Artificial Intelligence

*COMP 5/6600, Fall2020*

HW04: HAC clustering

Remember that only PDF submissions are accepted.

# Requirement

Please submit your homework on Canvas before the deadline. If you need to use your delay coupon, please include your name, the student ID number, and how many days late it is (if handed in late) in the headline, and also make acomment onyoursubmission on Canvasfile submission system. Ifyouhave any questions, please contact our TA. (\*), (\*\*), or (\*\*\*) indicates the difficulty of each question.

# Policy

We apply the late policy explained in syllabus to all homework. Any grading questions must be raised with the TA in two weeks after the homework is returned. The homework must be completed individually. However, you are encouraged to discuss the general algorithms and ideas with classmates in order to help you answer the questions. You are also allowed to share examples that are not on the homework in order to demonstrate how to solve problems. If you work with one or more other people on the general discussion of the assignment questions, please record their names over every question they participated. However,the following behaviors will receive heavy penalties (lose all points and apply the honest policy explained in syllabus)

* explicitly tell somebody else the answers;
* explicitly copy answers or code fragments from anyone or anywhere;
* allow your answers to be copied;
* get code from Web.

1

*HW04: HAC clustering* 2

# (\*) Hierarchical clustering

Suppose you have N points. Consider a brute force algorithm for agglomerative clustering using the single-link (minimum distance) criteria. Forapairofsets *A* and *B*, tocompute thedistance between themthe algorithm compares all pairs of distances between the elements of set *A* and set *B* to find the minimum. The time taken to do this is *O*(*|A||B|*) where *|A|* is the size of set *A*. What is the complexity of this algorithm (in “big *O*” notion)

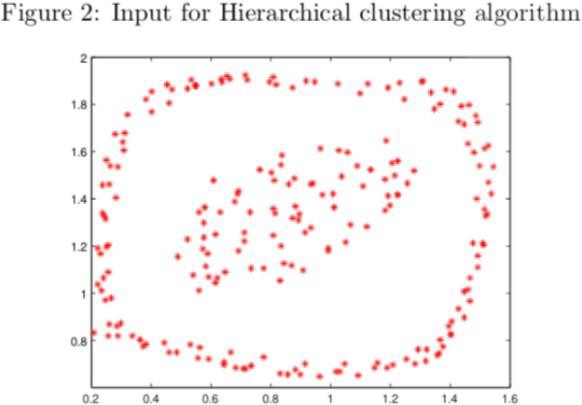
for clustering N points?

O(N2)

(proof on the last page)

# (\*\*) Hierarchical clustering: 8 points(programming homework)

Implement the hierarchical clustering algorithm. Choose the number of clusters as 2. The input data is a set of 2-D points shown in Figure 2 (You can download the coordinate file in “B.txt”). The reasonable clustering result is that the points in the center is a group while the points surrounding them is the other group. You should properly choose your cluster distance.



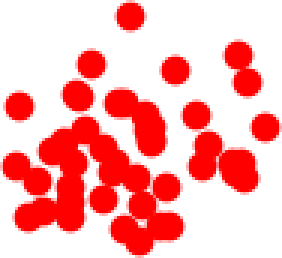
You need to report

* + your clustering result shown in a figure (you should use differentsigns to mark 2 clusters; Photoshop or other similar softwares are forbidden in this problem);
  + ~~your answer to the following question: can you get the same result using K-means and state your reason~~.

