aufgabe1

December 6, 2018

1 Aufgabe 19

1.1 Teilaufgabe a)

Im ersten Schritt ist für jeden Punkt bereits an der Abbildung erkennbar, zu welchem der Clusterzentren sich der geringste Abstand ergibt. Lediglich ein Punkt hat zu zwei Clusterzentren den gleichen Abstand. Es ist im vorliegenden Beispiel egal zu welchem Cluster man ihn zuordnet, das Verfahren konvergiert mit der gleichen Geschwindigkeit. Die Trennung der Cluster ist in der ersten Abbildung skizziert.

1.2 Teilaufgabe b)

Es werden 4 weitere Iterationen durchgeführt und die jeweilige Clusterzugehörigkeit der Punkte ausgegeben. Auch für diese Interationen wurde die Trennung der Cluster in den Abbildungen skizziert.

```
In [1]: import numpy as np

        #Arrays, sollen Cluster repräsentieren. Punkte nach 1. Iteration in zugehörigem Array
        x0 = np.array([[1,1,3,3,4,5],[4,5,3,2,1,1]])
        x1 = np.array([[6,6,8,8,8],[2,3,4,5,6]])
        x2 = np.array([[1],[6]])

        #Array mit neuen Clusterzentren
        c = np.array([[np.mean(x0[0,:]), np.mean(x1[0,:]), np.mean(x2[0,:])], [np.mean(x0[1,:])

        #Iterationen
        for i in range(1,4):
            print('Iteration:')
            print(i+1)
            print('Clusterzentren (Reihenfolge 0,1,2):')
            for k in range(len(c[0,:])):
                print('(',c[0,k],',',c[1,k],')')

            #Berechne Abstände der Punkte in Cluster 0 zu neuen Clusterzentren
            b0 = np.sqrt((c[0,0] - x0[0,:])**2 + (c[1,0] - x0[1,:])**2)
            b1 = np.sqrt((c[0,1] - x0[0,:])**2 + (c[1,1] - x0[1,:])**2)
```

```
b2 = np.sqrt((c[0,2] - x0[0,:])**2 + (c[1,2] - x0[1,:])**2)

#Falls nun Abstand zu anderem Clusterzentrum geringer ist, lösche Punkt aus bisher
k=0
while k<len(b0):
    if b0[k]>b1[k] or b0[k]>b2[k]:
        if b1[k]>b2[k]:
            x2 = np.concatenate((x2, np.array([[x0[0,k]],[x0[1,k]]])), axis =-1)
        else:
            x1 = np.concatenate((x1, np.array([[x0[0,k]],[x0[1,k]]])), axis =-1)
        x0 = np.delete(x0, k, 1)
        b0 = np.delete(b0, k)
        b1 = np.delete(b1, k)
        b2 = np.delete(b2, k)
    else:
        k+=1

#Berechne Abstände der Punkte in Cluster 1 zu neuen Clusterzentren
b0 = np.sqrt((c[0,0] - x1[0,:])**2 + (c[1,0] - x1[1,:])**2)
b1 = np.sqrt((c[0,1] - x1[0,:])**2 + (c[1,1] - x1[1,:])**2)
b2 = np.sqrt((c[0,2] - x1[0,:])**2 + (c[1,2] - x1[1,:])**2)

#Falls nun Abstand zu anderem Clusterzentrum geringer ist, lösche Punkt aus bisher
k=0
while k<len(b1):
    if b1[k]>b0[k] or b1[k]>b2[k]:
        if b0[k]>b2[k]:
            x2 = np.concatenate((x2, np.array([[x1[0,k]],[x1[1,k]]])), axis =-1)
        else:
            x0 = np.concatenate((x0, np.array([[x1[0,k]],[x1[1,k]]])), axis =-1)
        x1 = np.delete(x1, k, 1)
        b0 = np.delete(b0, k)
        b1 = np.delete(b1, k)
        b2 = np.delete(b2, k)
    else:
        k+=1

#Berechne Abstände der Punkte in Cluster 2 zu neuen Clusterzentren
b0 = np.sqrt((c[0,0] - x2[0,:])**2 + (c[1,0] - x2[1,:])**2)
