Problem_Set_1

January 23, 2018

1 Assignment 1, Problem 9 Computation Teamwork

1.1 Group: Austin Wang, Lawrence Moore, Joe Higgins

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

1.2 Introduction

In this notebook, we will be solving the two-dimensional sensor localization problem with one sensor and three anchors. We assume that we know each of the three distances between the sensor and the anchors. Our goal is to use this information to determine the location of the sensor.

We will formulate this problem as both an SOCP relaxation problem and an SDP relaxation problem to analyze how the two methods compare.

1.3 Problem Data

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

# Number of dimensions
n = 2

# Sensor locations
sensor_location_in = [0.44, 0.25] # Sensor inside convex hull
sensor_location_out = [1, 1]      # Sensor outside convex hull

# Distances from anchors
d_in = list(map(lambda a: np.linalg.norm(sensor_location_in - a), anchors))
d_out = list(map(lambda a: np.linalg.norm(sensor_location_out - a), anchors))
```

1.4 Part 1: SOCP Relaxation Problem Experimentation

We formulate the SOCP relaxation problem below for a sensor in the convex hull:

```
In [3]: # Construct the problem.
        x = cvx.Variable(1,n)
        objective = cvx.Minimize(cvx.norm(x - anchors[0]) +
                                 cvx.norm(x - anchors[1]) +
                                 cvx.norm(x - anchors[2]))

        constraints = [cvx.norm(x - anchors[0]) <= d_in[0],
                       cvx.norm(x - anchors[1]) <= d_in[1],
                       cvx.norm(x - anchors[2]) <= d_in[2]]

        prob = cvx.Problem(objective, constraints)
```

Solving the problem yields the following:

```
In [4]: # The optimal objective is returned by prob.solve().
        result = prob.solve(solver = 'MOSEK')

        # The optimal value for x is stored in x.value.
        print('Results for Sensor Inside of Convex Hull of Anchors')
        print('-------------------------------------------------')
        print('True Sensor Location: {}'.format(sensor_location_in))
        print('SOCP optimal sensor location : {}'.format(x.value))
```

```
Results for Sensor Inside of Convex Hull of Anchors
-------------------------------------------------
True Sensor Location: [0.44, 0.25]
SOCP optimal sensor location : [[ 0.43999996  0.25000003]]
```

As one can see, the SOCP relaxation problem solution yields the correct sensor location for a sensor in the convex hull of the anchors.

Now let us see what happens if the sensor is outside of the convex hull:

```
In [5]: # Update constraints for sensor outside of the convex hull
        constraints = [cvx.norm(x - anchors[0]) <= d_out[0],
                       cvx.norm(x - anchors[1]) <= d_out[1],
                       cvx.norm(x - anchors[2]) <= d_out[2]]

        prob = cvx.Problem(objective, constraints)
        result = prob.solve(solver = 'MOSEK')

        print('Results for Sensor Outside of Convex Hull of Anchors')
        print('--------------------------------------------------')
        print('True Sensor Location: {}'.format(sensor_location_out))
        print('SOCP optimal sensor location : {}'.format(x.value))
```

```
Results for Sensor Outside of Convex Hull of Anchors
--------------------------------------------------
True Sensor Location: [1, 1]
SOCP optimal sensor location : [[ 0.2  0.6]]
```

Interestingly enough, the SOCP relaxation problem solution does not yield the correct sensor location for a sensor outside of the convex hull of the anchors. Let us see if the SDP relaxation problem differs.

1.5 Part 2: SDP Relaxation Problem Experimentation

The following functions will be useful in the code that follows:

```
In [6]: def sum_elem_product(A,B):
            return cvx.sum_entries(cvx.mul_elemwise(A, B))

        def col_vec_3elem(a,b,c):
            return np.matrix([[a],[b],[c]])
```

We formulate the SDP relaxation problem below for a sensor in the convex hull:

```
In [7]: # 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(3)