2.00

1.75

1.50

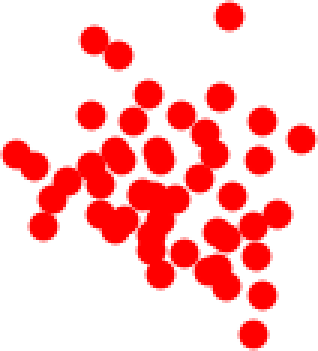
>, 1.25

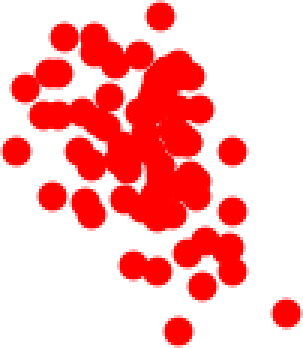
1.00

0.75

0.50

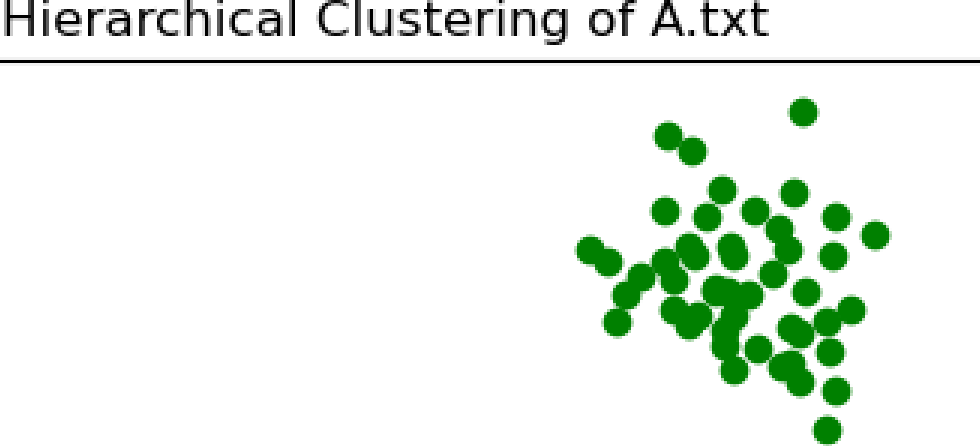
Scatter **Plot** of A.txt



0.2 0.4 0.6 0.8

**X**

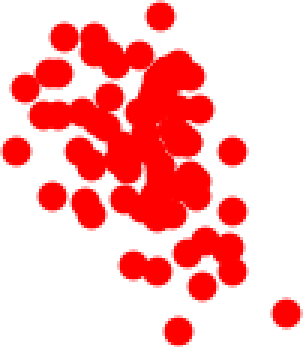
1.0 1.2 1.4

2.00

1.75

1.50

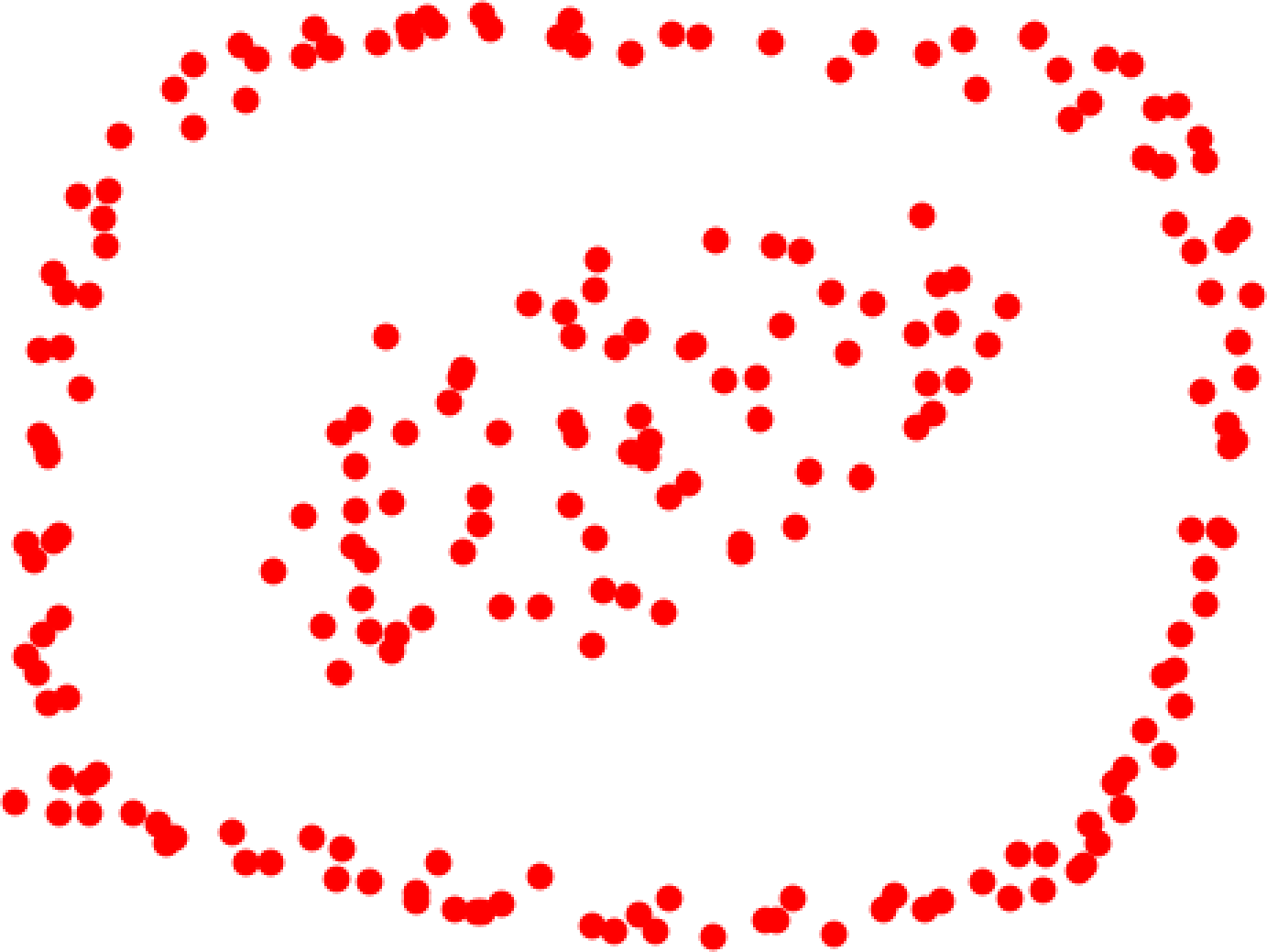
>, 1.25

1.00

0.75

0.50

0.2 0.4 0.6 0.8 1.0 1.2 1.4

Scatter **Plot** of B.txt

1.8 -

1.6 -

1.4 -

1.2 -

1.0 -

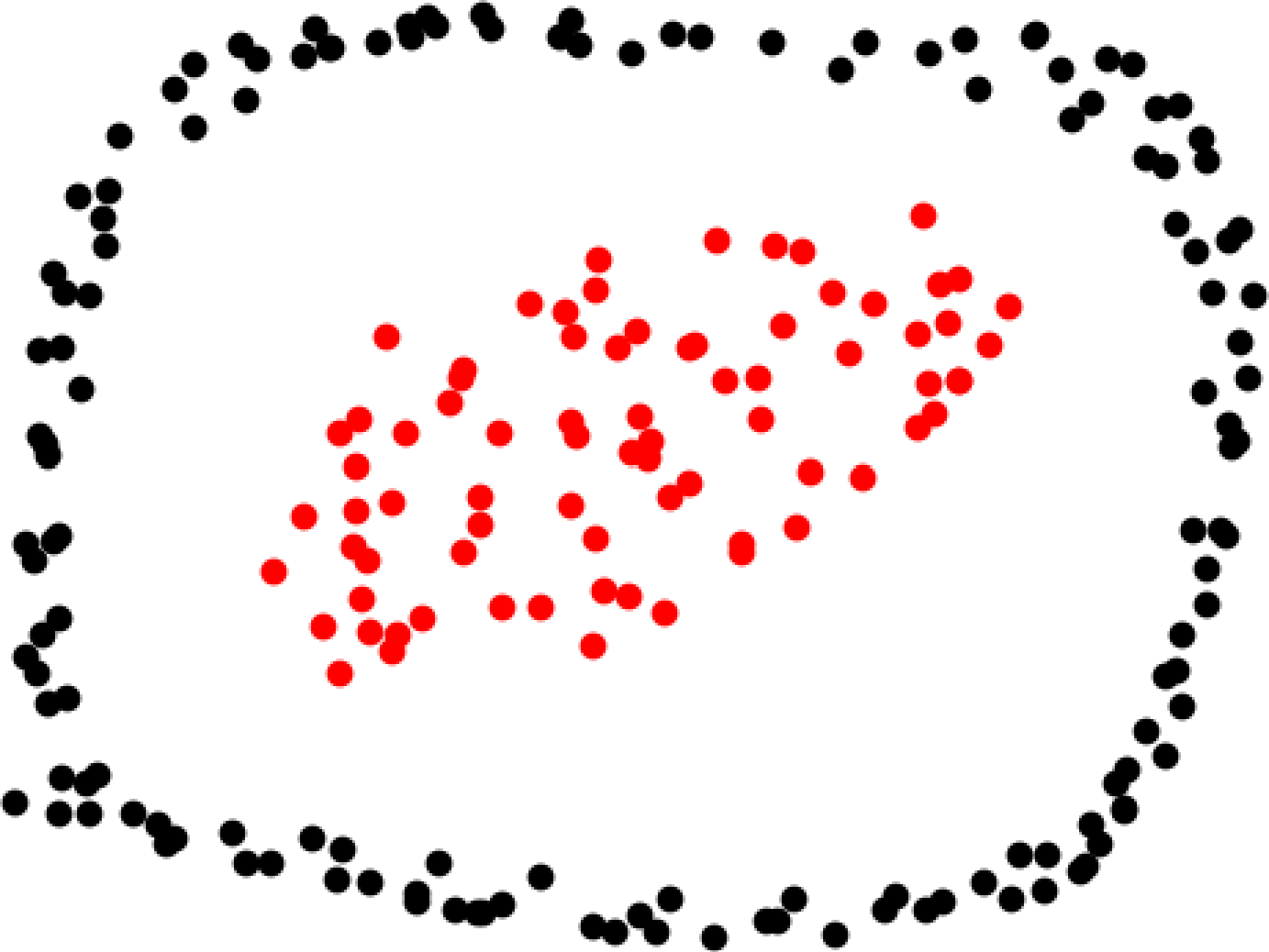
0.8 -

0.6 -... ,.,-- ,--- ,-- ,-- ,-- ,-- ,---,

a2 a4 a6 a8 io i2 L4 L6

**X**

|  |  |  |
| --- | --- | --- |
|  | 1H ierarclhical Cll.llstering of | B.t xt |
| 1.8 - |  |  |
| 1.6 - |  |  |
| 1.4 - |  |  |
| 1.2 - |  |  |
| 1.0 - |  |  |
| 0.8 - |  |  |
| 0.6 - | -.---------r--------r-------r--------r------r-------r r | |

0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6

"""

Seth Kinsaul [smk0036@auburn.edu](mailto:smk0036@auburn.edu) 9/19/2020

COMP 5600 Aritifical Intelligence

Hierarchical clustering algorithm on cordinates given in .txt files """

#imports

import numpy as np

from matplotlib import pyplot as plt class data:

