Probabilistic Programming

Marius Popescu

popescunmarius@gmail.com

2019 - 2020

Mock Exam | Discussion | Questions

Mock Exam

Question

Let us assume a probability distribution for count data. Which of the following type of distribution we would choose for modelling:

- ❑ Bernoulli distribution
- ❑ Exponential distribution
- ☑ Poisson distribution

Problem

Write two different PyMC stochastic variables that can represent a discrete random variable X that takes value in {0, 1, 2, 3, 4, 5}. Which one is the must uninformative?

- `X = pm.DiscreteUniform(“X", lower = 0, upper = 5)`
- `X = pm.Categorical(“X”, p = [0.1, 0.1, 0.1, 0.2, 0.2, 0.3])`
- `DiscreteUniform` (or both if the values of `p` are equal)

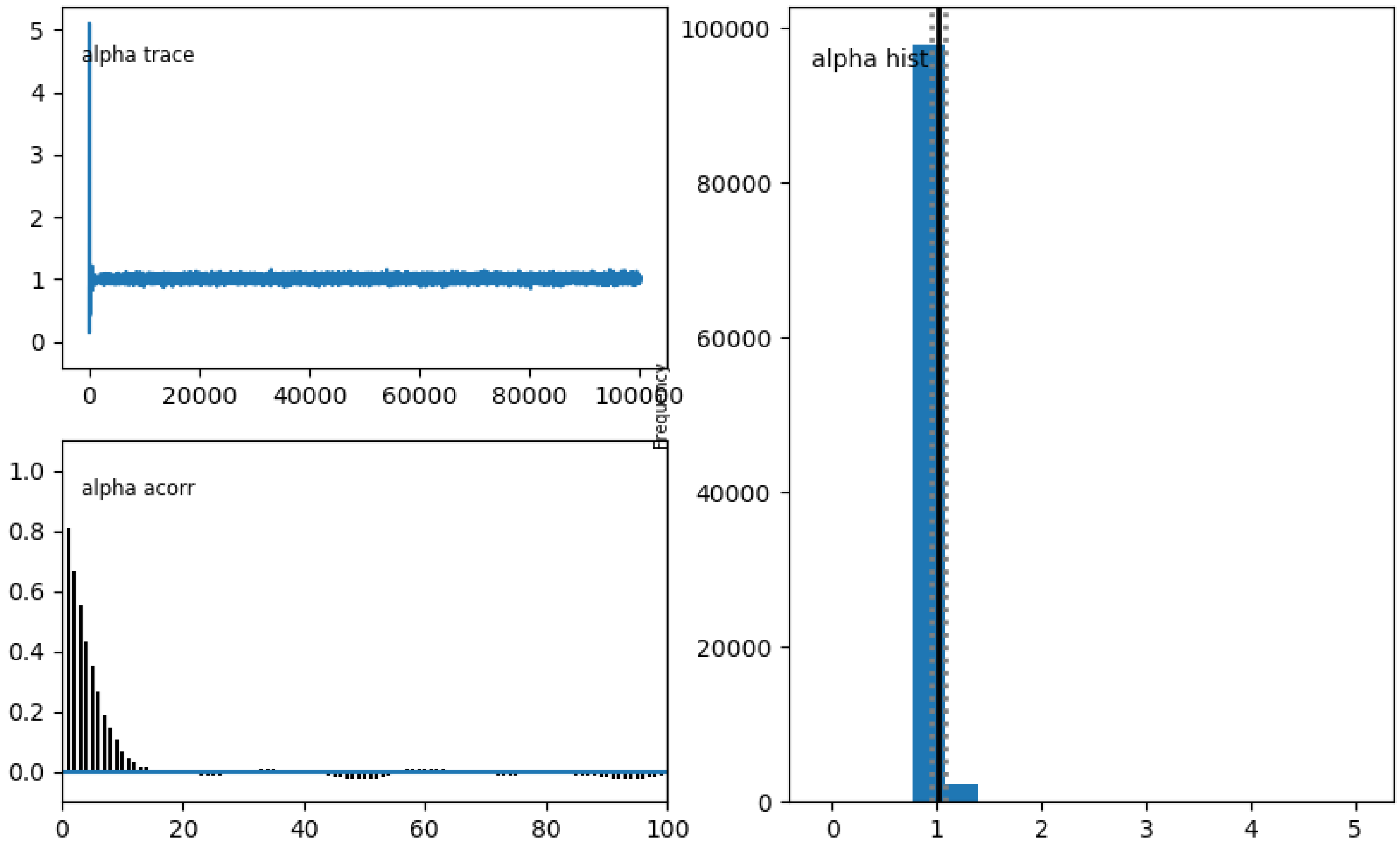
Question

From the following distribution which one is a conjugate prior for the Bernoulli distribution:

- ❑ Gamma distribution
- ☑ Beta distribution
- ❑ Uniform distribution on (0, 1)

Question

The figure bellow is the result of `pymc.Matplot.plot` of a stochastic variable **alpha** after MCMC. Did the MCMC converged? Justify your answer.



