Fictitious Play

鈴木花奈子

2014 年 6 月 28 日

はじめに

- Fictitious Play のシュミレーションを行った
- ゲームは Matching Pennies ゲームを行った
- コードは Python で記述した

Fictitious Play とは

- 下のような Matching Pennies ゲームを考える

プレイヤー 0 ＼ 1	0	1
0	1, -1	-1, 1
1	-1, 1	1, -1

- t 時点における各関数を以下のようにおく
  - $x_0(t)$、$x_1(t)$ はプレイヤー 0，1 の信念
  - $a_0(t)$、$a_1(t)$ はプレイヤー 0，1 の行動（0 か 1）
- 信念とは、例えばプレイヤー 0 が

  「プレイヤー 1 は確率 $1 - x_0(t)$ で行動 0 を、$x_0(t)$ で行動 1 をとる」

  と信じることを指す
- 初期信念 $x_0(0)$、$x_1(t)$ は $[0, 1]$ からランダムに選ばれる
- $a_0(t)$、$a_1(t)$ はそれぞれ $x_0(0)$、$x_1(t)$ に対する最適反応

Fictitious Play とは

- $x_0(t)$ は以下の式で求められる

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

- これを整理すると、$x_0(t)$ 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.

- 同様に、$x_1(t)$ も再帰的に表せる

コードの説明

```
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid.axislines import SubplotZero
from random import uniform
import numpy as np
t = 1000
def fict(t):
    pay0 = np.array([[1, -1], [-1, 1]])
    pay1 = np.array([[-1, 1], [1, -1]])
    cur_x0, cur_x1 = uniform(0, 1), uniform(0, 1)
    x0s = []
    x1s = []

    for i in range(t):
        pro0 = np.array([1-cur_x1, cur_x1])
        pro1 = np.array([1-cur_x0, cur_x0])
        exp0 = np.dot(pay0, pro1)
        exp1 = np.dot(pay1, pro0)
```

- fict(t) で Fictitious Play を行う関数を設定
- pay0 と pay1 で利得行列を設定
- x0s と x1s で各 t の $x_0(t)$、$x_1(t)$ を入れるリストを作った

コードの説明

```
if exp0[0] > exp0[1]:
    cur_a0 = 0
elif exp0[0] < exp0[1]:
    cur_a0 = 1
else:
    cur_a0 = random.choice([0, 1])

if exp1[0] > exp1[1]:
    cur_a1 = 0
elif exp1[0] < exp1[1]:
    cur_a1 = 1
else:
    cur_a1 = random.choice([0, 1])
```

- $a_0(t)$、$a_1(t)$ を if 文で決めるようにした

コードの説明

```
x0s.append(cur_x0)
        x1s.append(cur_x1)
        cur_x0 = cur_x0 + (cur_a1 - cur_x0)/(i + 2)
        cur_x1 = cur_x1 + (cur_a0 - cur_x1)/(i + 2)

    return x0s, x1s

def ficthist(t):
    x0_last = []
    for i in range(t):
        x0s, x1s = fict(t)
        x0_last.append(x0s[-1])
    return x0_last

x0s, x1s = fict(t)
```

- cur_x0 と cur_x1 で $x_0(t)$ と $x_1(t)$ を求めた
- ficthist(t) で最終期のプレイヤー 0 の信念の頻度分布を求める関数を設定した

