AB8

October 13, 2023

[2]:
```
import matplotlib.pyplot as plt
import numpy as np
import scipy as sp
```

0.1 Maximalfluss und minimaler Schnitt

0.1.1 a)

Zielfunktion: $\sum f_{sv} = \sum f_{vt} \Rightarrow max(s,a) + (s,b) = (e,t) + (f,t)$,

Nebenbedingungen: $\forall e \in E : 1. f_{uv} \leq c_{uv} 0; 2. \sum f_{uv} - \sum f_{vw} = 0; 3. f(e) \geq 0$

[3]:
```
#Kantenindizes:
#[0]=sa, [1]=sb, [2]=ab, [3]=ac, [4]=ad, [5]=bd, [6]=bf, [7]=cd,
#[8]=ce, [9]=dc, [10]=de, [11]=df, [12]=ef, [13]=et, [14]=fe, [15]=ft

target_edges = [-1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ,0 , 0, 0, 0]
 ↪ #Zielfunktion sa+sb (bzw. -sa-sb)

constraint1_ub_mat = np.identity(len(target_edges), dtype=int)
 ↪ #Koeffizientenatrix für NB1
constraint1_ub_vec = [12,8,7,3,9,5,2,5,6,2,2,11,8,16,6,11]
 ↪ #b=Kantengewichte für NB1


 ↪ #MKoeffizientenatrix für NB2:
constraint2_eq_mat = [[1, 0, -1, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 ↪ #sa-ab-ac-ad
                       [0, 1, 1, 0, 0, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 ↪ #sb+ab-bd-bf
                       [0, 0, 0, 1, 0, 0, 0, -1, -1, 1, 0, 0, 0, 0, 0, 0],
 ↪ #ac+dc-cd-ce
                       [0, 0, 0, 0, 1, 1, 0, 1, 0, -1, -1, -1, 0, 0, 0, 0],
 ↪ #ad+bd+cd-dc-de-df
                       [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, -1, -1, 1, 0],
 ↪ #ce+de+fe-ef-et
                       [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, -1, -1]]
 ↪ #bf+df+ef-fe-ft
```

```
constraint2_eq_vec = [0 for _ in range(6)]
 ↪ #b=Nullvektor für NB2

max_flow = sp.optimize.linprog(c = target_edges, A_ub = constraint1_ub_mat,
 ↪ #Löse min -sa-sb
                               b_ub = constraint1_ub_vec,
 ↪ #NB3 per default setting:
                               A_eq = constraint2_eq_mat,
 ↪ #bounds=(0,none)
                               b_eq = constraint2_eq_vec)

print("Maximaler Fluss: ", np.asarray(max_flow.x, dtype = 'int'))
print("Maximalwert: ", int(-max_flow.fun))
```

```
Maximaler Fluss:  [12  7  0  3  9  5  2  0  5  2  1 11  0 12  6  7]
Maximalwert:  19
```

0.1.2 b)

Minimaler Schnitt: {(s,a), (b,d), (b,f)}

0.2 Trennende Hyperebene

0.2.1 a)

$max\delta$ unter den Nebenbedingungen:

$1. x_i a + b + \delta \leq y_i$ für die Punkte aus Gruppe 1, $a, b \in \mathbb{R}$

$2. x_i a + b - \delta \geq y_i \iff -x_i a - b + \delta \leq -y_i$ für die Punkte aus Gruppe 2, $a, b \in \mathbb{R}$,

