2017/5/19 Pythonで体験するベイズ推論輪読会LT

簡単な問題をPyMCで

@currypurin

自己紹介

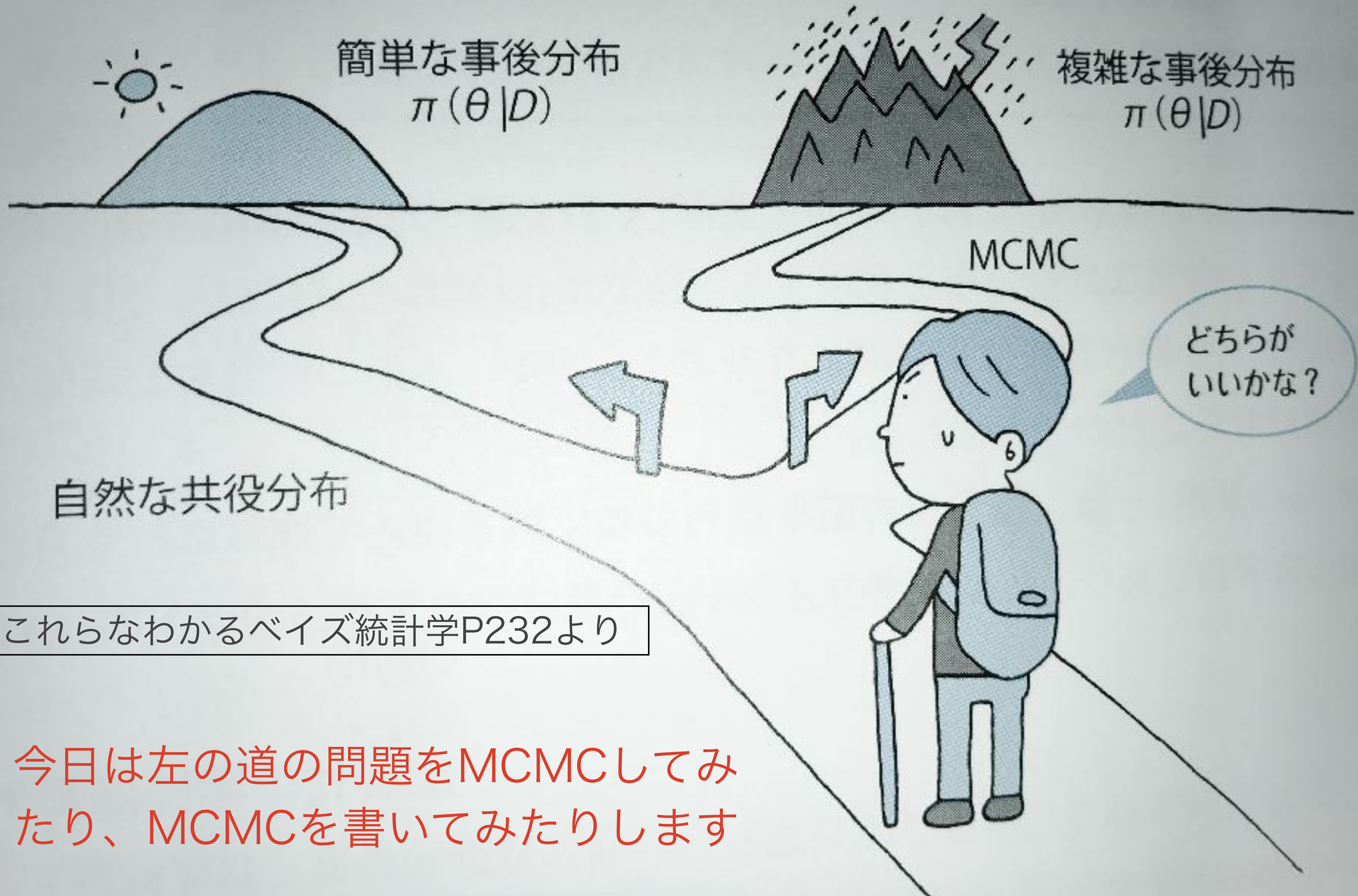
- 半年くらい前から、競馬予測のために統計を勉強しています。普段は事務職やっている。

