**『Pythonによるベイズ統計モデリング』メモ**

【事前にやっておくこと】

“BAP\_Chapter2.py”を起動してみたが、うまく動作しない。どうも、PyMc3がうまくインストールされていないらしい。そこで、いろいろネットで調べて、次のようにやって、再度、PyMc3をインストールし、bs4、seaborn、mingwなど足らないモジュールをインストールしたら、うまくいった。

C:\Users\yokan >conda create -c msys2 -c conda-forge -n mypm3env python=3.8 mkl-service libpython m2w64-toolchain scipy matplotlib pandas

C:\Users\yokan>pip install pymc3

C:\Users\yokan>pip install bs4

C:\Users\yokan>pip install seaborn

C:\Users\yokan>pip install mingw

プログラムによっては、デスクトップ上にフォルダー“\_pycache\_”を作成し、その中に“hpd.cpython-311.pyc”と“plot\_post.cpython-311.pyc”を格納しておく必要がある。そのために、次の2つのプログラム“hpd.py”、“plot\_post.py”を実行する。さらに、実行する前にanacondaプロンプトをmypm3envの環境にしておく。

C:\Users\yokan\desktop>python hpd.py

C:\Users\yokan\desktop>python plot\_post.py

C:\Users\yokan\desktop>conda activate mypm3env

【「Pythonによるベイズ統計学入門」2023年8月1日改定】

PyMCの仕様変更に伴う修正（ArviZの必須化）

バージョン3.11より，記述統計や作図に関連する関数はPyMCからArviZに移された．そのためArviZを“import arviz as az”と読み込み，ArviZの関数を呼び出す必要が生じる．

・pm.traceplot()の代わりにaz.plot\_trace()を，pm.plot\_posterior()の代わりにaz.plot\_posterior()を使わなければならない．

・さらに，pm.traceplot()（つまりaz.plot\_trace()）はオプションaxを受け付けなくなった．したがって，最初のplt.subplots()の部分を削除し，続くpm.traceplot()内のax=ax1も削除しなければならない．

・また，pm.plot\_posterior()（つまりaz.plot\_posterior()）もオプションkde\_plotを無視するようになった．デフォルトでは滑らかなカーネル推定が描かれるが(kind='kde')，ヒストグラムを描く際には引数にkind='hist'を指定しなければならない．

pm.mc\_error()をaz.mcse()へ

pm.hpd()をaz.hdi()へ

pm.gelman\_rubin()をaz.rhat()へ

【それでも起こるエラーに対する対処】

・TypeError: function() got an unexpected keyword argument 'njobs'

に対しては

Njobs　⇒　chains

・TypeError: kdeplot() takes from 0 to 1 positional arguments but 2 were given

に対しては

sns.kdeplot(trace\_red['beta'][:,0], trace\_red['beta'][:,1])

⇓

x1,y1 = (trace\_red['beta'][:,0], trace\_red['beta'][:,1])

sns.kdeplot(x=x1,y=y1)

・AttributeError: module 'pymc3' has no attribute 'sample\_ppc'

に対しては、

ppc = pm.sample\_ppc(trace\_n, samples=1000, model=model)

⇓

ppc = pm.sample\_posterior\_predictive(trace\_n, samples=1000, model=model)

・ AttributeError: module 'pymc3' has no attribute 'hpd'

に対しては、

sig0 = pm.hpd(ppc['y\_pred'], alpha=0.5)[idx]

⇓

import arviz as az

sig0 = az.hdi(ppc['y\_pred'], hdi\_prob=0.5)[idx]

・RuntimeError:

An attempt has been made to start a new process before the

current process has finished its bootstrapping phase.

