HW4_Question_8

March 11, 2018

Group: Austin Wang, Joe Higgins, Lawrence Moore

In [1]:
```
# Import necessary modules
import cvxpy as cvx
import numpy as np
```

In [2]:
```
# Helper functions for SDP matrix multiplications
def sum_elem_product(A,B):
    return cvx.sum_entries(cvx.mul_elemwise(A, B))

def col_vec_4elem(a,b,c,d):
    return np.matrix([[a],[b],[c],[d]])
```

0.1 Introduction

In this problem, we will test to see if adding regulative objectives to the SDP sensor network localization problem will allow us to successfully locate sensors that we failed to without the regulative objective.

0.2 Data

Below is the data for the anchors and sensors. In particular, we choose sensors such that one of the sensors is not in the convex hull of the other sensor and the two corresponding anchors. In the previous homeworks, SDP failed to find these sensor locations; we will see if adding the regulative objectives will allow us to be more successful.

In [3]:
```
# Define anchors
anchors = np.matrix([
    [ 1, 0],
    [-1, 0],
    [ 0, 2]
])

# Number of dimensions
n = 2

# Sensor locations
s1 = np.matrix([
```

```
    [ 0, .1],
    [-.15, 1.8]])

# Distances from anchors
d0 = list(map(lambda a: np.linalg.norm(s1[0, :] - a), anchors))
d1 = list(map(lambda a: np.linalg.norm(s1[1, :] - a), anchors))
d_both_sensor = np.linalg.norm(s1[0, :] - s1[1, :])
```

0.3 Problem Construction

In [4]:
```
# Constraints will define the proper structure for Z. The optimal value will be
# located in the top right two elements of Z.
Z = cvx.Semidef(4)

v0 = col_vec_4elem(1,0,0,0)
v1 = col_vec_4elem(0,1,0,0)
v2 = col_vec_4elem(1,1,0,0)

a0 = col_vec_4elem(anchors[0,0], anchors[0,1], -1, 0)
a1 = col_vec_4elem(anchors[1,0], anchors[1,1], -1, 0)

a2 = col_vec_4elem(anchors[1,0], anchors[1,1], 0, -1)
a3 = col_vec_4elem(anchors[2,0], anchors[2,1], 0, -1)

v3 = col_vec_4elem(0, 0, 1, -1)

constraints = [
    sum_elem_product(v0*np.transpose(v0), Z) == 1,
    sum_elem_product(v1*np.transpose(v1), Z) == 1,
    sum_elem_product(v2*np.transpose(v2), Z) == 2,
    sum_elem_product(a0*np.transpose(a0), Z) == cvx.square(d0[0]),
    sum_elem_product(a1*np.transpose(a1), Z) == cvx.square(d0[1]),
    sum_elem_product(a2*np.transpose(a2), Z) == cvx.square(d1[1]),
    sum_elem_product(a3*np.transpose(a3), Z) == cvx.square(d1[2]),
    sum_elem_product(v3*np.transpose(v3), Z) == cvx.square(d_both_sensor)
]
```

0.3.1 Regulative Objective 1: Minimize Trace of Z

In [5]:
```
C = np.identity(4)
objective = cvx.Minimize(sum_elem_product(C, Z))
```

In [6]:
```
prob = cvx.Problem(objective, constraints)
result = prob.solve(solver = 'MOSEK')

# Optimal solution
x_1_star = [Z[0,2].value, Z[1,2].value]
x_2_star = [Z[0,3].value, Z[1,3].value]
```

```
        print('Results for Sensor Inside of Convex Hull of Anchors')
        print('-------------------------------------------------')
        print('True Sensor Locations: {}'.format(s1))
        print('SDP optimal sensor 1 location : {}'.format(x_1_star))
        print('SDP optimal sensor 2 location : {}'.format(x_2_star))
```

```
Results for Sensor Inside of Convex Hull of Anchors
-------------------------------------------------
True Sensor Locations: [[ 0.    0.1 ]
 [-0.15  1.8 ]]
SDP optimal sensor 1 location : [0.0, 0.05681656621615909]
SDP optimal sensor 2 location : [-0.06999867480845734, 1.7599993374042289]
```