- PyMC使いたいけど、なかなかわからない。

LTの目的

- 簡単な問題にPyMCを使ってみて、慣れてみたい。
- MCMCをPythonで書いてみて、雰囲気を分かりたかった。
- あわよくば、この使い方で良いのか教えて欲しい。

例のやつ



これらなわかるベイズ統計学P232より

今日は左の道の問題をMCMCしてみたり、MCMCを書いてみたりします

LTの内容

簡単な問題をPyMCで

MCMCをPythonで書いてみる

まとめ

【問題】コイン投げ

- 表の出る確率が θ である１枚のコインがある。このコインを４回投げたとき１回目→表、２回目→表、３回目→表、４回目→裏と出たとする。このとき「表のでる確率 θ 」の確率分布を求めよ。

・第１回目のLTや5/8の輪読会でやった問題。

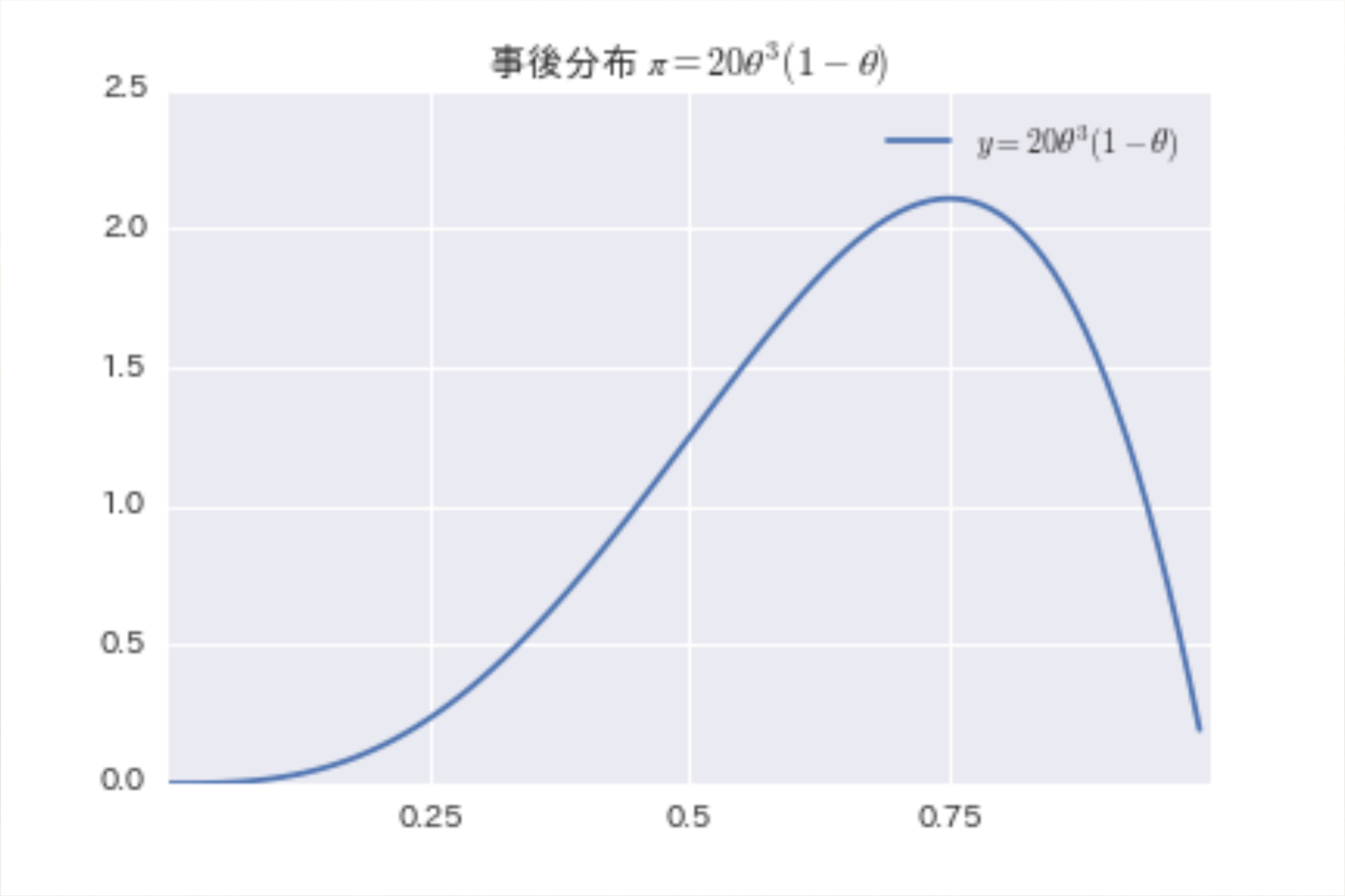
（参考）計算の仕方

$$\pi(\theta|\text{表表表裏}) = kf(\text{表表表裏}|\theta)\pi_0(\theta)$$

- 尤度　　　：f(表表表裏| θ)= θ ^3 ×(1- θ)
- 事前分布： $\pi_0(\theta)=1$
- π (θ |表表表裏)=k× θ ^3×(1- θ)
- k　　　　：k=20　（kは積分で求める）

$$\pi(\theta|\text{表表表裏}) = 20\theta^3(1-\theta)$$