def init (self, file\_content, \*colors, graph\_name): self.file\_content = file\_content

self.colors = colors self.graph\_name = graph\_name self.clusters = [] #list of clusters

def init\_clusters(self):

for x in self.file\_content: self.clusters.append([[(x[0],x[1])],0])

def display\_plot(self): plt.figure()

plt.scatter(self.file\_content[:,0], self.file\_content[:,1], color = "red") plt.title("Scatter Plot of %s.txt" % self.graph\_name) plt.xlabel("x")

plt.ylabel("y") plt.show()

def display\_clusters(self): plt.figure()

for i,x in enumerate(self.clusters): pts = x[0]

x = []

y = []

for pt in pts: x.append(pt[0])

y.append(pt[1])

plt.scatter(x, y, color = '%s' % colors[i]) plt.title("Hierarchical Clustering of %s.txt" % self.graph\_name) plt.xlabel("x")

plt.ylabel("y") plt.show()

def hier\_cluster(self, k): self.init\_clusters()

while len(self.clusters) > k: best\_d = 9999

best\_pt = self.clusters[0][0] best\_pt2 = self.clusters[1][0] best\_distance\_x = 0

best\_distance\_x2 = 1

for x in range(len(self.clusters)): for y in range(len(self.clusters)):

if y == x: continue else:

pt = self.clusters[x][0] pt2 = self.clusters[y][0]

x\_pt = []

y\_pt = []

