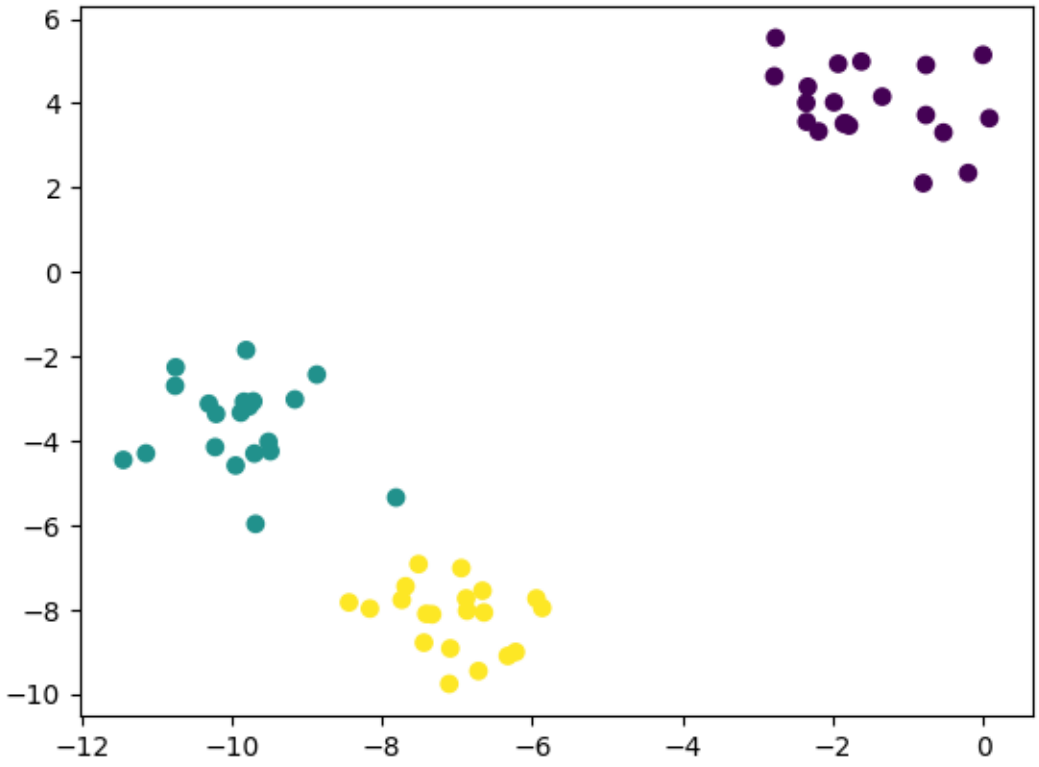
HCA

July 23, 2024

1 Hierarchische Clusteranalyse (HCA)

[1]:
```
import numpy as np
from sklearn.datasets import make_blobs
from matplotlib import pyplot as plt

# generate data
X, y = make_blobs(n_samples=60, n_features=2, centers=3, random_state=1)
```

[2]:
```
plt.scatter(*X.T, c=y)
plt.show()
```



Relevante Größen für das HCA ist das - *dissimilarity measure*, ein Unterschiedlichkeitsmaß auf

Ebene der Datenpunkte (z.B. euklidische Distanz oder Korrelationskoeffizient) und - *linkage*, ein Unterschiedlichkeitsmaß auf Ebene der Cluster (z.B. complete, single, average).