答え



尤度は$\theta^3(1-\theta)$

【問題】コイン投げ

❖ 表の出る確率が θ である１枚のコインがある。このコインを４回投げたとき１回目→表、２回目→表、３回目→表、４回目→裏と出たとする。このとき「表のでる確率 θ」の確率分布を求めよ。

→ この問題をPyMCでやってみる

こんな感じで大丈夫…？

```python
# θは0から1の一様分布
theta = pm.Uniform("theta",lower=0,upper=1)

# 観測データを作成
occurrences = np.array([1,1,1,0])

# ベルヌーイ分布に従う分布を観測済みにする
obs = pm.Bernoulli("obs",theta,value=occurrences,observed=True)

# MCMCクラスにモデルを入れる
mcmc = pm.MCMC([theta,obs])
```

```python
# サンプリング
mcmc.sample(150000,10000)
```

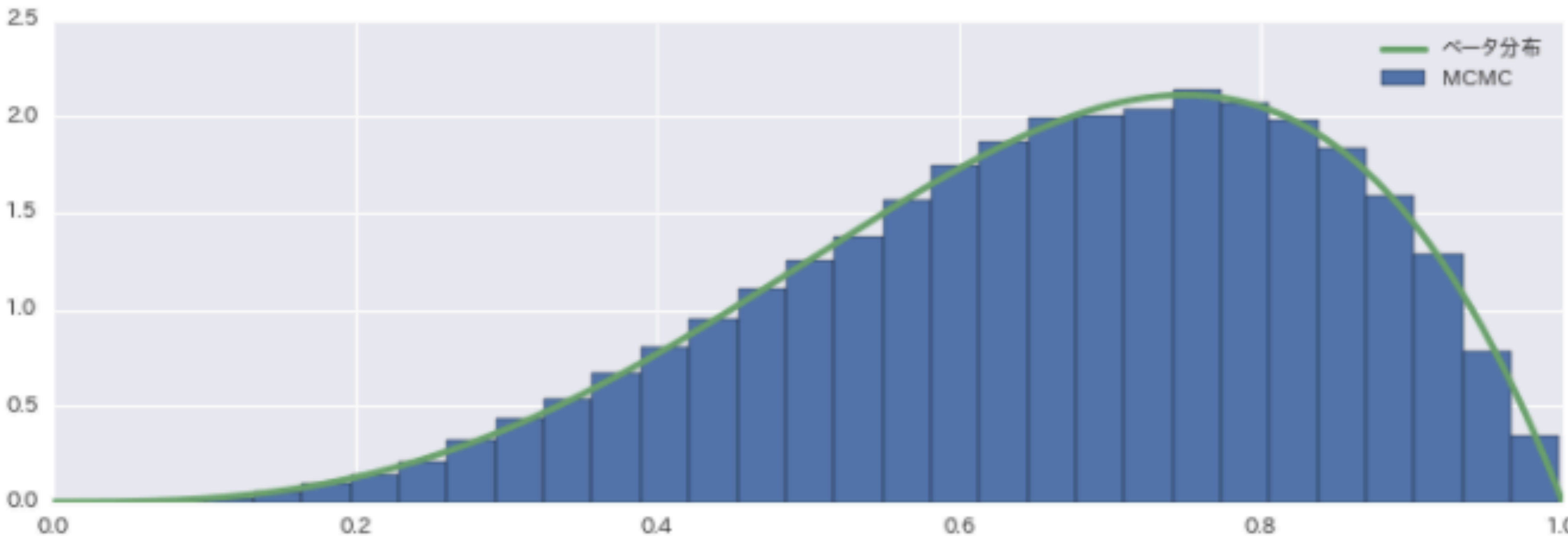
```
[-----------------100%-----------------] 150000 of 150000 complete in 13.6 sec
```