に対しては、次の１行を挿入する

if \_\_name\_\_ == '\_\_main\_\_':

【第2章　確率プログラミングーーPyMC3入門】

●非マルコフ的方法

―グリッドコンピューティング

格子の各点における尤度と事前分布を掛け合わせて求める（例：コイン投げ問題）

―2次近似（ラプラス法）

正規分布によって事後分布を近似する

―変分法

単純な分布によって事後分布を近似する（自動微分変分推論（ADVI））など）

●マルコフ的方法（マルコフ連鎖モンテカルロ法（MCMC法））

モンテカルロ法：乱数を使用する（例：円周率の計算）

マルコフ連鎖：次の状態への遷移確率が現在の状態のみに依存する

（例：ランダムウォーク、可逆過程）

―メトロポリス＝ヘイスティングス（MH）

目標とする確率分布からのサンプルとして適当な候補を提案する分布を用意し、その候補を確率的に受容するか棄却するかを決めることで、マルコフ連鎖を構成する。提案分布と目標分布の比率に応じて、受容率を計算し、その受容率に基づいて候補を採択するか否かを決める。

―ハミルトニアンモンテカルロ（HMC）/ノーＵターンサンプラー（NUTS）

サンプリングしたいパラメータを位置（高さ）に、乱数を運動量に対応させて、ハミルトニアンを定義する。そして、ハミルトニアンの一定性を保ちながら、運動量と位置を交互に更新することで、パラメータの候補を選ぶ。

np.linspace()は等差数列を生成する。第一引数に最初の値、第二引数に最後の値、第三引数に要素数を指定する。それらに応じた間隔が自動的に算出される。

print(np**.**linspace(0, 10, 3))

# [ 0. 5. 10.]

print(np**.**linspace(0, 10, 4))

# [ 0. 3.33333333 6.66666667 10. ]

print(np**.**linspace(0, 10, 5))

# [ 0. 2.5 5. 7.5 10. ]

np.repeat(a, n) はスカラーや配列 a の各要素を n 回 繰り返す 配列を生成します。

# 1を5個並べる

a = np.repeat(1, 5)

print(a)

# [1 1 1 1 1]

#

# 1と2を3回繰り返す

b = np.repeat([1, 2], 3)

print(b)

# [1 1 1 2 2 2]

二項分布：投げると確率 0.4 で表が出る硬貨を10回投げて表が3回出る確率は0.214990848 で、この値は SciPy ライブラリを使っても求められる。SciPy ライブラリは非常に大きいので、統計の部分 stats だけインポートする。

from scipy import stats

stats.binom.pmf(3, 10, 0.4)

# 0.21499084799999982

投げると確率 0.5 で表が出る硬貨を10回投げて表が0〜10回出る確率を全部出力するには次のようにする。

stats.binom.pmf([0,1,2,3,4,5,6,7,8,9,10], 10, 0.5)

# ([0.00097656, 0.00976563, 0.04394531, 0.1171875, 0.20507812, 0.24609375, 0.20507812, 0.1171875, 0.04394531, 0.00976563, 0.00097656])

#

# coin flip

##################################################################

import matplotlib.pyplot as plt

import scipy.stats as stats

import numpy as np

import seaborn as sns

# import pymc3 as pm

plt.style.use('seaborn-darkgrid')

def posterior\_grid(grid\_points=100, heads=6, tosses=9):

grid = np.linspace(0, 1, grid\_points)

print(grid)

prior = np.repeat(5, grid\_points)

print(prior)

likelihood = stats.binom.pmf(heads, tosses, grid)

print(likelihood)

unstd\_posterior = likelihood \* prior

posterior = unstd\_posterior / unstd\_posterior.sum()

return grid, posterior

points = 15

h, n = 1, 4

grid, posterior = posterior\_grid(points, h, n)

plt.plot(grid, posterior, 'o-', label='heads = {}\ntosses = {}'.format(h, n))

plt.xlabel(r'$\theta$', fontsize=14)

plt.legend(loc=0, fontsize=14)

plt.savefig('img201.png')

・・・

(mypm3env) PS C:\Users\yokan\desktop> python coin.py

[0. 0.07142857 0.14285714 0.21428571 0.28571429 0.35714286

0.42857143 0.5 0.57142857 0.64285714 0.71428571 0.78571429

0.85714286 0.92857143 1. ]

[5 5 5 5 5 5 5 5 5 5 5 5 5 5 5]