        # Objective does not matter;
        # We are simply concerned with solving feasibility conditions
        objective = cvx.Minimize(0)


        v0 = col_vec_3elem(1,0,0)
        v1 = col_vec_3elem(0,1,0)
        v2 = col_vec_3elem(1,1,0)

        a0 = col_vec_3elem(anchors[0,0],anchors[0,1],-1)
        a1 = col_vec_3elem(anchors[1,0],anchors[1,1],-1)
        a2 = col_vec_3elem(anchors[2,0],anchors[2,1],-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(d_in[0]),
            sum_elem_product(a1*np.transpose(a1), Z) == cvx.square(d_in[1]),
            sum_elem_product(a2*np.transpose(a2), Z) == cvx.square(d_in[2])
        ]
```

Solving the problem and extracting the solution from the top right two elements of Z yields:

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

        # Optimal solution
        x_star = [Z[0,2].value, Z[1,2].value]

        print('Results for Sensor Inside of Convex Hull of Anchors')
        print('-------------------------------------------------')
        print('True Sensor Location: {}'.format(sensor_location_in))
        print('SDP optimal sensor location : {}'.format(x_star))
```

```
Results for Sensor Inside of Convex Hull of Anchors
-------------------------------------------------
True Sensor Location: [0.44, 0.25]
SDP optimal sensor location : [0.43999999999999995, 0.25]
```

Like in the SOCP example, the SDP relaxation problem solution yields the correct location for the sensor located inside of the convex hull of the anchors.

Now we test SDP for the sensor outside of the convex hull:

```
In [9]: # Update constraints for sensor outside of the convex hull
        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(d_out[0]),
            sum_elem_product(a1*np.transpose(a1), Z) == cvx.square(d_out[1]),
            sum_elem_product(a2*np.transpose(a2), Z) == cvx.square(d_out[2])
        ]

        prob = cvx.Problem(objective, constraints)
        result = prob.solve(solver = 'MOSEK')

        # Optimal solution
        x_star = [Z[0,2].value, Z[1,2].value]

        print('Results for Sensor Outside of Convex Hull of Anchors')
        print('-------------------------------------------------')
        print('True Sensor Location: {}'.format(sensor_location_out))
        print('SDP optimal sensor location : {}'.format(x_star))
```

```
Results for Sensor Outside of Convex Hull of Anchors
-------------------------------------------------
True Sensor Location: [1, 1]
SDP optimal sensor location : [1.0000000000000002, 1.0]
```

The SDP relaxation problem solution does yield the correct location for the sensor located outside of the convex hull. Therefore, we conclude that the SDP formulation is preferable to the

SOCP formulation in solving our particular sensor localization problem if we care to generalize the possible locations for the sensor.

As a final test of the SDP formulation, we consider choosing a sensor that is very far outside the convex hull of the anchors:

```
In [10]: sensor_location_out2 = [1000, 1000]
         d_out2 = list(map(lambda a: np.linalg.norm(sensor_location_out2 - a), anchors))

         # Update constraints for sensor very far outside of the convex hull
         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(d_out2[0]),
             sum_elem_product(a1*np.transpose(a1), Z) == cvx.square(d_out2[1]),
             sum_elem_product(a2*np.transpose(a2), Z) == cvx.square(d_out2[2])
         ]

         prob = cvx.Problem(objective, constraints)
         result = prob.solve(solver = 'MOSEK')

         # Optimal solution
         x_star = [Z[0,2].value, Z[1,2].value]

         print('Results for Sensor Outside of Convex Hull of Anchors')
         print('-------------------------------------------------')
         print('True Sensor Location: {}'.format(sensor_location_out2))
         print('SDP optimal sensor location : {}'.format(x_star))
```

```
Results for Sensor Outside of Convex Hull of Anchors
-------------------------------------------------
True Sensor Location: [1000, 1000]
SDP optimal sensor location : [1000.0, 999.9999999999418]
```

As one can see, SDP is even able to find the correct sensor location for a point very far outside of the convex hull of the anchors. Therefore, we conclude that SDP is likely able to find the exact correct solution for every sensor location on the plane.