5行で書ける！

可視化

```
plt.hist(mcmc.trace("theta")[:],bins=30,normed=True,label="MCMC")
x = np.linspace(0, 1, 1000)
y = beta.pdf(x, 4, 2)
plt.plot(x, y, linewidth=3,label="ベータ分布")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x11575a278>
```



MCMCのなかを覗いてみる

```
mcmc.trace("theta")[1:1000]
```

```
array([ 0.75936177, 0.75936177, 0.64827568, 0.64827568, 0.64827568,
        0.64827568, 0.64827568, 0.64827568, 0.61553543, 0.61553543,
        0.61553543, 0.61553543, 0.70542229, 0.82587208, 0.82587208,
        0.82587208, 0.82587208, 0.82587208, 0.82587208, 0.78221299,
        0.41386153, 0.87944492, 0.87944492, 0.70696191, 0.70696191,
        0.70696191, 0.70696191, 0.70696191, 0.70696191, 0.65950599,
        0.65950599, 0.57928671, 0.57472891, 0.68084205, 0.68084205,
        0.68084205, 0.68084205, 0.68084205, 0.68084205, 0.68084205,
        0.60698461, 0.60698461, 0.45533072, 0.45533072, 0.54054086,
        0.36615812, 0.36615812, 0.7753729 , 0.7753729 , 0.7753729 ,
        0.7753729 , 0.7753729 , 0.7753729 , 0.7753729 , 0.79496773,
        0.79496773, 0.79496773, 0.79496773, 0.79496773, 0.6155069 ,
        0.6155069 , 0.6155069 , 0.6155069 , 0.6155069 , 0.6155069 ,
        0.6155069 , 0.6155069 , 0.6155069 , 0.6155069 , 0.6155069 ,
        0.6155069 , 0.6155069 , 0.8686471 , 0.8686471 , 0.8686471 ,
        0.8686471 , 0.8686471 , 0.8686471 , 0.94162053, 0.94162053,
        0.94162053, 0.94162053, 0.94162053, 0.93961123, 0.86400752,
```

同じ数字が多い→メトロポリス法？

メトロポリス法

1. パラメータθの初期値を選ぶ
2. θを増やすか減らすかをランダムに決める（新しく選んだθをθ_{new}とする）
3. θ_{new}において尤度が大きくなる（あてはまりがよくなる）ならθの値をθ_{new}に更新する
4. θ_{new}で尤度が小さくなる（あてはまりが悪くなる）場合であっても、確率rでθの値を θ_{new}に変更する

