Problem Set 4

Jonathan Tregde
ECON815: Topics in Microeconomics

Fall 2019

Problem 1. Maximum Score Estimator

For this problem set, we used radio station merger data to estimate the parameters of the payoff function of the merger. We began by estimating the function,

$$f_m(b,t) = x_{1bm}y_{1tm} + \alpha x_{2bm}y_{1tm} + \beta distance_{btm} + \epsilon_{btm}$$

After estimating α and β for that equation, we then estimate a payoff function which includes target characteristics

$$f_m(b,t) = \delta x_{1bm}y_{1tm} + \alpha x_{2bm}y_{1tm} + \gamma HHI_{tm} + \beta distance_{btm} + \epsilon_{btm}$$

and we again use maximum score estimation to estimate δ, α, γ, and β for this function.

The MSE looks something like

$$\hat{\beta} = argmaxQ(\beta) = \sum_{y=1}^{Y}\sum_{b=1}^{M_y-1}\sum_{b'=b+1}^{M_y} \mathbb{1}[f(b,t \mid \beta) + f(b^{prime}, t^{prime} \mid \beta) \geq f(b^{prime}, t \mid \beta) + f(b, t^{prime} \mid \beta)]$$

Problem 2. Procedure

First, after reading in our data, we calculated the distances and put them in a new column for the data frame of the observed matches.

```
from geopy import distance as dist
rad['d'] = rad.apply(lambda row: dist.geodesic((row.buyer_lat, row.buyer_long)
                                        , (row.target_lat, row.target_long)),
                                        axis = 1)
```

Next, we split the data by market in order to allow proper calculation of the payoff function. After this, we created a matrix of counterfactuals to be used in the objective function for the maximum score estimator.

```
rad_CF = [cf[B].iloc[b].values.tolist() + cf[T].iloc[t].values.tolist() for cf
                                         in years for b in range(len(cf) - 1)
                                        for t in range(b + 1, len(cf))]
rad_CF = pd.DataFrame(rad_CF, columns = B + T)
```

At this point, we were then ready to create the distance column for the counterfactuals and then write a function to calculate the payoffs as follows:

```
# Create the distance column
rad_CF['d'] = rad_CF.apply(lambda row: dist.geodesic((row.buyer_lat, row.
                                        buyer_long), (row.target_lat, row.
                                        target_long)), axis = 1)

def payoffs(data, i, params):
    '''
    A function to calculate the payoffs of the mergers.

    Args:
        data: the data to be used
        i: used for iterating through the rows
        params: a list of length (2) of initial parameter estimates
    Returns:
        f: the payoff
    '''
    alpha = params[0]
    beta = params[1]

    f = data['num_stations_buyer'].iloc[i] * data['logk_pop'].iloc[i] + alpha
                                        * data['corp_owner_buyer'].iloc[i]
                                        * data['logk_pop'].iloc[i] + beta *
                                         data['d'].iloc[i]


    return(f)
```

Unfortunately, something was wrong with running the payoff function, so I was unable to actually calculate the payoffs to use in the MSE. But I did write up code (or psuedo code) for how I would calculate the objective function to be optimized.

```
def objective(freal7, freal8, fCF7, fCF8):
    '''
    A function to calculate the value of the objective function to be
                                        maximized
```

```
    Args:
        data:

    Returns:
        score: the score
    '''
    score = 0
    for i in [[freal7, fCF7], [freal8, fCF8]]:
        for j in range(len(i[0])):
            for k in range(len(i[0])):
                if i[0][j] + i[0][k] >= i[1][k, j] + i[1][k, (j - 1)]:
                    score = score - 1

    return(score)

results = opt.minimize(objective, Params, method = 'Nelder-Mead', options = {'
                                        maxiter': 5000})
print(results)
```