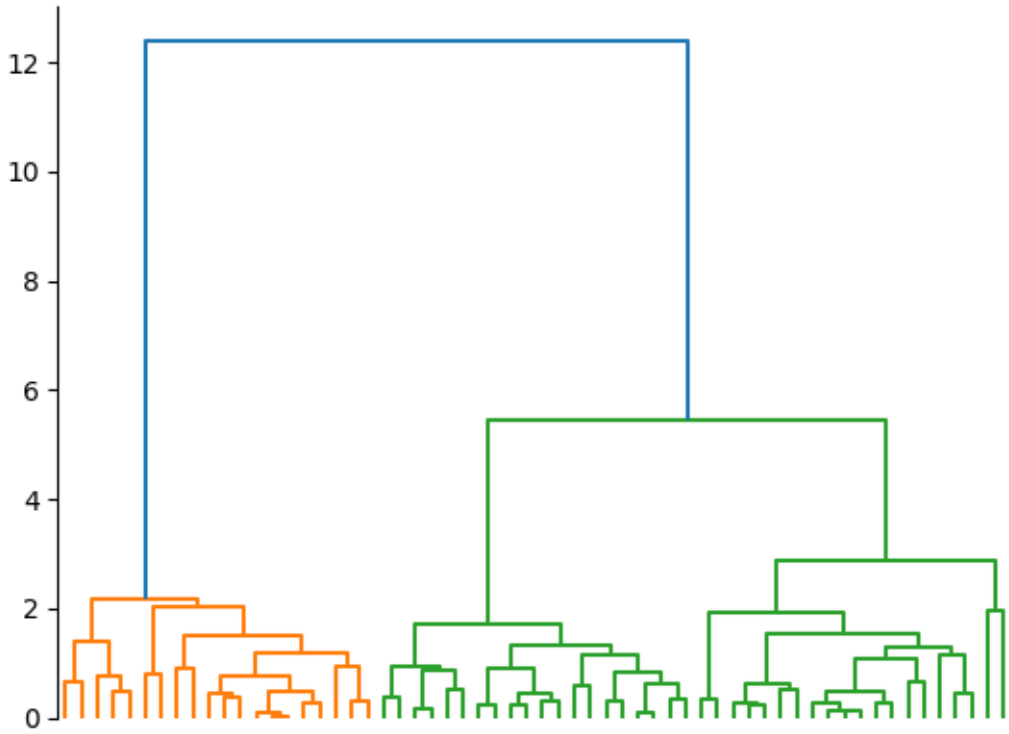
Beim **Complete Linkage** (maximale Inter-Cluster-Unterschiedlichkeit) werden alle paarweisen Unterschiedlichkeiten zwischen Punkten aus dem ersten und Punkten aus dem zweiten Cluster bestimmt und das Maximum genommen.

Beim **Single Linkage** (minimale Inter-Cluster-Unterschiedlichkeit) wird die minimale paarweise Unterschiedlichkeit zwischen den Punkten aus den Clustern verwendet.

Beim **Average Linkage** (mittlere Inter-Cluster-Unterschiedlichkeit) wird der Mittelwert der paarweisen Unterschiedlichkeiten zwischen den Punkten aus den Clustern verwendet.

[3]:
```
def dissimilarity(x1, x2) -> float:
    return np.linalg.norm(x1 - x2)

def linkage(C1, C2, type: str = 'average') -> float:
    diss_pairs: list[float] = []
    for c1 in C1:
        for c2 in C2:
           assert c1 != c2, "Hier ist ein Punkt doppelt?!"
           diss_pairs.append(dissimilarity(c1, c2))

    diss_np: np.ndarray = np.array(diss_pairs)
    if type == 'average':
        return diss_np.mean()
    elif type == 'single':
        return diss_np.min()
    elif type == 'complete':
        return diss_np.max()
```

[4]:
```
from scipy.cluster import hierarchy
Z = hierarchy.linkage(X, 'average')

fig, axs = plt.subplots(1, 1)
dn = hierarchy.dendrogram(Z, truncate_mode='none', show_leaf_counts=True,␣
 ↪no_labels=True)
#plt.axis('off')

axs.spines['top'].set_visible(False)
axs.spines['right'].set_visible(False)
axs.spines['bottom'].set_visible(False)

plt.show()
```

