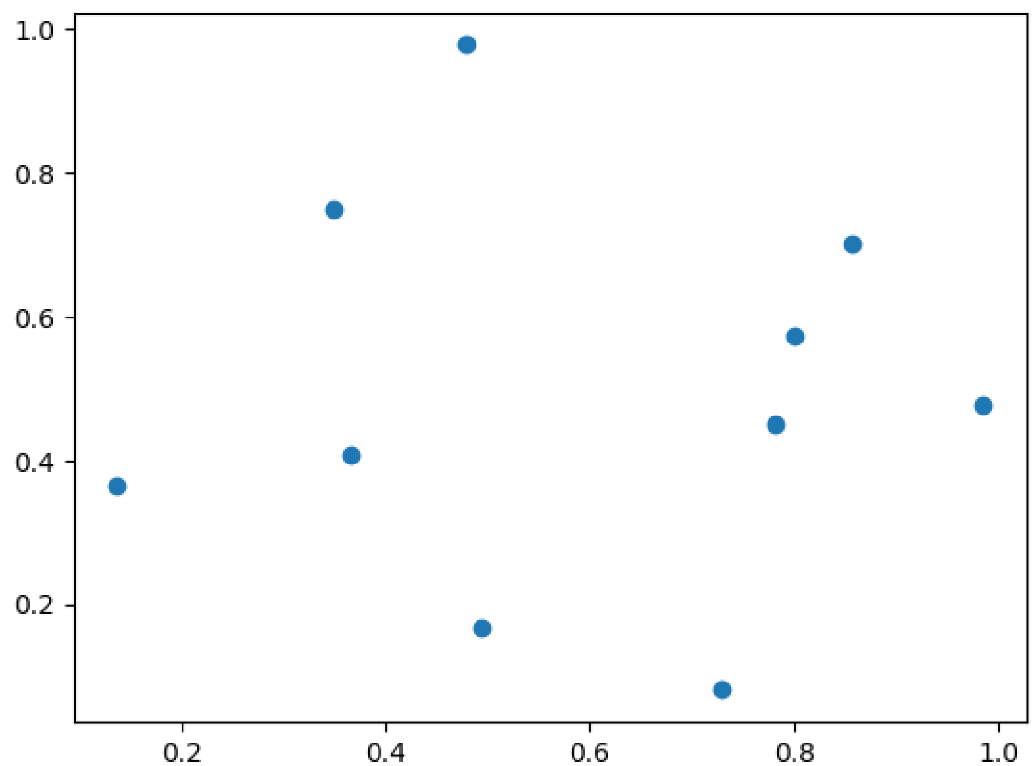
HW4

October 5, 2024

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import math
import scipy as sp
```

[20]:
```
# Create a data (for HW4 you will read from a file)
n = 10
x = np.random.rand(n, 2) # 5 data items with 2 features
plt.scatter(x[:, 0], x[:, 1])
```

[20]: <matplotlib.collections.PathCollection at 0x265c52a8150>



[21]:
```python
# Find the distance matrix
d = np.zeros((n, n))
for i in range(n):
    for j in range(0, i):
        d[i, j] = np.linalg.norm(x[i] - x[j]) # distance between item i and␣
↪item j
        d[j, i] = d[i, j]
np.round(d, 2)
```

[21]:
```
array([[0.  , 0.65, 0.42, 0.37, 0.13, 0.53, 0.26, 0.61, 0.4 , 0.21],
       [0.65, 0.  , 0.23, 0.66, 0.7 , 0.44, 0.8 , 0.7 , 0.41, 0.86],
       [0.42, 0.23, 0.  , 0.49, 0.47, 0.34, 0.57, 0.58, 0.27, 0.62],
       [0.37, 0.66, 0.49, 0.  , 0.5 , 0.77, 0.63, 0.93, 0.25, 0.47],
       [0.13, 0.7 , 0.47, 0.5 , 0.  , 0.48, 0.14, 0.52, 0.51, 0.21],
       [0.53, 0.44, 0.34, 0.77, 0.48, 0.  , 0.51, 0.26, 0.6 , 0.69],
       [0.26, 0.8 , 0.57, 0.63, 0.14, 0.51, 0.  , 0.47, 0.64, 0.26],
       [0.61, 0.7 , 0.58, 0.93, 0.52, 0.26, 0.47, 0.  , 0.81, 0.71],
       [0.4 , 0.41, 0.27, 0.25, 0.51, 0.6 , 0.64, 0.81, 0.  , 0.58],
       [0.21, 0.86, 0.62, 0.47, 0.21, 0.69, 0.26, 0.71, 0.58, 0.  ]])
```

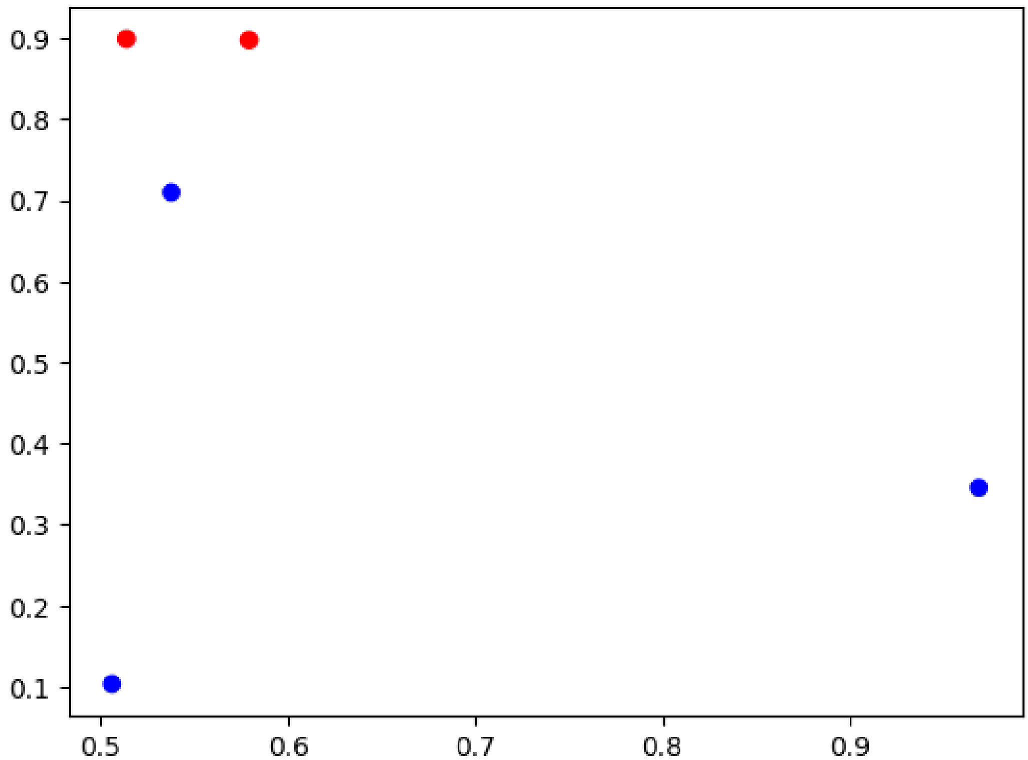
[5]:
```python
# Instead of removing columns and rows (like in lecture example), we are going␣
↪to add columns or rows for new clusters.
ad = np.ones((2 * n, 2 * n)) * math.inf # we are going to find minimum␣
↪distances (to merge clusters)
for i in range(n):
