fictitious play

小川紗里奈

はじめに

- Fictitious play とは
- プログラムコード
- まとめ (今後の課題)

Fictitious play とは

- 戦略形ゲームが $t = 1, 2, ...$ の各期にプレイされるとする
- t 期において各プレイヤーは、他のプレイヤーが 1 期から $t-1$ 期に選択した比率と等しい確率で各純粋戦略を選択すると予測→最適反応戦略を選択
- このような動学モデルを Fictitious play という

Fictitious play とは

- 各プレイヤーを 0、1、戦略を 0、1 とする
- t 期におけるプレイヤー 0
  - プレイヤー 1 は確率 $1 - x_0(t)$ で 0 をとる
  - プレイヤー 1 は確率 $x_0(t)$ で 1 をとる
  - →この信念のもとで期待利得が最大になるような行動をとる
- プレイヤー 1 も同様

Fictitious play とは

- 初期信念 $x_0(0)$ は $[0,1]$ 上の一様分布に従いランダムに決まる
- 各 $t \geq 1$ 時点においてプレイヤー 1 が過去にとった行動を $a_1(0), ..., a_1(t-1)$ とすると

$$x_0(t) = \frac{x_0(0) + a_1(0) + a_1(1) + ... + a_1(t-1)}{t+1}$$

ここで $x_0(t)$ は

$$x_0(t+1) = x_0(t) + \frac{1}{t+2}(a_1(t) - x_0(t))$$

と再帰的に書くことができる．これをプレイヤー 0,1 について考える

Fictitious play とは

- ただし
  - $x_0(0), x_1(0) \sim Uniform[0, 1]$
  - $a_0(t)$ は $x_0(t)$ に対する最適反応、$a_1(t)$ は $x_1(t)$ に対する最適反応
  - 最適反応が複数ある場合は等確率でランダムに選ぶ

Fictitious play とは

- 例として次のような Matching pennies ゲームを考える

$$\begin{bmatrix} 1,-1 & -1,1 \\ -1,1 & 1,-1 \end{bmatrix}$$

プログラムコード I

- 信念形成のシミュレーション

```
import matplotlib.pyplot as plt
import random

t = 1000
x0 = random.uniform(0,1)
x1 = random.uniform(0,1)

x0_values = [x0]
x1_values = [x1]
profit = [(1,-1),(-1,1),(-1,1),(1,-1)]
 # [左上、右上、左下、右下]
```

プログラムコード II

```
for i in range(t):

    if profit[0][0]*(1-x0) + profit[1][0]*x0
     > profit[2][0]*(1-x0) + profit[3][0]*x0:
         a0 = 0
    elif profit[0][0]*(1-x0) + profit[1][0]*x0
     < profit[2][0]*(1-x0) + profit[3][0]*x0:
         a0 = 1
    else:
         a0 = random.choice([0,1])

    if profit[0][1]*(1-x1) + profit[2][1]*x1
     > profit[1][1]*(1-x1) + profit[3][1]*x1:
```

プログラムコード III

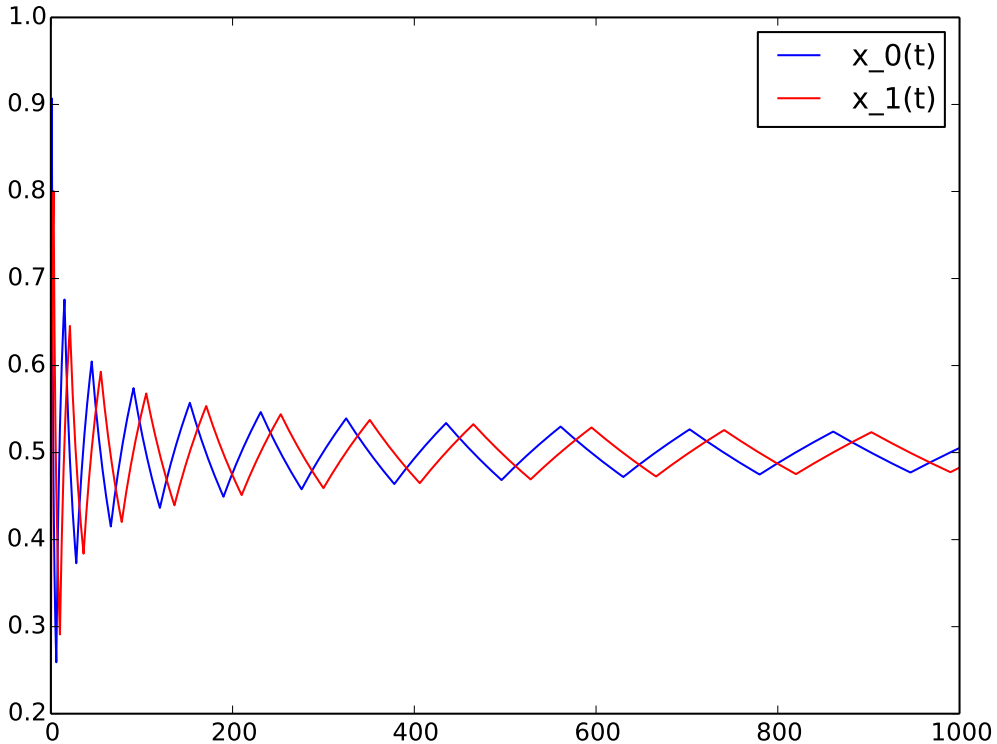
```
            a1 = 0
        elif profit[0][1]*(1-x1) + profit[2][1]*x1
         < profit[1][1]*(1-x1) + profit[3][1]*x1:
            a1 = 1
        else:
            a1 = random.choice([0,1])

        x0 = x0 + (a1 - x0)/(i + 2)
        x1 = x1 + (a0 - x1)/(i + 2)

        x0_values.append(x0)
        x1_values.append(x1)

    plt.plot(x0_values, 'b-', label = 'x_0(t)')
```

プログラムコード IV

```
plt.plot(x1_values, 'r-', label = 'x_1(t)')
plt.legend()
plt.show()
```

- 既存の信念をリストに加える→それをもとに期待利得を計算して戦略決定→その期の相手の行動から、さらに信念を更新を繰り返している

信念のプロット



- プレイヤーの信念は 0.5 付近に収束

今後の課題

- numpy を用いて利得を行列で計算するとよかった
- 各プレイヤーの行動決定の際プレイヤー 0 と 1 で同じ過程を繰り返していたので関数としてまとめるべきだった