Fictitious play

Yusuke Murakami

28 Jun. 2014

Fictitous Play とは

- Ficititious play の前提
  - プレーヤーは 2 人.

    それぞれのプレーヤーをプレーヤー 0、プレーヤー 1 とする.
  - 各プレーヤーには行動が m 種類ある.

    （プレーヤーごとに行動の種類数に差があっても良い）.

    それぞれの行動を行動 0、行動 1、$\cdots$、行動 $m-1$ とする.
  - 各プレーヤーにとって、その行動をとった時の利得が予め定まっている. それは利得行列の形で表される.
  - 期間は 0 期から T 期までの繰り返しゲーム.
  - 各プレーヤーは、後に説明する「信念」を元に、その期に利得が最大になるような行動を選択する. 利得が最大になる行動が複数ある場合はその中から等確率でランダムに選択する.
  - 将来利得を見据えた行動は取らない.

Fictitous Play とは

- **記号**
  - t **期にプレイヤー** i **が選択した行動は** $a_i(t)$ **で表す.**
  - t **期にプレイヤー** i **が持っている「信念」は** $x_i(t)$ **で表す.**
- **信念**
  - t **期のプレーヤー** 0 **が持つ信念は、それまでの期間に相手（プレーヤー** 1**）が取った行動の「平均」で表される. すなわち、**

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

    となる.
  - **プレーヤー** 1 **も同様に定式化する.**
  - **初期の行動時には信念がないので、** $x_0(0)$ **および** $x_1(0)$ **はランダムに行動する.**

今回の設定と漸化式

- **今回はコードの都合で、いずれの行動をとっても利得が同じ場合は、ランダムではなく若い数の行動をとることにした.**
- **また、今回はシミュレーションを面白くするために、信念を以下のようにした.**

$$x_0(t) = \frac{x_0(0) + a_1(0) + \cdots + a_1(t-1)}{t+1} \tag{2}$$

$$x_1(t) = \frac{x_1(0) + a_0(0) + \cdots + a_0(t-1)}{t+1} \tag{3}$$

- **(2) と (3) から、以下のような連立 1 階漸化式を考えることができる.**

$$x_0(t+1) = x_0(t) + \frac{1}{t+2}(a_1(t) - x_0(t)) \tag{4}$$

$$x_1(t+1) = x_1(t) + \frac{1}{t+2}(a_0(t) - x_1(t)) \tag{5}$$

コードの説明

- **初期読み込みと初期変数の設定**

```
import matplotlib.pyplot as plt
import numpy as np
from random import uniform
fig, ax = plt.subplots()

T = 1000
s = ((1,-1),(-1,1),
     (-1,1),(1,-1))
```

`T` **が期間、**`s` **が利得行列（この場合は** Matching Pennies **ゲーム）**.

コードの説明

- **諸関数の設定**

```
def ss(a, s):
    return ((s[0][a], s[1][a]),
            (s[2][a], s[3][a]))

def res(x, s):
    vec = (1-x, x)
    a = np.dot(s, vec)
    return a.argmax()
```

ss はあるプレイヤーの利得行列を取り出す関数、res は最適反応を返す関数. argmax() を使っているので利得が同じ場合は若い数の行動を選ぶ.

コードの説明

- **初期値の設定と漸化式の計算**

```
x0 = [uniform(0, 1)]
x1 = [uniform(0, 1)]

s0 = np.array(ss(0, s))
s1 = np.array(ss(1, s))

for i in range(T-1):
    x0.append(x0[i]+(res(x1[i], s1)-x0[i])/(i+2))
    x1.append(x1[i]+(res(x0[i], s0)-x1[i])/(i+2))
```

x0 と x1 に値を加えていく．s0 と s1 はそれぞれの利得を関数 s で取り出している．

コードの説明

- **グラフ化して保存、表示**

```
ax.plot(x0, 'r-')
ax.plot(x1, 'b-')
plt.savefig('matchingpennies_plot.pdf',
   bbox_inches='tight', pad_inches=0)
plt.show()
```

グラフを書いて保存し、最後に表示する.

図

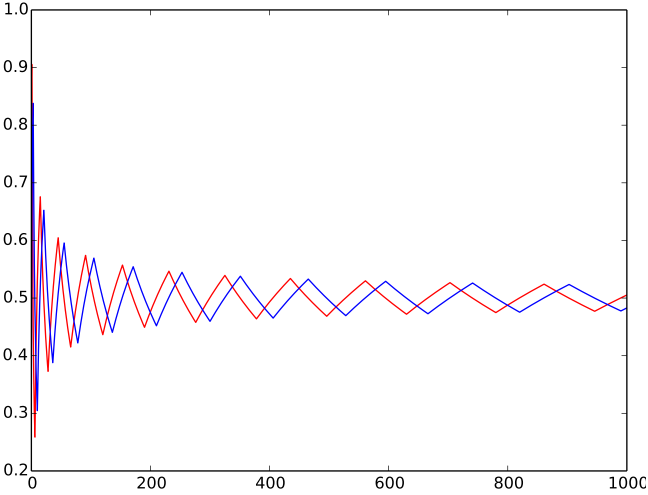

Figure: matching pennies の場合の信念の推移

matching pennies の考察

- **最初は大きく信念が揺らぐが、時間を経るにつれて徐々に収束して行く様子が分かる**.
- **相手の行動を追いかけるような挙動を見せている**.
- **収束先はどちらも** 0.5 **のように見える**[1]. **これは行動** 0 **と行動** 1 **の混合戦略となり、**Nash **均衡に一致する**.

[1] より詳しく見るには、このゲームを何度も繰り返し、$T = 1000$ での頻度をヒストグラム化すると良い

今後検討したいこと

- $T \to \infty$ で行き着く先は必ず混合戦略も含めた Nash 均衡の 1 つになるのか？
  - そもそも行動列は必ず収束するのか？
  - (1) 式と (2) 式の間に差はないか？
- $T \to \infty$ で行き着く先が複数あるとき、その頻度に差はあるのか？
- 頻度に差があったとすると、それにはどんな意味があるのか？
- プレーヤーを増やすとどうなるか？
- 利得行列を確率変数にするとどうなるか？
- 利得を考えるときに「信念」だけでなく「予想される将来利得（の割り引いたもの）」も含めるとどうなるか？