コードの説明

- ```
  fig, ax = plt.subplots()
  ax.plot(x0s, 'r-')
  ax.plot(x1s, 'b-')
  #plt.savefig('fictitious.png')
  #plt.savefig('fictitious.pdf')
  plt.show()

  fig = plt.figure()
  ax = SubplotZero(fig, 111)
  fig.add_subplot(ax)
  ax.hist(ficthist(t))
  #plt.savefig('fictitious_hist.png')
  #plt.savefig('fictitious_hist.pdf')
  plt.show()
  ```
- 最後にそれぞれのグラフを描くようにした
  ```

図

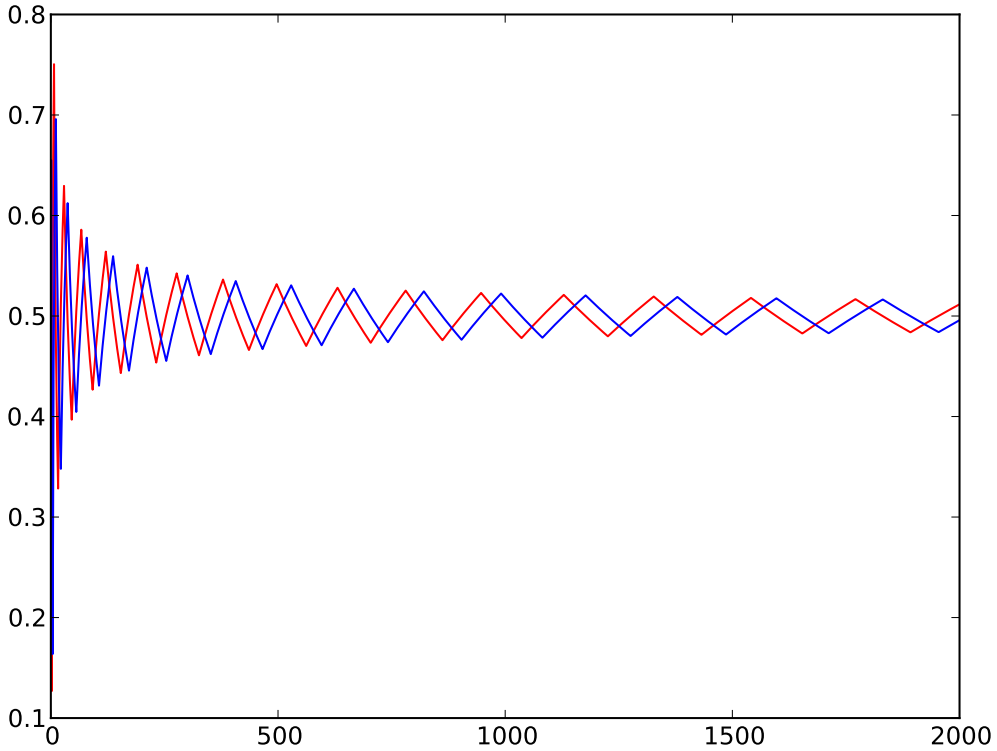

Figure : $x_0(t)$ と $x_1(t)$ の推移

- 回数を重ねるほどどちらも 0.5 に近づく

ヒストグラム

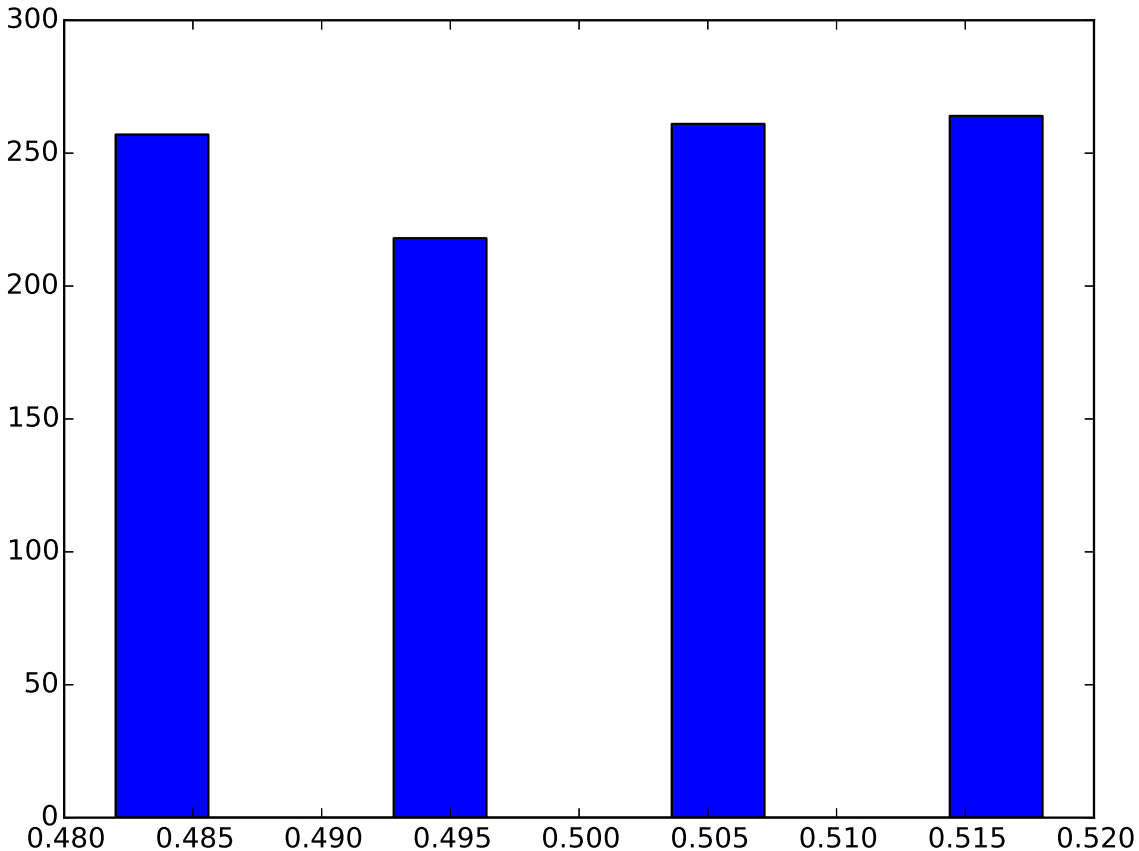

Figure : 1000 回 Matching Pennies ゲームを行う試行を 1000 回繰り返したときの最終期の信念の頻度分布

まとめ

- 今後の課題
  - class を定義するのがまだ自分のなかで理解しきれていない
  - 利得行列をプレイヤー 0 と 1 それぞれ入力しないといけない状態のままになっている
- LaTeX は少しずつですが慣れてきました

  もう少しスピードアップをはかりたいです