[0. 0.22875885 0.35985006 0.41576426 0.41649313 0.37952936

0.31986672 0.25 0.17992503 0.11713869 0.0666389 0.03092461

0.00999584 0.0013536 0. ]

・・・

np.random.seed(123)：発生した乱数をシード123に固定する

stats.bernoulli.rvs(p, size)：二項分布（1か０）のランダムデータのサンプリング

pm.Beta('theta', alpha=1, beta=1)変数Θ、α＝１、β＝１のベータ分布

pm.Bernoulli('y', p=theta, observed=data)変数ｙ、確率Θ、観測値＝０か１の二項分布

pm.sample（1000,chains=4）： NUTSのサンプリング1000個によって事後分布を4ケース推定する

burnin = 100：最初の方の削除するサンプル数

multi\_chain = multi\_trace[burnin:]：burninの数を削除したサンプル

pm.summary(trace)：事後分布の要約（平均、標準偏差、HPI区間、誤差の推定値等）を出力

　　　　　誤差の推定値　MCerror＝σ(x)/√n

mean sd hdi\_3% hdi\_97% mcse\_mean mcse\_sd ess\_bulk ess\_tail r\_hat

theta 0.335 0.179 0.044 0.661 0.006 0.004 1065.0 1268.0 1.0

pm.traceplot：左側に事後分布、右側にバーンイン以降に得られたサンプルの軌跡を表示

pm.autocorrplot(trace)：自己相関グラフを表示

pm.plot\_posterior(trace)：事後分布と95%HPIを出力

pm.plot\_posterior(trace, rope=[0.45,.55])：事後分布とROPE（実質同値域）を出力

pm.plot\_posterior(trace, ref\_val=0.5)：：事後分布と参照点を出力

【第3章　複数パラメータの取り扱いと階層モデル】

３．１　迷惑パラメータの周辺化：

P(Θ1|y)=∫p(Θ1,Θ2|y)dΘ2

2つのパラメータΘ1、Θ2をもつ事後分布p(Θ1,Θ2|y)を、迷惑パラメータΘ2で積分することで周辺化する。

３．２　推論のやり方

1. ガウシアン推論

正規分布を仮定し、事前分布としては区間[l,h]で一様な分布と半正規分布を用いる。

1. 頑健推論

スチューデントのｔ分布（パラメータνが無限大に近づくと正規分布に近づく）を仮定し、事前分布としては区間[l,h]で一様な分布と半正規分布と指数分布を用いる。

３．３　グループ間の比較

新薬テストにおけるグループ間の比較：

・トリートメントグループ：新薬を処方

・コントロールグループ：新薬を処方しない

チップのデータセットの比較：

客が払う曜日ごとのチップの合計（木、金、土、日の４曜日のデータあり）

コントロールグループは存在せず、曜日間の比較になる

ただし、参照基準として木曜日に決めたら、これがコントロールグループになりうる

効果量の測定

・コーエンのｄ

δ＝(μ1-μ2)/√(σ12+σ22)/2

・優越率

Ps=Φ(δ/√2)

３．４　階層モデル

事前分布として、パラメーターを定数に固定するのではなく、さらに事前分布を（＝ハイパー事前分布）を設定して、データから直接に事前分布を推定する。

【第4章　線形回帰モデルによるデータの理解と予測】

線形回帰モデルとは、ある変数の別の変数への影響を調べるために用いられる手法で、変数間の関係を線形に表現して統計学的な分析を可能にするもの。単回帰、線形重回帰、ロジスティック回帰、分散分析（ANOVA）、共分散分析（ANCOVA）などがある。確率的には、線形回帰モデルは次のように表現される。ベクトル**ｙ**が平均（α＋β**ｘ**）、標準偏差εの正規分布に従うと仮定するのである。