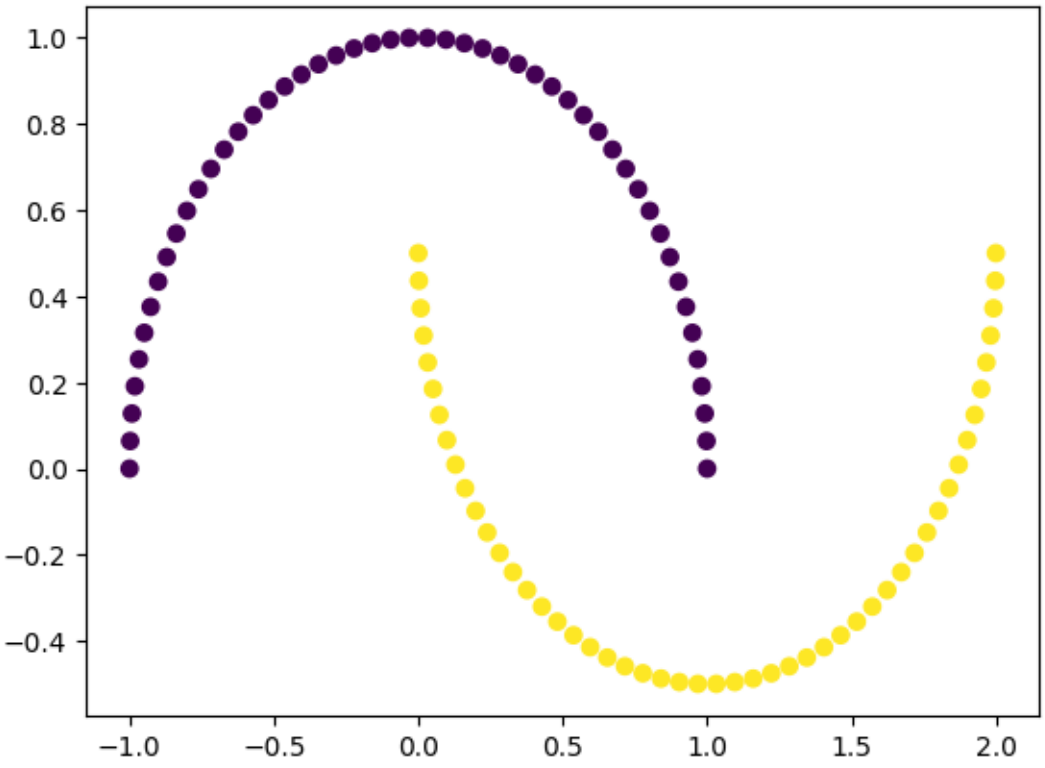


[5]:
```
Z
```

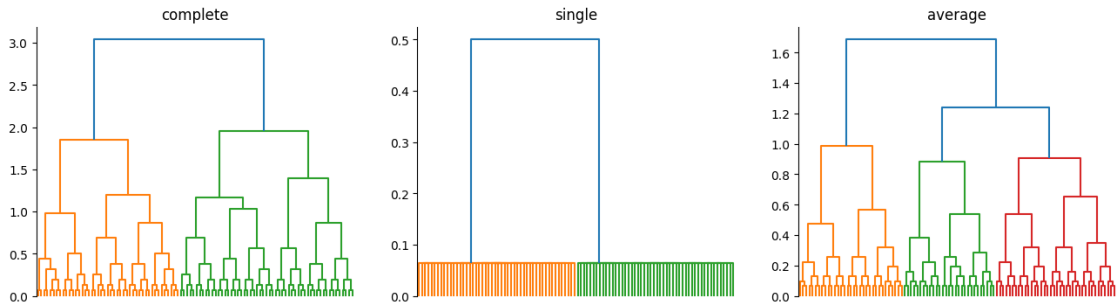
[5]:
```
array([[1.50000000e+01, 3.10000000e+01, 2.17280603e-02, 2.00000000e+00],
       [7.00000000e+00, 3.40000000e+01, 7.84081035e-02, 2.00000000e+00],
       [1.40000000e+01, 6.00000000e+01, 8.01486344e-02, 3.00000000e+00],
       [4.10000000e+01, 4.30000000e+01, 1.25870441e-01, 2.00000000e+00],
       [3.90000000e+01, 6.30000000e+01, 1.34498817e-01, 3.00000000e+00],
       [9.00000000e+00, 1.00000000e+01, 1.41502638e-01, 2.00000000e+00],
       [5.00000000e+01, 5.20000000e+01, 2.19544043e-01, 2.00000000e+00],
       [2.20000000e+01, 2.80000000e+01, 2.29017234e-01, 2.00000000e+00],
       [2.50000000e+01, 4.50000000e+01, 2.36786207e-01, 2.00000000e+00],
       [5.70000000e+01, 6.40000000e+01, 2.52907290e-01, 4.00000000e+00],
       [1.30000000e+01, 1.60000000e+01, 2.60230122e-01, 2.00000000e+00],
       [4.70000000e+01, 5.90000000e+01, 2.77771987e-01, 2.00000000e+00],
       [2.70000000e+01, 6.60000000e+01, 2.79279277e-01, 3.00000000e+00],
       [1.20000000e+01, 3.30000000e+01, 2.84149341e-01, 2.00000000e+00],
       [0.00000000e+00, 3.80000000e+01, 3.14084714e-01, 2.00000000e+00],
       [5.00000000e+00, 3.70000000e+01, 3.14507818e-01, 2.00000000e+00],
       [2.40000000e+01, 5.50000000e+01, 3.24447961e-01, 2.00000000e+00],
       [3.50000000e+01, 5.10000000e+01, 3.39150555e-01, 2.00000000e+00],
       [1.10000000e+01, 4.60000000e+01, 3.69461861e-01, 2.00000000e+00],
       [2.60000000e+01, 4.20000000e+01, 3.76273196e-01, 2.00000000e+00],
       [6.70000000e+01, 7.30000000e+01, 4.33347292e-01, 4.00000000e+00],
```

```
[1.70000000e+01, 1.80000000e+01, 4.39646057e-01, 2.00000000e+00],
[3.20000000e+01, 7.80000000e+01, 4.47076194e-01, 3.00000000e+00],
[6.20000000e+01, 7.10000000e+01, 4.67010940e-01, 5.00000000e+00],
[8.00000000e+00, 3.60000000e+01, 4.76907410e-01, 2.00000000e+00],
[6.90000000e+01, 7.00000000e+01, 4.91126811e-01, 6.00000000e+00],
[2.30000000e+01, 2.90000000e+01, 4.93573171e-01, 2.00000000e+00],
[2.10000000e+01, 4.40000000e+01, 5.14690628e-01, 2.00000000e+00],