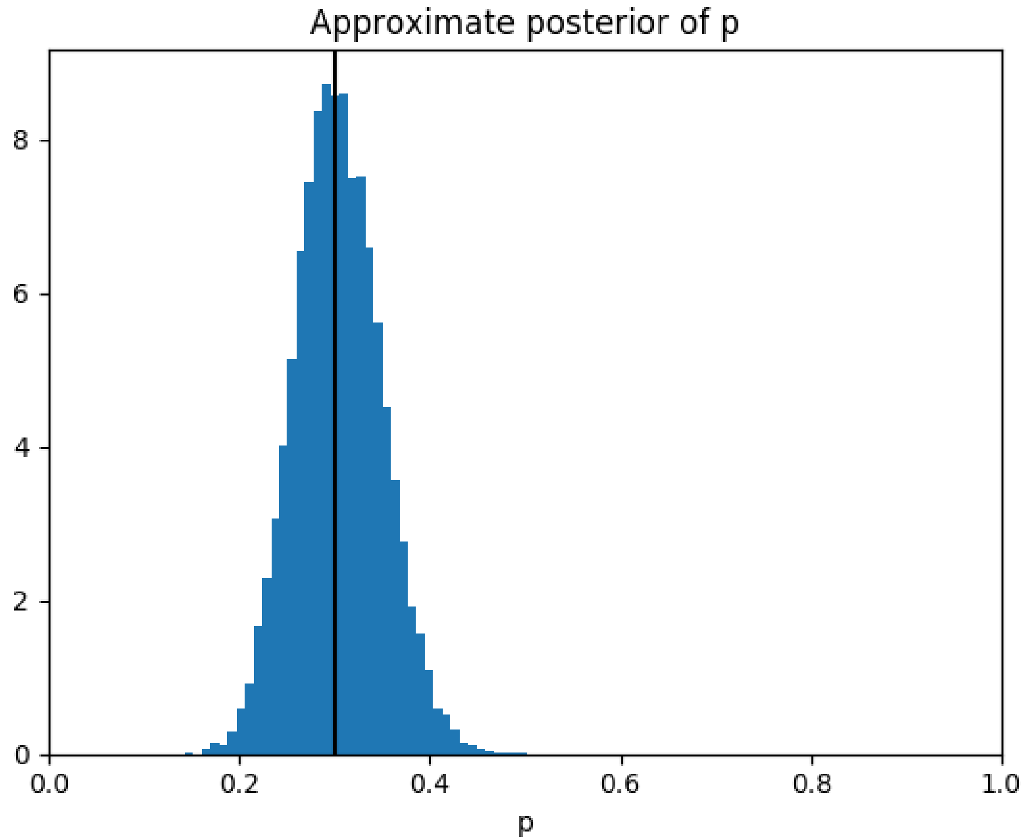
The MCMC has converged. Reasons:

- The trace shows that after varying a lot at the beginning (1000 – 2000 iterations) the chain tends toward an equilibrium value, with a smaller variance.
- The histogram of the posterior distribution of alpha is concentrated in a small area (around 1)
- The autocorrelation becomes and remains small (after lag 16 – 18)

Problem

Write a PyMC program that infers the bias of a coin given 100 observations of the coin flips

```
import pymc as pm
import numpy as np
import matplotlib.pyplot as plt

true_coin_bias = 0.3 # The (unknown) bias of the coin
num_flips = 100

# The given data, the result of 100 coin flips
data = np.random.choice([0, 1], p=[1-true_coin_bias, true_coin_bias], size=num_flips)

# We want to infer the bias, p. Since p is unknown, it is a random variable.
# The distribution we assign to it here is our prior distribution on p, uniform on the range [0, 1].
p = pm.Uniform("p", lower=0, upper=1)

# We need another random variable for our observations.
# We give the relevant data to the value argument.
# The observed flag stops the value changing during MCMC exploration.
observations = pm.Bernoulli("obs", p=p, value=data, observed=True)

model = pm.Model([p, observations])
mcmc = pm.MCMC(model)
mcmc.sample(60000, 10000) # 60000 steps, with a burn in period of 10000
p_samples = mcmc.trace("p")[:]   # Samples from our posterior on p


print()
print(p_samples.mean())

plt.hist(p_samples, bins=40, normed=True)
plt.axvline(x=true_coin_bias, c="k")
plt.xlabel("p")
plt.title("Approximate posterior of p")
plt.xlim(0,1);
plt.show()
```



{C:\My\ProbabilisticProgrammingCourse\PyMC2\14} - Far 3.0.5225 x64 Administrator

```
C:\...listicProgrammingCourse\PyMC2\14>                                     9:37 AM
C:\...listicProgrammingCourse\PyMC2\14>
C:\...listicProgrammingCourse\PyMC2\14>
C:\...listicProgrammingCourse\PyMC2\14>
C:\...listicProgrammingCourse\PyMC2\14>
C:\...listicProgrammingCourse\PyMC2\14>
C:\...listicProgrammingCourse\PyMC2\14>
C:\...listicProgrammingCourse\PyMC2\14>
C:\...listicProgrammingCourse\PyMC2\14>
C:\...listicProgrammingCourse\PyMC2\14>
C:\...listicProgrammingCourse\PyMC2\14>
C:\...listicProgrammingCourse\PyMC2\14>
C:\...listicProgrammingCourse\PyMC2\14>
C:\...listicProgrammingCourse\PyMC2\14>
C:\...listicProgrammingCourse\PyMC2\14>
C:\...listicProgrammingCourse\PyMC2\14>
C:\...listicProgrammingCourse\PyMC2\14>
C:\...listicProgrammingCourse\PyMC2\14>
C:\...listicProgrammingCourse\PyMC2\14>
C:\...listicProgrammingCourse\PyMC2\14>python coin_bias.py
[-----------------100%-----------------] 60000 of 60000 complete in 2.0 sec
0.3039609316290384

C:\...listicProgrammingCourse\PyMC2\14>
1Left  2Right  3View..  4Edit..  5Print  6MkLink  7Find  8Histry  9Video  10
```

Discussion | Questions