**ｙ**～Ｎ（μ＝α＋β**ｘ**，σ＝ε）

階層線形回帰と線形重回帰の例を考える。

まず、階層線形回帰。

ここでは、以下の散布図で示される8種類のデータに対して、回帰直線を求める。

本ケースでは2つの半コーシー分布（σα、σβ）と2つの正規分布（μα、μβ）のハイパー事前分布から2つの正規分布（μα、μβ）の事後分布を求め、これに半コーシー分布（σσ）と階層分布（ν）を加えたものを再度事前分布として、ｔ分布の事後分布を求める。得られた結果は以下の通り。データ点が一つしかないものでも回帰直線が求まっているから驚きだ。これがベイズ統計の魅力なのかもしれない。

次は、線形重回帰。このケースでは、従属変数の平均値を次のようにモデル化する。ここで、**β**はm次のベクトルで、**Ｘ**はm×nの行列である。線形重回帰は本質的には線形単回帰と同じモデルだが、係数**β**がベクトルであり、独立変数**Ｘ**が行列であるという違いがある。

μ＝α＋β1ｘ1＋β2ｘ2＋β3ｘ3＋・・・＋βmｘm＝α＋**βＸ**

今回はm＝2の場合を扱う。事前分布としてα、βは正規分布、εは半コーシー分布を採用し、正規分布でμの事後分布を求める。得られた事後分布とトレース、および要約統計量は以下の通り。

―――――――――――――――――――――――――――――――

　　　　平均 　　　標準偏差　　　HDI 3%　HDI 97% 　　誤差推定値

α 　 -0.001 　　　　 0.10 　　 -0.18 0.19 　　 2.00e-03

β1　 0.955 　　　　0.16 　　 0.65 1.24 　　 3.00e-03

β2 　-0.052 　　　 　0.09 　　　 -0.23 0.12 　　 2.00e-03

ε 　 1.0 　　　　　0.07 　　0.87 1.14 　　1.00e-03

―――――――――――――――――――――――――――――――

係数β（β1，β2）の2次元の事後分布から係数同士に強い相関があることが分かる。

【第5章　ロジスティック回帰による結果変数の分類】

線形回帰モデルを質的（カテゴリカル）な被予測変数の場合に拡張する。カテゴリカルな変数を扱う場合は、観測データをクラスに割り当てる課題である“分類”が必要となる。

2クラスの場合：ロジスティック回帰

独立変数が1つの場合：単純ロジスティック回帰

独立変数が複数の場合：多重ロジスティック回帰

3クラス以上の場合：多項ロジスティック回帰（ソフトマックス回帰）

μ＝ｆ（α＋β**Ｘ**）

ここで、ｆは逆連結関数（出力変数を線形モデルに連結する関数）で、線形回帰モデルを分類問題に効果的に変換してくれる。逆連結関数としては、これまでは、恒等関数（引数の値と同じ値を返す関数）だったが、ここではロジスティック関数を用いる。実数全体を区間［０，１］に圧縮してくれ、シグモイド関数の一つである。

Logistic（ｚ）＝１/（１＋exp(―ｚ)）

ロジスティック回帰モデルは、以下の式で表される。正規分布やスチューデントのｔ分布の代わりにベルヌ―分布を用い、ベルヌ―分布に値を渡す［０，１］の範囲のパラメータΘを生成している。Θ＝0.5のとき、α＋β**Ｘ**＝0、すなわち、**Ｘ**＝―α/βとなる。

Θ＝logistic(α＋β**Ｘ**)

ｙ～Bern(Θ)

ここでは、seabornライブラリのアイリスデータセットを使う。アイリスの3つの品種（setosa、versicolor、virginica）がクラスとなり、それぞれ50個の個体があり、各品種に４つの独立変数（ガク片の長さ、ガク片の幅、花弁の長さ、花弁の幅）が含まれている。

単純ロジスティック回帰：

2つのクラスとしてsetosaとversicolor、独立変数としてガク片の長さ（x）をとる。setosaとversicolorを０と１にコード化（y）する。

theta：μにロジスティック関数を適用した結果

花の品種がversicolorである確率：Θ＝p(y=1|x）

bd：連続的な数値をクラスへと分ける作業に使われる境界値

y＝0.5に対応するxの値：－α/β

多重ロジスティック回帰：

ここでは、独立変数として、ガク片の長さとガク片の幅を組み入れる。

Θ＝logistic(α＋β0x0＋β1x1)