[4.00000000e+00, 5.80000000e+01, 5.76830510e-01, 2.00000000e+00],
[7.20000000e+01, 8.70000000e+01, 6.11822772e-01, 5.00000000e+00],
[6.10000000e+01, 7.60000000e+01, 6.15691415e-01, 4.00000000e+00],
[3.00000000e+00, 4.80000000e+01, 6.43680235e-01, 2.00000000e+00],
[1.00000000e+00, 5.60000000e+01, 6.57439507e-01, 2.00000000e+00],
[8.20000000e+01, 8.30000000e+01, 7.49255371e-01, 8.00000000e+00],
[5.30000000e+01, 8.40000000e+01, 7.73065958e-01, 3.00000000e+00],
[2.00000000e+00, 5.40000000e+01, 7.95153274e-01, 2.00000000e+00],
[7.40000000e+01, 9.00000000e+01, 8.17698606e-01, 6.00000000e+00],
[6.50000000e+01, 8.60000000e+01, 8.46906210e-01, 4.00000000e+00],
[6.80000000e+01, 8.00000000e+01, 8.91659240e-01, 6.00000000e+00],
[3.00000000e+01, 4.90000000e+01, 9.10461022e-01, 2.00000000e+00],
[1.90000000e+01, 7.50000000e+01, 9.26479534e-01, 3.00000000e+00],
[7.90000000e+01, 9.70000000e+01, 9.47594946e-01, 6.00000000e+00],
[8.50000000e+01, 9.20000000e+01, 1.07425674e+00, 8.00000000e+00],
[4.00000000e+01, 8.10000000e+01, 1.14620027e+00, 3.00000000e+00],
[8.80000000e+01, 9.60000000e+01, 1.15873747e+00, 8.00000000e+00],
[9.30000000e+01, 1.00000000e+02, 1.18712115e+00, 1.10000000e+01],
[1.02000000e+02, 1.03000000e+02, 1.29296893e+00, 1.10000000e+01],
[9.80000000e+01, 1.04000000e+02, 1.32422272e+00, 1.40000000e+01],
[9.10000000e+01, 9.40000000e+01, 1.40895941e+00, 5.00000000e+00],
[9.90000000e+01, 1.05000000e+02, 1.49390755e+00, 1.30000000e+01],
[8.90000000e+01, 1.06000000e+02, 1.53631806e+00, 1.60000000e+01],
[1.01000000e+02, 1.07000000e+02, 1.69966602e+00, 2.00000000e+01],
[7.70000000e+01, 1.10000000e+02, 1.93718720e+00, 1.80000000e+01],
[6.00000000e+00, 2.00000000e+01, 1.97184784e+00, 2.00000000e+00],
[9.50000000e+01, 1.09000000e+02, 2.03221798e+00, 1.50000000e+01],
[1.08000000e+02, 1.14000000e+02, 2.15349804e+00, 2.00000000e+01],
[1.12000000e+02, 1.13000000e+02, 2.85779533e+00, 2.00000000e+01],
[1.11000000e+02, 1.16000000e+02, 5.44979660e+00, 4.00000000e+01],
[1.15000000e+02, 1.17000000e+02, 1.24091400e+01, 6.00000000e+01]])
```