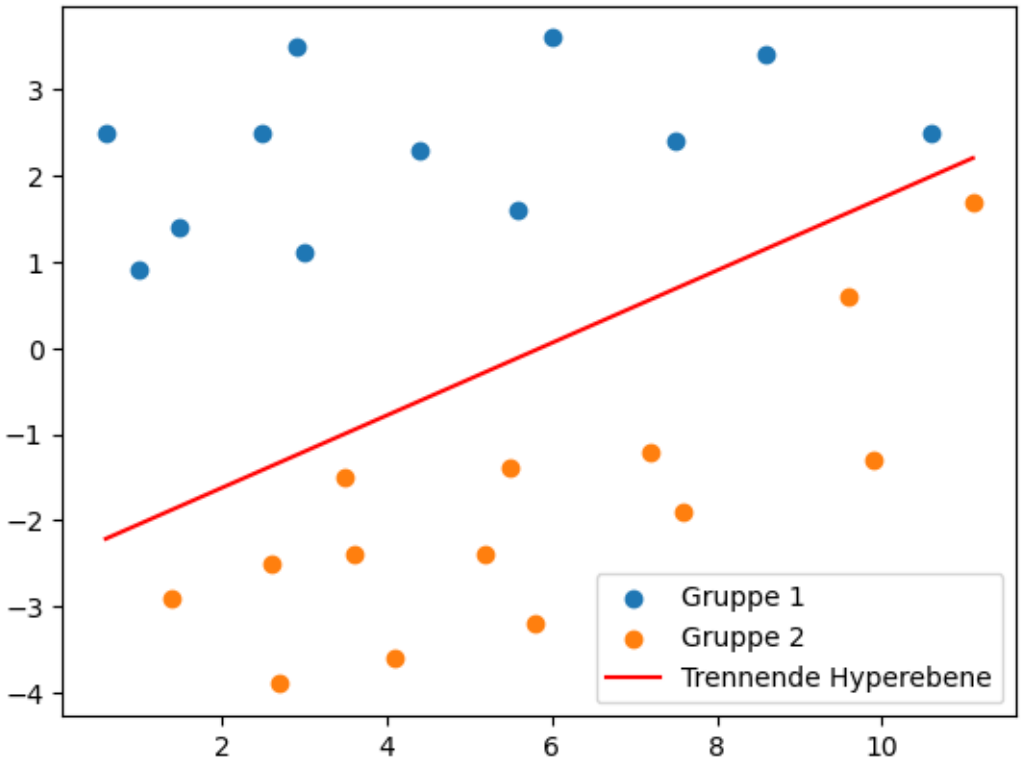
$3. \delta \geq 0$

[4]:
```
gr1_x = [0.6, 1.0, 1.5, 2.5, 2.9, 3.0, 4.4, 5.6, 6.0, 7.5, 8.6, 10.6]
 ↪             #x Gruppe 1
gr1_y = [2.5, 0.9, 1.4, 2.5, 3.5, 1.1, 2.3, 1.6, 3.6, 2.4, 3.4, 2.5]
 ↪             #y Gruppe 1

gr2_x = [1.4, 2.6, 2.7, 3.5, 3.6, 4.1, 5.2, 5.5, 5.8, 7.2, 7.6, 9.6, 9.9, 11.1]
 ↪             #x Gruppe 2
gr2_y = [-2.9, -2.5, -3.9, -1.5, -2.4, -3.6, -2.4, -1.4, -3.2, -1.2, -1.9, 0.6,
 ↪-1.3, 1.7]   #y Gruppe 2

target_delta = [0, 0, -1]
 ↪             #Zielfunktion: delta

gr1_mat = np.array([[1.0 for _ in range(3)]for _ in range(len(gr1_x))])
for i in range(len(gr1_x)): gr1_mat[i,0] *= gr1_x[i]
 ↪             #Koeffizientenatrix für NB1
```

```
gr2_mat = np.array([[-1.0, -1.0, 1.0]for _ in range(len(gr2_x))])
for i in range(len(gr2_x)): gr2_mat[i,0] *= gr2_x[i]
 ↪              #Koeffizientenatrix für NB2

d_constraint_ub_mat = np.concatenate((gr1_mat, gr2_mat), axis=0)
 ↪              #Koeffizientenatrix zusammengefügt
d_constraint_ub_vec = np.append(gr1_y, np.multiply(gr2_y, -1))
 ↪              #b-Vektor zusammengefügt

hyperplane = sp.optimize.linprog(c=target_delta, A_ub=d_constraint_ub_mat,
 ↪              #Löse min -delta
                              b_ub=d_constraint_ub_vec,
                              bounds=[(None,None), (None,None), (0,None)])
 ↪              #bounds für a,b auf reelle Werte

print("Abstand delta: ", -hyperplane.fun)
print("Trennende Hyperebene: y =", hyperplane.x[0],"x",hyperplane.x[1])
```

```
Abstand delta:  0.5052631578947364
Trennende Hyperebene: y = 0.4210526315789474 x -2.46842105263158
```

[7]:
```
x = np.linspace(0.6, 11.1)
y = hyperplane.x[0]*x+hyperplane.x[1]
plt.scatter(gr1_x, gr1_y, label='Gruppe 1')
plt.scatter(gr2_x,gr2_y, label='Gruppe 2')
plt.plot(x,y, color='red', label='Trennende Hyperebene')
plt.legend()
plt.show()
```


3
2
1
0
−1
−2
−3
−4
2
4
6
8
10
Gruppe 1
Gruppe 2
Trennende Hyperebene