Θ＝0.5のときα＋β0x0＋β1x1＝0であるから、

x1＝-α/β1―（β0/β1）x0

境界決定は1本の直線となる。

クラスによってデータ数が異なる場合は、アンバランスなクラスとなり、正確な境界決定ができなくなる。

多項ロジスティック回帰：

3つ以上のクラスを扱う場合、ロジスティック関数の代わりにソフトマックス関数を使う。∑はk=1～Ｋの和。ソフトマックス関数は統計力学のボルツマン関数と同じ形をしており、Ｋ＝2の場合にロジスティック関数となる。

Softomaxi(**μ**)＝exp(μi)/∑exp(μi)

ソフトマックス回帰モデルでは、ロジスティック回帰モデルのベルヌ―分布をカテゴリカル分布に置き換える。ベルヌ―分布が2項分布の特殊な場合であるように、カテゴリカル分布は多項分布の特殊な場合になっている。

ここでは、アイリスデータのすべて、すなわち3つのクラス、4つの独立変数を使う。その結果は、識別不能問題が生じて、事後分布が幅広くなってしまう。これを避けるために、余分なパラメータをある値、例えば0にしてしまうことで、良い結果が得られる。

判別モデルと生成モデル：

判別モデル：独立変数から従属変数へと直接的に写像するモデルを作り、その後、ある閾値を使って、計算された連続的な確率をクラスに割り当て可能にする境界に変換する。例：ロジスティック

回帰モデル。

生成モデル：ｐ（**ｘ**｜**ｙ**）、すなわち、各クラスに関して**ｘ**の分布をモデル化し、その後、クラスへ割り当てる。例：線形判別分析（LDA）および、2つ以上の特殊変数に拡張した2次線形判別分析（QDA）。

特殊関数が正規分布に従っている場合は生成モデルが良い結果を与え、正規分布に従わない場合は判別モデルが良い結果を与える。

【第6章　モデル比較】

モデルの選び方：

1. 単純な方を選ぶ（オッカムのカミソリ）
2. モデルの精度の高い方を選ぶ（例えば、決定係数Ｒ2で比較）

モデルの精度：

1. サンプル内精度：モデルにフィットさせたデータによる精度
2. サンプル外精度：モデルにフィットさせたデータとは異なるデータによる精度

・多すぎるパラメータは過剰適合をもたらす

・少なすぎるパラメータは過少適合をもたらす

・データに適合する能力のないモデルを使うと過少適合を起こす：ハイバイアス（偏り）

・個々のデータに対して非常に敏感なモデルを使うと過剰適合を起こす：ハイバリアンス（分散）

事前分布の正規化：

1. リッジ回帰：β係数に対して正規分布を使う
2. ラッソ回帰：β係数に対してラプラし事前分布を使う

サンプル内データのみを用いてサンプル外予測精度を推定する方法：

1. 交差確認：データをパラメータ推定やフィッティングに使う部分と推定された結果を評価する部分に分ける、例：1個抜き交差確認（LOOCV）
2. 情報量規準：交差確認によって得られるような結果を近似する数式

・誤差の2乗平均

・対数尤度

・逸脱度

・赤池情報量規準（AIC）：平坦な事前分布を仮定し事後分布を使わない

・逸脱度情報量規準（DIC）：事後分布の一部を使う

・広く使える情報量規準（WAIC）：事後分布の前部を使う

・パレート平滑化重点サンプリング（PSIS）： LOOCVの結果をLOOCVを実行せずに近似

・ベイジアン情報量規準（BIC）：平坦な事前分布を仮定し、最尤推定値を使う

ベイズファクター：

ベイズ分析において、モデルを評価したり比較したりする一般的な方法で、次式で表される。ここで、p(y|M)はベイズの定理の右辺分母。

BF=p(y|M0)/p(y|M1)

