Optimization Assignment Report 2

Dajin Han

June 2021

1 Concepts

1.1 Zero-Sum Game

Zero-sum game is a situation when the sum of advantages of single winning is same with the sum of disadvantages of single winning in same game. For example, two-player play "rock paper scissors" game. If player wins, player gets 1 score, if player draws, the player gets 0 score and if player loses, player gets -1 score. For every case sum of points two player gained in single game is zero.

$$p_1^{*T} A p_2^* \leq p_1^T A p_2^* \tag{1}$$

$$p_1^{*T} B p_2^* \leq p_1^{*T} B p_2 \tag{2}$$

1.2 Saddle Point

Saddle point is a minmax point which is minimum on one axis and also is maximum on the other axis.

2 Implementation

There is one class, which is optimizer class. Optimizer optimize probability vector for each row and column player.

2.1 Optimizer

Optimizer gets an 2-dimensional matrix and calculate the probability vector for each row and column player. p_r and p_c are probability vector for each player. When optimizer run self.optimize() function, it calculate the probability vector for row and column.

3 Experiments

With those objects, I run the optimizer to get the optimized, converged point about various condition.

In [3]:
```
class Optimizer:
    def __init__(self, matrix):
        self.A = matrix
        self.p_r = np.ones(self.A.shape[0]) / self.A.shape[0]
        self.p_c = np.ones(self.A.shape[1]) / self.A.shape[1]

    def optimize_r(self):
        A_eq = np.ones((1, self.p_r.size))
        b_eq = np.ones(1)
        A_ub = self.A
        b_ub = np.zeros(self.p_c.size)

        # Set input parameters for linprog function
        c = np.zeros((1, self.p_r.size + 1))
        c[0, -1] = -1
        A_eq = np.concatenate((A_eq, np.zeros((1,1))), axis=1)
        b_eq = b_eq
        A_ub = np.concatenate((-A_ub, np.ones((1, self.p_c.size))), axis=0).T
        b_ub = b_ub

        #print(c.shape, A_eq.shape, b_eq.shape, A_ub.shape, b_ub.shape)
        lp_result = linprog(c=c,
                            A_eq=A_eq,
                            b_eq=b_eq,
                            A_ub=A_ub,
                            b_ub=b_ub,
                            bounds=[(0,1)]*self.p_r.size + [(np.min(self.A), np.max(self.A))])
        return lp_result

    def optimize_c(self):
        A_eq = np.ones((1, self.p_c.size))
        b_eq = np.ones(1)
        A_ub = self.A.T
        b_ub = np.zeros(self.p_r.size)

        # Set input parameters for linprog function
        c = np.zeros((1, self.p_c.size + 1))
        c[0, -1] = -1
        A_eq = np.concatenate((A_eq, np.zeros((1,1))), axis=1)
        b_eq = b_eq
        A_ub = np.concatenate((-A_ub, np.ones((1, self.p_r.size))), axis=0).T
        b_ub = b_ub

        #print(c.shape, A_eq.shape, b_eq.shape, A_ub.shape, b_ub.shape)
        lp_result = linprog(c=c,
                            A_eq=A_eq,
                            b_eq=b_eq,
                            A_ub=A_ub,
                            b_ub=b_ub,
                            bounds=[(0,1)]*self.p_c.size + [(np.min(self.A), np.max(self.A))])
        return lp_result

    def optimize(self):
        result_r = self.optimize_r().x
        result_c = self.optimize_c().x
        print("probability of row is ", result_r[:-1])
        print("value is ", result_r[-1])
        print("probability of col is ", result_c[:-1])
        print("value is ", result_c[-1])


    def print_saddle_point(self):
        mins = np.amin(self.A, axis=1)
        maxs = np.amax(self.A, axis=0)
        max_min = np.max(mins)
        min_max = np.min(maxs)

        if max_min != min_max :
            print("No Saddle Point")
        else :
            row_idxs = np.argwhere(self.A == max_min)
            for row_idx in row_idxs:
                print("(%f, %f)" %(row_idx[0]+1, row_idx[1]+1))
```

Figure 1: Optimizer implementation

```
matrix_0 = np.array([[1,2,3],
                     [4,5,6],
                     [7,8,9]])
problem_0 = Optimizer(matrix_0)

problem_0.optimize()
problem_0.print_saddle_point()
```

```
probability of row is  [2.90043314e-12 7.37572829e-12 1.00000000e+00]
value is  6.999999999969044
probability of col is  [2.04386885e-09 2.77558904e-10 1.00000000e+00]
value is  2.9999999964046764
(3.000000, 1.000000)
```

Figure 2: example 0

```
matrix_1 = np.array([[4,3,1,4],
                     [2,5,6,3],
                     [1,0,7,0]])
problem_1 = Optimizer(matrix_1)

problem_1.optimize()
problem_1.print_saddle_point()
```

```
probability of row is  [5.71428572e-01 4.28571430e-01 4.96688225e-11]
value is  3.1428571418520375
probability of col is  [6.66666666e-01 6.80372325e-10 3.33333334e-01 3.54626394e-10]
value is  2.9999999984208547
No Saddle Point
```

Figure 3: example 1

```
matrix_2 = np.array([[0, 5, -2],
                     [-3, 0, 4],
                     [6, -4, 0]])
problem_2 = Optimizer(matrix_2)

problem_2.optimize()
problem_2.print_saddle_point()
```

```
probability of row is  [0.36363636 0.34965035 0.28671329]
value is  0.6713286713212918
probability of col is  [0.30769231 0.29370629 0.3986014 ]
value is  0.6713286713185154
No Saddle Point
```

Figure 4: example 2

```
matrix_3 = np.array([[5,8,3,1,6],
                     [4,2,6,3,5],
                     [2,4,6,4,1],
                     [1,3,2,5,3]])
problem_3 = Optimizer(matrix_3)

problem_3.optimize()
problem_3.print_saddle_point()
```

```
probability of row is  [1.62162162e-01 5.40540541e-01 1.55761033e-10 2.97297297e-01]
value is  3.270270269636707
probability of col is  [1.18721307e-10 1.93277311e-01 1.63865546e-01 4.36974790e-01
 2.05882353e-01]
value is  3.7100840333440637
No Saddle Point
```

Figure 5: example 3