    for j in range(i + 1, n): # row index always smaller than col index (for␣
↪tie breaking rule)
        ad[i, j] = d[i, j]
np.round(ad, 2)
```

[5]:
```
array([[ inf, 0.52, 0.61, 0.8 , 0.8 ,  inf,  inf,  inf,  inf,  inf],
       [ inf,  inf, 0.56, 0.68, 0.72,  inf,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf, 0.19, 0.19,  inf,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf,  inf, 0.07,  inf,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf]])
```

[8]:
```python
# Find the closest two clusters (items), expected: merge row 4 (indexed 3) and␣
↪col 5 (indexed 4)
index = np.argmin(ad) # this is the index of the element in the flattened array␣
↪corresponding to the smallest distance.
(row, col) = np.unravel_index(index, ad.shape) # finding the row and col number␣
↪of index.
```

```
color = ["blue"] * 5
color[row] = "red"
color[col] = "red"
plt.scatter(x[:, 0], x[:, 1], c = color)
```

[8]: <matplotlib.collections.PathCollection at 0x265a5e04d50>



[13]:
```
# Merge the two clusters, update the distance matrix (add 1 col and 1 row to␣
↪represent the new cluster, remove 2 rows and cols.
c = 0 # iteration number
for i in range(n + c):
    ad[i, n + c] = min(ad[min(i, row), max(i, row)], ad[min(i, col), max(col,␣
↪i)]) # We are doing single-linkage, that's why it's a min
    # Trick 1: ad[a, b] is the distance only when a < b, use ad[min(a, b),␣
↪max(a, b)]
    # Trick 2: (see lecture example), dist(a, {b, c}) = min{dist(a, b), dist(a,␣
↪c)} due to single-linkage.
    # Trick 3: keep track of upper triangular matrix, only need to add column␣
↪(not row).
    # Result: the column contains distances from the merged cluster to existing␣
↪clusters.
```