   - このときに悪くなる方向にθを変化させる確率rは尤度比 $r=\dfrac{L(\theta_{new})}{L(\theta)}$ に等しいと設定する。

- データ解析のための統計モデリング入門176頁を参考に記載

- PyMCは、モデルが離散変数の場合には、メトロポリス法でサンプリングを行う初期設定のようです。
- メトロポリス法を自力で書いてみる。

LTの内容

簡単な問題をPyMCで

MCMCをPythonで書いてみる

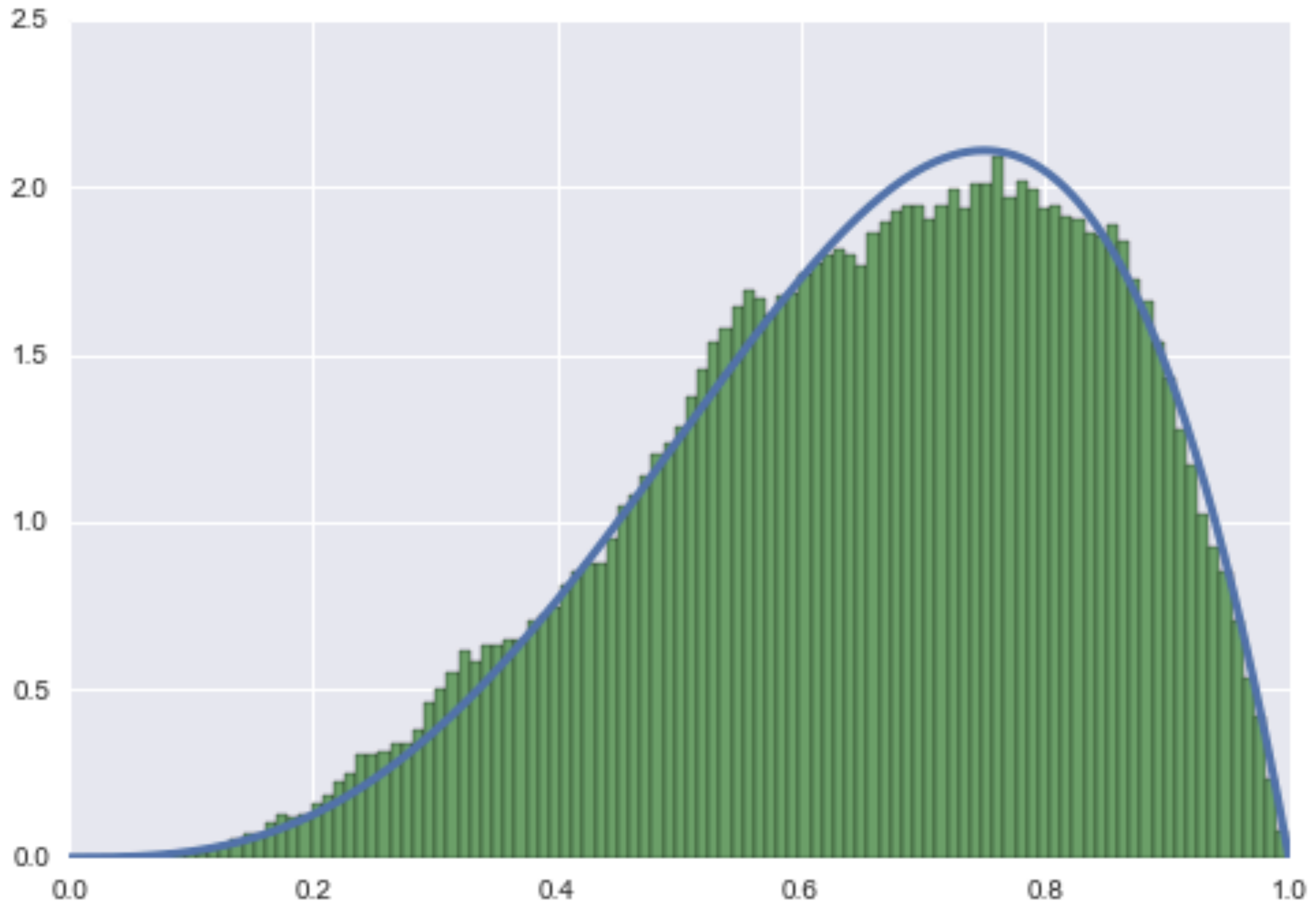
まとめ

メトロポリス法を実装してみる

```
# 尤度を計算する関数を定義
def likelihood(theta):
    if 0 < theta < 1:
        return (theta**3)*(1-theta)
    else:
        return 0

theta = np.random.random()
theta_list = []
burnin = 10000
D = np.array([1, 1, 1, 0])
l = likelihood(theta)

# Start MCMC
for i in range(300000):
    theta_new = np.random.normal(theta, 10 ** -2) # 平均θ,分散1/100の正規分布からひとつチョイス
    l_new = likelihood(theta_new)  #   theta_newでの尤度を計算

# theta_newを採用するか否か
    if l_new/l > np.random.random(): #   np.random.random()は0から1の乱数
        theta = theta_new
        l = l_new
    theta_list.append(theta)
theta_list = np.array(theta_list)
theta_list = theta_list[burnin:]
```

可視化

```
x = np.linspace(0, 1, 1000)
y = beta.pdf(x, 4, 2)
plt.plot(x, y, linewidth=3)

plt.hist(theta_list, normed=True, bins=100)
plt.show()
```