b1 = np.sqrt((c[0,1] - x2[0,:])**2 + (c[1,1] - x2[1,:])**2)
b2 = np.sqrt((c[0,2] - x2[0,:])**2 + (c[1,2] - x2[1,:])**2)

#Falls nun Abstand zu anderem Clusterzentrum geringer ist, lösche Punkt aus bisher
k=0
while k<len(b2):
    if b2[k]>b1[k] or b2[k]>b0[k]:
        if b1[k]>b0[k]:
            x0 = np.concatenate((x0, np.array([[x2[0,k]],[x2[1,k]]])), axis =-1)
```

```
                    else:
                        x1 = np.concatenate((x1, np.array([[x1[0,k]],[x1[1,k]]])), axis =-1)
                    x2 = np.delete(x2, k, 1)
                    b0 = np.delete(b0, k)
                    b1 = np.delete(b1, k)
                    b2 = np.delete(b2, k)
                else:
                    k+=1

            #Berechne neue Clusterzentren
            c = np.array([[np.mean(x0[0,:]), np.mean(x1[0,:]), np.mean(x2[0,:])], [np.mean(x0[

            #Gebe Punktzugehörigkeit zu entsprechendem Cluster aus
            print('Zu Cluster 0 gehören die folgenden Punkte:')
            for k in range(len(x0[0,:])):
                print('(',x0[0,k],',',x0[1,k],')')
            print('Zu Cluster 1 gehören die folgenden Punkte:')
            for k in range(len(x1[0,:])):
                print('(',x1[0,k],',',x1[1,k],')')
            print('Zu Cluster 2 gehören die folgenden Punkte:')
            for k in range(len(x2[0,:])):
                print('(',x2[0,k],',',x2[1,k],')')
            print('-------------------------------------------------------')
```

```
Iteration:
2
Clusterzentren (Reihenfolge 0,1,2):
( 2.8333333333333335 , 2.6666666666666665 )
( 7.2 , 4.0 )
( 1.0 , 6.0 )
Zu Cluster 0 gehören die folgenden Punkte:
( 3 , 3 )
( 3 , 2 )
( 4 , 1 )
( 5 , 1 )
Zu Cluster 1 gehören die folgenden Punkte:
( 6 , 2 )
( 6 , 3 )
( 8 , 4 )
( 8 , 5 )
( 8 , 6 )
Zu Cluster 2 gehören die folgenden Punkte:
( 1 , 6 )
( 1 , 4 )
( 1 , 5 )
-------------------------------------------------------
Iteration:
3
```

```
Clusterzentren (Reihenfolge 0,1,2):
( 3.75 , 1.75 )
( 7.2 , 4.0 )
( 1.0 , 5.0 )
Zu Cluster 0 gehören die folgenden Punkte:
( 3 , 3 )
( 3 , 2 )
( 4 , 1 )
( 5 , 1 )
( 6 , 2 )
Zu Cluster 1 gehören die folgenden Punkte:
( 6 , 3 )
( 8 , 4 )
( 8 , 5 )
( 8 , 6 )
Zu Cluster 2 gehören die folgenden Punkte:
( 1 , 6 )
( 1 , 4 )
( 1 , 5 )
-------------------------------------------------------
Iteration:
4
Clusterzentren (Reihenfolge 0,1,2):
( 4.2 , 1.8 )
( 7.5 , 4.5 )
( 1.0 , 5.0 )
Zu Cluster 0 gehören die folgenden Punkte:
( 3 , 3 )
( 3 , 2 )
( 4 , 1 )
( 5 , 1 )
( 6 , 2 )
Zu Cluster 1 gehören die folgenden Punkte:
( 6 , 3 )
( 8 , 4 )
( 8 , 5 )
( 8 , 6 )
Zu Cluster 2 gehören die folgenden Punkte:
( 1 , 6 )
( 1 , 4 )
( 1 , 5 )
-------------------------------------------------------
```

1.3 Teilaufgabe c)

Es wurde oben ein weiterer 5 Iterationsschritt durchgeführt. Es ist zu erkennen, dass kein Punkt mehr neu zugeordnet wird. Das Ergebnis entspricht nicht ganz den Erwartungen. Der Punkt (6,3)

wäre noch in Cluster 0 vermutet worden.