0.3.2 Regulative Objective 2: Maximize Trace of Z

```
In [7]: C = np.identity(4)
        objective = cvx.Maximize(sum_elem_product(C, Z))
```

```
In [8]: prob = cvx.Problem(objective, constraints)
        result = prob.solve(solver = 'MOSEK')

        # Optimal solution
        x_1_star = [Z[0,2].value, Z[1,2].value]
        x_2_star = [Z[0,3].value, Z[1,3].value]

        print('Results for Sensor Inside of Convex Hull of Anchors')
        print('-------------------------------------------------')
        print('True Sensor Locations: {}'.format(s1))
        print('SDP optimal sensor 1 location : {}'.format(x_1_star))
        print('SDP optimal sensor 2 location : {}'.format(x_2_star))
```

```
Results for Sensor Inside of Convex Hull of Anchors
-------------------------------------------------
True Sensor Locations: [[ 0.    0.1 ]
 [-0.15  1.8 ]]
SDP optimal sensor 1 location : [0.0, 0.10000003811232931]
SDP optimal sensor 2 location : [-0.15000006351395934, 1.80000003175698]
```

0.3.3 Regulative Objective 3: Minimize Sum of Non-Edge Distance Squares

```
In [9]: c1 = col_vec_4elem(anchors[2,0], anchors[2,1], -1, 0)
        c2 = col_vec_4elem(anchors[0,0], anchors[0,1], 0, -1)
        C = c1*np.transpose(c1) + c2*np.transpose(c2)
        objective = cvx.Minimize(sum_elem_product(C, Z))
```

```
In [10]: prob = cvx.Problem(objective, constraints)
         result = prob.solve(solver = 'MOSEK')

         # Optimal solution
         x_1_star = [Z[0,2].value, Z[1,2].value]
         x_2_star = [Z[0,3].value, Z[1,3].value]

         print('Results for Sensor Inside of Convex Hull of Anchors')
         print('-------------------------------------------------')
         print('True Sensor Locations: {}'.format(s1))
         print('SDP optimal sensor 1 location : {}'.format(x_1_star))
         print('SDP optimal sensor 2 location : {}'.format(x_2_star))
```

```
Results for Sensor Inside of Convex Hull of Anchors
-------------------------------------------------
True Sensor Locations: [[ 0.    0.1 ]
 [-0.15  1.8 ]]
SDP optimal sensor 1 location : [0.0, 0.05719579163857793]
SDP optimal sensor 2 location : [-0.07025636374641686, 1.7601281818732086]
```

0.3.4 Regulative Objective 4: Maximize Sum of Non-Edge Distance Squares

```
In [11]: c1 = col_vec_4elem(anchors[2,0], anchors[2,1], -1, 0)
         c2 = col_vec_4elem(anchors[0,0], anchors[0,1], 0, -1)
         C = c1*np.transpose(c1) + c2*np.transpose(c2)
         objective = cvx.Maximize(sum_elem_product(C, Z))
```

```
In [12]: prob = cvx.Problem(objective, constraints)
         result = prob.solve(solver = 'MOSEK')

         # Optimal solution
         x_1_star = [Z[0,2].value, Z[1,2].value]
         x_2_star = [Z[0,3].value, Z[1,3].value]

         print('Results for Sensor Inside of Convex Hull of Anchors')
         print('-------------------------------------------------')
         print('True Sensor Locations: {}'.format(s1))
         print('SDP optimal sensor 1 location : {}'.format(x_1_star))
         print('SDP optimal sensor 2 location : {}'.format(x_2_star))
```

```
Results for Sensor Inside of Convex Hull of Anchors
-------------------------------------------------
True Sensor Locations: [[ 0.    0.1 ]
 [-0.15  1.8 ]]
SDP optimal sensor 1 location : [0.0, 0.10000006145272888]
SDP optimal sensor 2 location : [-0.1500001069335295, 1.800000053466765]
```

0.4 Conclusion

Our results are as follows:

1. Minimize Trace of Z: FAILURE
2. Maximize Trace of Z: SUCCESS
3. Minimize Sum of Non-Edge Distance Squares: FAILURE
4. Maximize Sum of Non-Edge Distance Squares: SUCCESS

We find that adding a regulative objective that maximizes the trace of Z or a regulative objective that maximizes the sum of the non-edge distance squares allows us to successfully locate the sensors we otherwise could not. The minimizing counterparts to these two objectives, however, led to no improvement.