[6]:
```
from sklearn.datasets import make_moons
[X2, y2] = make_moons(random_state=1)

plt.scatter(*X2.T, c=y2)
plt.show()
```



[7]:
```
fig, axs = plt.subplots(1, 3, figsize=(16, 4))

for i, m in enumerate(['complete', 'single', 'average']):
    Z = hierarchy.linkage(X2, m)
    hierarchy.dendrogram(Z, truncate_mode='none', show_leaf_counts=True,␣
 ↪no_labels=True, ax=axs[i])

    axs[i].set_title(m)
    axs[i].spines['top'].set_visible(False)
    axs[i].spines['right'].set_visible(False)
    axs[i].spines['bottom'].set_visible(False)

plt.show()
```



Nun mit NumPY

[8]:
```
from scipy.spatial.distance import cdist

def hca_numpy(X: np.ndarray, K: int, linkage_type: str = 'average'):
    # --- Distanzmatrix ---
    d = cdist(X, X)

    # --- Erstelle n Cluster ---
    y_pred = np.array(range(X.shape[0]))

    while True:
        # --- Performance ---
        C = np.unique(y_pred)

        # --- Wiederhole bis K-Cluster ---
        if C.shape[0] <= K:
            break

        if linkage_type == 'average':
            # Average Linkage
            linkage = np.array([[d[y_pred==i][:,y_pred==j].mean() if i!=j else␣
 ↪np.inf for j in C] for i in C])
        elif linkage_type == 'single':
            # Single Linkage
            linkage = np.array([[d[y_pred==i][:,y_pred==j].min() if i!=j else␣
 ↪np.inf for j in C] for i in C])
        elif linkage_type == 'complete':
            # Complete Linkage
            linkage = np.array([[d[y_pred==i][:,y_pred==j].max() if i!=j else␣
 ↪np.inf for j in C] for i in C])

        # --- Kombiniere gleichartige Cluster ---
        idx = linkage.argmin()
        i = idx % C.shape[0]
```

```
        j = idx // C.shape[0]

        y_pred[y_pred==C[i]] = C[j]

    # --- Visualisierung ---
    plt.scatter(*X.T, c=y_pred)
    plt.show()


hca_numpy(X2, 2, linkage_type='single')
```