BF数値範囲 解釈例

1～3 少し望ましい

3～10 適度に望ましい

10～30 かなり望ましい

30～100 非情に望ましい

＞100 極端に望ましい

【第7章　混合モデル】

混合モデル：分布の混合からデータが発生していると仮定するモデル

混合モデルの例：

・カテゴリカル分布：ベルヌーイ分布をｋ個の結果へと一般化したもの

・ディリクレ分布：ベータ分布を一般化したもの

・周辺化されたガウス混合モデル：潜在変数を迷惑変数と考え、これを周辺化する

・ポアソン回帰：カウントデータのモデル化にポアソン分布を用いる

・ゼロ過剰ポアソン（ZIP）回帰：ポアソン分布を確率ψ、過剰な0を確率（1-ψ）で組み合わせる

・頑健ロジスティック回帰：過剰な0に対処するのにロジスティック回帰を使う

・クラスタリング：正しいラベルが分からない場合、互いに似ているデータ点は同じグループに、似ていないデータ点は異なるグループにグルービングする

ハードクラスタリング：各データが１つのクラスターのみに属する

ソフトクラスタリング：各データが複数のクラスターに属する

・連結型混合モデル：

ロジスティック回帰

階層モデル（β2項分布、連続型ガンマ-ポアソン混合モデル、スチューデントのt分布）

【第8章　ガウス過程】

機械学習：

《パラメトリックモデル》

ｙ＝ax+bのように形を決めておいて、データからパラメター（aとb）を求める

《ノンパラメトリックモデル》

形を仮定しないで、データにフィットするように滑らかな線を引く

カーネル：2つの入力から常に正の値を1つ出力する対称関数、2つの入力間の類似度を表わす

（例）ガウスカーネル（ガウス動径既定関数）

Ｋ(x,x’)＝exp（-‖x-x’‖2/w）

‖x-x’‖2：ユークリッド平方距離

w：帯域幅

ここで、**x**：データ、**x**’：ノットまたは重心、ガウスカーネルはx＝x’で1、xとx’が遠く離れると0に近づくので平滑関数とも呼ばれる。

カーネルベースモデル：

・サポートベクターマシン（SVM）：確率的方法ではない

・重み空間モデル：確率的方法

・関数空間モデル：ガウス過程（GP）、確率的方法

多項式回帰の関数φにガウスカーネルを用いる。

**μ**＝∑γiΦi(**x**)＝∑γiＫi(**x**,**x**’)

重み空間モデル：

グリッドアプローチ、すなわち、各ノットにガウス関数を置き、データ点に応じてガウス関数の重みを上げたり下げたりすると、**μ**の値が平滑化される。

ガウス過程：

確率変数の同時分布を次式で定義する

ｆ(**x**)～GP(**μ**(**x**)，Ｋ(**x**,**x**’))

ここで、**μ**(**x**)：平均関数（ベクトル）、Ｋ(**x**,**x**’)：共分散関数（行列）

GPを使ってｐ(ｆ|**ｘ**)を推定できる

無限のGP事前分布を有限の多変量正規分布へと縮小させる

通常、平均関数は0とする、すなわち、**μ**＝［0・・・0］

共分散行列はパラメータ化されたカーネルによって定義する（ハイパーパラメータ）

Ｋij＝ηexp(-ρD）(i≠jの場合)、η+σ(i=jの場合)

ここで、D：ユークリッド平方距離（SED）

η：垂直方向の尺度を制御する

ρ：帯域幅、関数の滑らかさを制御する

σ：データに含まれるノイズを捉える

GP事前分布と正規分布の尤度を組み合わせると、GPの事後分布が得られる

平均と共分散の計算には逆行列を求める必要があり、計算時間がかかりすぎるので、代替え案としてコレスキー分解を行なう。

以上、まとめると、

・GP事前分布：ｆ(**x**)～GP(**μ**＝［0・・・0］，Ｋ(**x**,**x**’))

・正規分布による尤度：ｐ(y|x,f(x))～Ｎ(**f**,σ2**I**)

・GP事後分布：ｐ(f(x)|x,y)～GP(μPOST,∑POST)

データが周期的であると知っている時は、周期カーネルを使うべきである。データの少ない周辺部での不確実性が抑え込まれる。