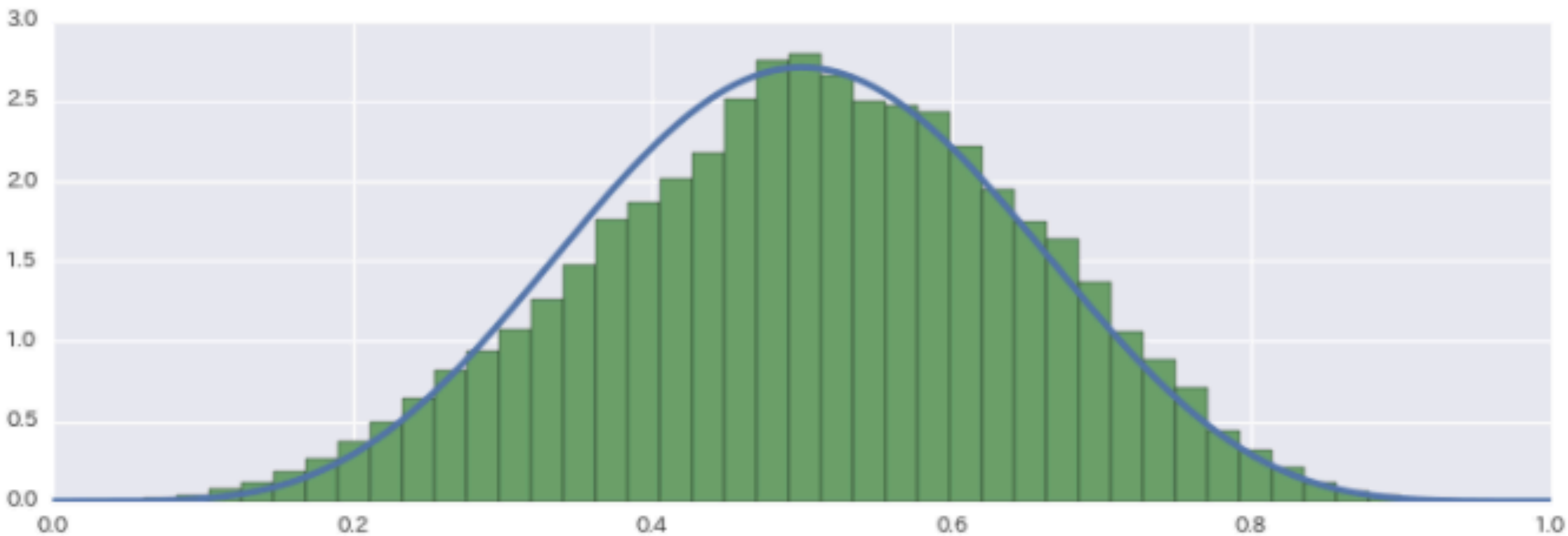
(参考)任意の回数のコイン投げの場合

```
# 対数尤度を計算する関数を定義
def logLikelihood(theta,D):
    if 0 < theta < 1:
        return np.sum(D * np.log(theta) + (1 - D) * np.log(1 - theta))
    else:
        return -np.inf
```

```
theta = np.random.random()
theta_list = []
burnin = 10000
D = np.array([1, 1, 1, 0,1,0,0,1,0,0])
N = D.size
l = logLikelihood(theta,D)

# Start MCMC
for i in range(200000):
    theta_new = np.random.normal(theta, 10 ** -2) # 平均theta,分散1/100の正規分布からひとつ
    l_new = logLikelihood(theta_new,D)  #   l_

    # theta_newを採用するか否か
    if min(1, np.exp(l_new-l)) > np.random.random():

        theta = theta_new
        l = l_new
    theta_list.append(theta)
theta_list = np.array(theta_list)
theta_list = theta_list[burnin:]
```

(参考)可視化

```
x = np.linspace(0, 1, 1000)
y = beta.pdf(x, sum(D) + 1, N - sum(D) + 1)
plt.plot(x, y, linewidth=3)

plt.hist(theta_list, normed=True, bins=40)
plt.show()
```



まとめ

- 一応、コイン投げの例でメトロポリス法の実装できました。

参考にした本

- データ解析のための統計モデリング入門
- 基礎からのベイズ統計学

課題・わからなかったこと

- パラメータθを連続として扱っている場合には、θnewの候補の探し方に工夫が必要になるらしい。
- メトロポリス法で、どうして事後分布をサンプリングできるのかわからなかった。
- 本当はもっと色々な問題をやりたかった。