for i in pt: x\_pt.append(i[0]) y\_pt.append(i[1])

x\_pt2 = [] y\_pt2 = [] for i in pt2:

x\_pt2.append(i[0]) y\_pt2.append(i[1])

sub\_d\_best = 9999 for i in range(len(pt)):

for j in range(len(pt2)):

delx = (x\_pt[i]-x\_pt2[j])\*\*2 dely = (y\_pt[i]-y\_pt2[j])\*\*2 sub\_d = np.sqrt(delx+dely) if sub\_d <= sub\_d\_best:

sub\_d\_best = sub\_d d = sub\_d\_best

if d <= best\_d: best\_d = d

best\_pt = self.clusters[x] best\_pt2 = self.clusters[y] best\_distance\_x = x best\_distance\_x2 = y

temp\_pt\_list = []

temp\_pt\_list = self.clusters[best\_distance\_x][0] + self.clusters[best\_distance\_x2][0] self.clusters.remove(best\_pt)

self.clusters.remove(best\_pt2) self.clusters.append([temp\_pt\_list,best\_d])

if name == ' main ': #load A.txt and B.txt

a = np.loadtxt('A.txt') b = np.loadtxt('B.txt') #colors for clusters

colors = np.array(["red", "black", "green", "yellow"])

#set\_a

set\_a = data(a, colors, graph\_name = 'A') set\_a.hier\_cluster(3)

set\_a.display\_plot() set\_a.display\_clusters()

#set\_b

set\_b = data(b, colors, graph\_name = 'B') set\_b.hier\_cluster(2)

set\_b.display\_plot() set\_b.display\_clusters()

Proof for Number 1:

