Fictitious play

花嶋　陽

はじめに

- Fictitious Play **について説明**
- Matching Pennies **ゲームを例にとった、**Fictitious Play **の**
  シュミレーションプログラムのコードについて説明

Ficititious 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。**
- **「$t + 1$ 期に、他プレイヤーは 1 期から t 期に選択した行動の比率に等しい確率で各純戦略を選択する」と、各プレイヤーは予測する。**
- **各プレイヤーはその予測に対する最適反応戦略を選択する。**
- **このような動学モデルを”Fictitious Play”と言う。**

具体例　２×２ゲーム

- 各期において０か１の行動をとる、プレイヤー０とプレイヤー１を用意する。
- t 期において、プレイヤー０は
  - プレイヤー１は確率 $1 - x_0(t)$ で行動０をとる
  - プレイヤー１は確率 $x_0(t)$ で行動１をとる

  と予測するとする。（プレイヤー１も同様）
- この下で、各プレイヤーは期待利得が最大になるように行動を決定する。（期待利得が等しい場合は等確率にどちらかを選ぶ）
- また、この $x_0(t)$ を「t 時点における、プレイヤー 0 の、プレイヤー 1 の行動に関する信念 (belief)」と呼ぶことにする。

具体例　２×２ゲーム

- **プレイヤー０の信念 $x_0(t)$ は次のように定まる。**
  - **初期信念 $x_0(0)$ は [0,1] 上の一様分布にしたがってランダムに与えられるとする。**
  - **各 $t \geq 1$ 期において、プレイヤー１が過去とった行動を $a_1(0), ..., a_1(t-1)$ とすると、信念 $x_0(t)$ は、**

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

  で与えられる。
  - **これは、**

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

  と再帰的に書くことができる。

具体例　２×２ゲーム

- つまりプレイヤー０とプレイヤー１を合わせて、

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

$$x_1(t+1) = x_1(t) + \frac{1}{t+2}(a_0(t) - x_1(t))$$

という連立１階漸化式を考えることになる。

- ただし、
  - $a_0(t)$ 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ときは等確率でランダムに選ぶ

である。

コードの説明

- Matching Pennies **ゲームのシュミレーション**
- Matching Pennies **ゲームの利得表は以下の通り**

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

ただし、行プレイヤーをプレイヤー０、列プレイヤーをプレイヤー１とし、
各プレイヤーの行動を行動０、行動１と呼ぶ。

コードの説明

- **コードの内容**

```
# -*- coding: utf-8 -*-
from __future__ import division
import matplotlib.pyplot as plt
from random import uniform, choice
import numpy as np

def fictplay(t):
    x_0_t = [uniform(0, 1)]
    x_1_t = [uniform(0, 1)]
#player0 の利得
    gain_0 = np.array([[1, -1], [-1, 1]])
#player1 の利得
     gain_1 = np.array([[-1, 1], [1, -1]])
```

コードの説明

```
for i in range(t):
#probability: player1 choice 0 or 1
     pro_1 = np.array([1-x_0_t[i], x_0_t[i]])

#expected gain = gain ×   probability

      exp_gain_0 = np.dot(gain_0, pro_1)

#player0 の最適反応
       if exp_gain_0[0] > exp_gain_0[1]:
          a_0_i = 0
      elif exp_gain_0[0] == exp_gain_0[1]:
          a_0_i = choice([0, 1])
      else:
          a_0_i = 1
```

コードの説明

```
        pro_0 = np.array([1-x_1_t[i], x_1_t[i]])
        exp_gain_1 = np.dot(gain_1, pro_0)

    #player1 の最適反応
        if exp_gain_1[0] > exp_gain_1[1]:
            a_1_i = 0
        elif exp_gain_1[0] == exp_gain_1[1]:
            a_1_i = choice([0, 1])
        else:
            a_1_i = 1

    #i+1 期の信念を計算
        x_0_i1 = x_0_t[i] + (a_1_i - x_0_t[i]) / (i+2)
        x_1_i1 = x_1_t[i] + (a_0_i - x_1_t[i]) / (i+2)

        x_0_t.append(x_0_i1)
        x_1_t.append(x_1_i1)
```

コードの説明

```
x_0_t, x_1_t = fictplay(1000)

plt.plot(x_0_t, 'r-', label="x_0(t)")
plt.plot(x_1_t, 'b-', label="x_1(t)")
plt.legend()
#plt.savefig('matchingpennies_plot.pdf')
plt.show()

#x_0_t = []
#for i in range(100):
#    x_0, x_1 = fictplay(100)
#    x_0_t.append(x_0[-1])
#plt.hist(x_0_t, label="x_0(t)")
#plt.legend()
#plt.savefig('matchingpennies_plot1.pdf')
#plt.show()
```

図 1

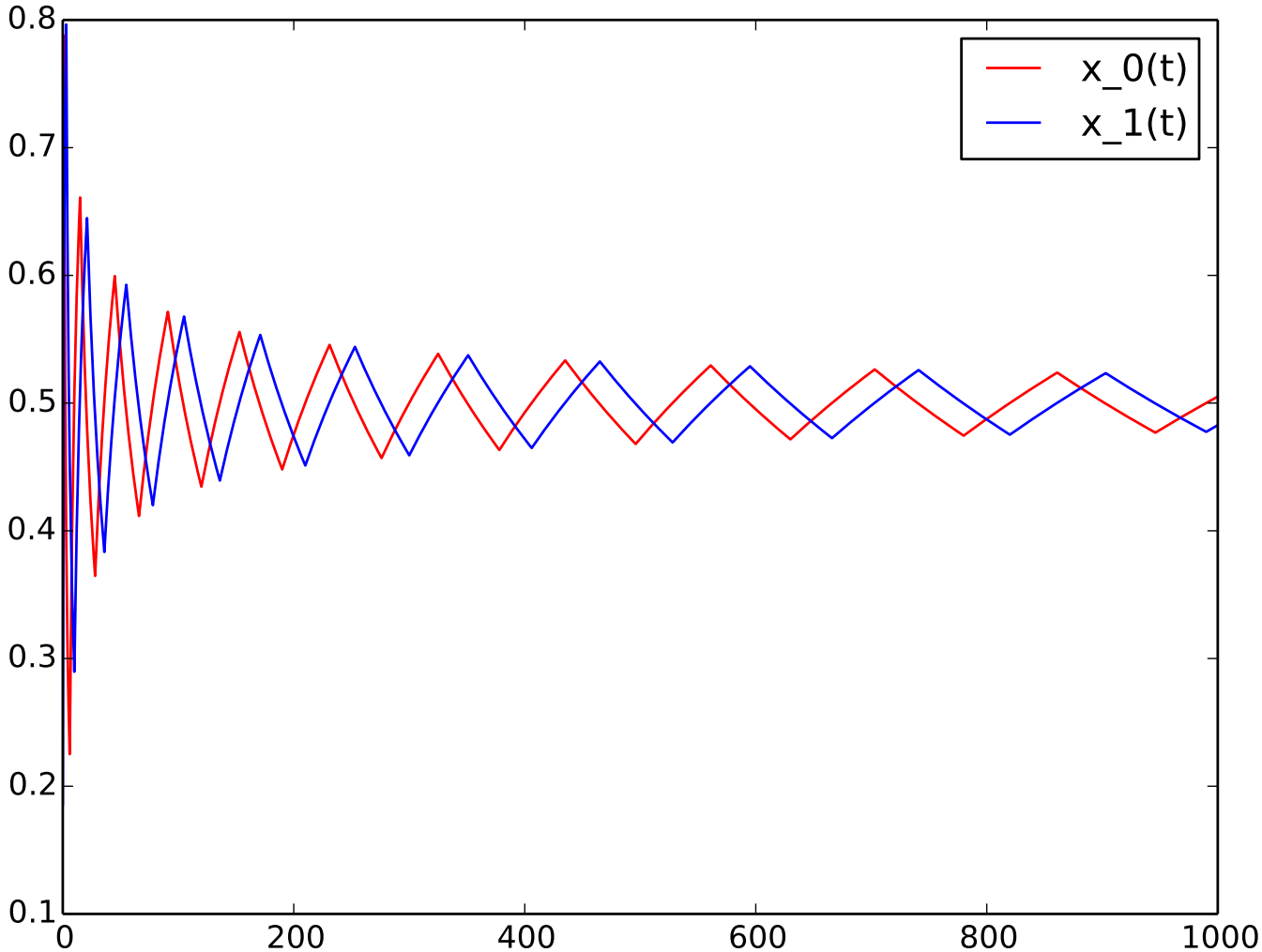

Figure: t=1000 期までの両プレイヤーの信念の推移

図 2

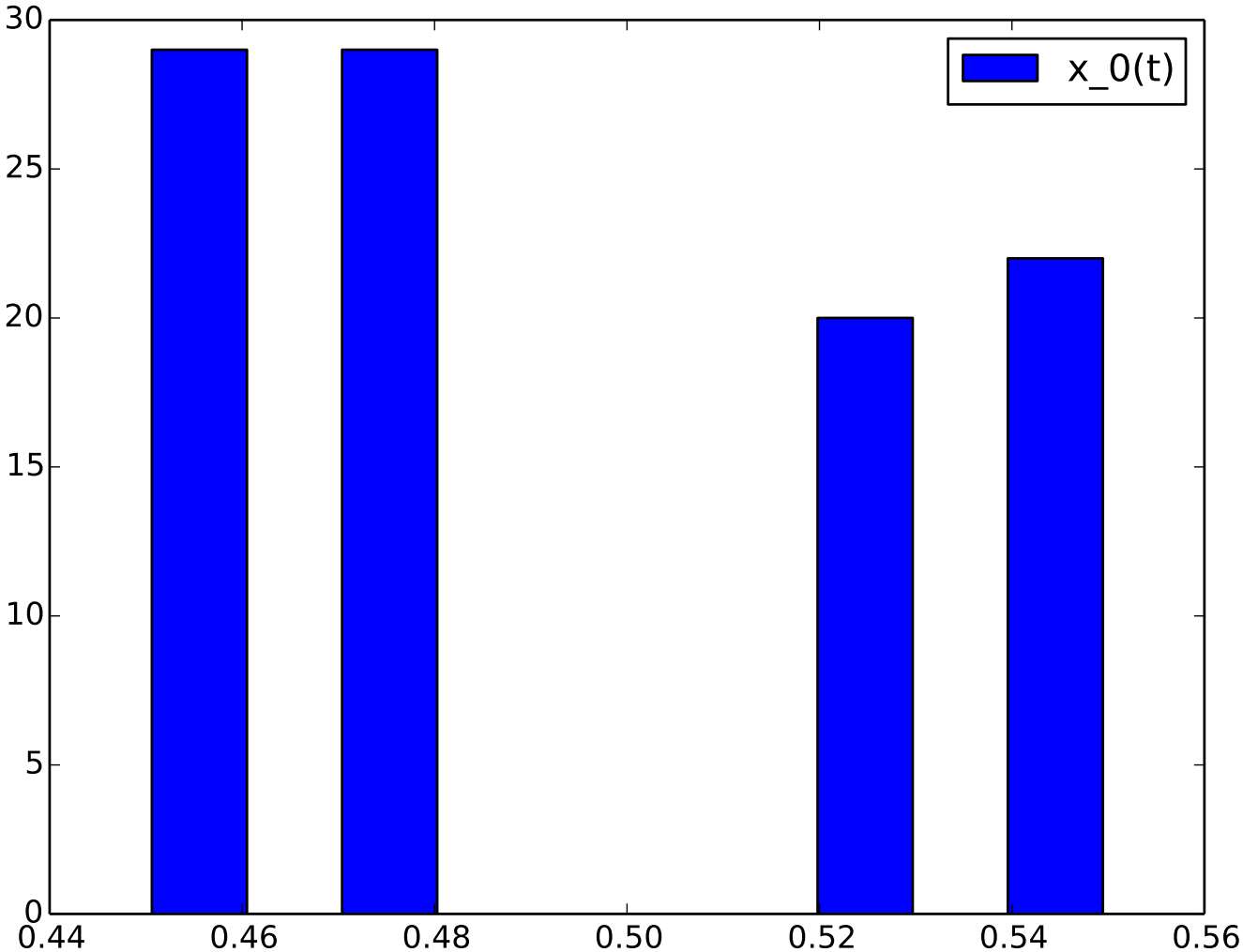

Figure: 100 回シュミレーションした時の $x_0(t)$ の値

コードの説明　まとめ

- 工夫した点
  - 期待利得を行列計算によって求めるようにした。
- 今後の課題
  - プレイヤー０と１の最適反応を導くコードが重複しているので、うまく関数化したい。