```
np.round(ad, 2)
```

[13]:
```
array([[ inf, 0.52, 0.61, 0.8 , 0.8 , 0.8 ,  inf,  inf,  inf,  inf],
       [ inf,  inf, 0.56, 0.68, 0.72, 0.68,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf, 0.19, 0.19, 0.19,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf,  inf, 0.07, 0.07,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf,  inf,  inf, 0.07,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf]])
```

[15]:
```
# Update the cluster
cluster = {i: [i] for i in range(n)}
cluster[n + c] = cluster[row] + cluster[col]
cluster
```

[15]:
```
{0: [0], 1: [1], 2: [2], 3: [3], 4: [4], 5: [3, 4]}
```

[16]:
```
# Update output matrix
output = np.zeros((n - 1, 4))
output[c, 0] = row
output[c, 1] = col
output[c, 2] = ad[row, col] # since row < col
output[c, 3] = len(cluster[n + c]) # size of the new cluster
np.round(output[c, :], 2)
```

[16]:
```
array([3.  , 4.  , 0.07, 2.  ])
```

[17]:
```
# Remove the merged clusters
ad[row, :] = math.inf
ad[:, row] = math.inf
ad[col, :] = math.inf
ad[:, col] = math.inf
np.round(ad, 2)
```

[17]:
```
array([[ inf, 0.52, 0.61,  inf,  inf, 0.8 ,  inf,  inf,  inf,  inf],
       [ inf,  inf, 0.56,  inf,  inf, 0.68,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf,  inf,  inf, 0.19,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf],
       [ inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf,  inf]])
```

[22]:
```python
# Combine everything:

# Set up everything (distance matrix, cluster list, output)
ad = np.ones((2 * n, 2 * n)) * math.inf
for i in range(n):
    for j in range(i + 1, n):
        ad[i, j] = d[i, j]
cluster = {i: [i] for i in range(n)}
output = np.zeros((n - 1, 4))

# Main loop
for c in range(n - 1):
    # Merge step
    index = np.argmin(ad)
    (row, col) = np.unravel_index(index, ad.shape)
    # Update distance
    for i in range(n + c):
        ad[i, n + c] = min(ad[min(i, row), max(i, row)], ad[min(i, col),␣
 ↪max(col, i)])
    # Update cluster
    cluster[n + c] = cluster[row] + cluster[col]
    # Update output
    output[c, 0] = row
    output[c, 1] = col
    output[c, 2] = ad[row, col]
    output[c, 3] = len(cluster[n + c])
    # Clean up distance
    ad[row, :] = math.inf
    ad[:, row] = math.inf
    ad[col, :] = math.inf
    ad[:, col] = math.inf

cluster
```

[22]:
```
{0: [0],
 1: [1],
 2: [2],
 3: [3],
 4: [4],
 5: [5],
 6: [6],
 7: [7],
 8: [8],
 9: [9],
 10: [0, 4],
 11: [6, 0, 4],
 12: [9, 6, 0, 4],
```

```
 13: [1, 2],
 14: [3, 8],
 15: [5, 7],
 16: [1, 2, 3, 8],
 17: [5, 7, 1, 2, 3, 8],
 18: [9, 6, 0, 4, 5, 7, 1, 2, 3, 8]}
```

[23]:
```python
# Plot the dendrogram
sp.cluster.hierarchy.dendrogram(output)
```

[23]:

```
{'icoord': [[25.0, 25.0, 35.0, 35.0],
  [15.0, 15.0, 30.0, 30.0],
  [5.0, 5.0, 22.5, 22.5],
  [45.0, 45.0, 55.0, 55.0],
  [65.0, 65.0, 75.0, 75.0],
  [85.0, 85.0, 95.0, 95.0],
  [70.0, 70.0, 90.0, 90.0],
  [50.0, 50.0, 80.0, 80.0],
  [13.75, 13.75, 65.0, 65.0]],
 'dcoord': [[0.0, 0.12558580080674156, 0.12558580080674156, 0.0],
  [0.0, 0.1391740458632717, 0.1391740458632717, 0.12558580080674156],
  [0.0, 0.2060354281777875, 0.2060354281777875, 0.1391740458632717],
  [0.0, 0.2639818896113143, 0.2639818896113143, 0.0],
  [0.0, 0.23314352949461054, 0.23314352949461054, 0.0],
  [0.0, 0.2510655645839191, 0.2510655645839191, 0.0],
  [0.23314352949461054,
   0.2724642205241889,
   0.2724642205241889,
   0.2510655645839191],
  [0.2639818896113143,
   0.3407811383410931,
   0.3407811383410931,
   0.2724642205241889],
  [0.2060354281777875,
   0.3729837381250288,
   0.3729837381250288,
   0.3407811383410931]],
 'ivl': ['9', '6', '0', '4', '5', '7', '1', '2', '3', '8'],
 'leaves': [9, 6, 0, 4, 5, 7, 1, 2, 3, 8],
 'color_list': ['C1', 'C1', 'C1', 'C0', 'C2', 'C3', 'C0', 'C0', 'C0'],
 'leaves_color_list': ['C1',
  'C1',
  'C1',
  'C1',
  'C0',
  'C0',
  'C2',
```

```
'C2',
'C3',
'C3']}
```