quando o K-means foi inventado e quais foram suas primeiras aplicações? O que são K-medoides?

R=O termo “k-means” foi usado pela primeira vez por James MacQueen em 1967.O algoritmo padrão foi proposto pela primeira vez por Stuart Lloyd do Bell Labs em 1957 como uma técnica para modulação por código de pulso. O k-means clustering é um método de quantização vetorial, originário do processamento de sinais, que visa particionar n observações em k clusters em que cada observação pertence ao cluster com a média mais próxima (cluster centers ou cluster centroid), servindo como um protótipo de o aglomerado.

O que são medoides?

Medoids são objetos representativos de um conjunto de dados ou um cluster dentro de um conjunto de dados cuja soma das distâncias para outros objetos no cluster é mínima. Lembre-se que para K-Means, estávamos trabalhando com centroides. A relação entre centroides e medoides é semelhante à relação entre médias e medianas. Medoides e medianas sempre serão uma das observações nos dados, enquanto isso não é necessariamente o caso de centroides e médias.

O K-Means é sensível a valores discrepantes, mas o K-Medoids atenuou essa sensibilidade por não depender de centroides

4) Como as árvores K-d são usadas no scikit-learn para acelerar o agrupamento K-means?
R=
Ao usar a árvore para acelerar a etapa de atribuição, as informações resumidas coletadas nos permitem podar rapidamente centróides candidatos de grupos de pontos mantidos em nós internos. Quando um grupo de pontos tem apenas um candidato a centróide restante, determinamos que o candidato é o centroide mais próximo para todo o grupo e não precisa mais continuar percorrendo o subárvore. Isso elimina muitos cálculos de distância e comparações em comparação com outras implementações de k-means. Como a árvore é construída apenas uma vez, o algoritmo é melhor otimizado desde que a sobrecarga de tempo de construção da árvore seja menor do que o tempo economizado usando a árvore.