SDS386D: Monte Carlo Methods in Statistics **Cesar N. Yahia**
end-of-term-exam Prof. S.G. Walker

Bayesian inference for two models:
$M_1 = \{g_1(x|\theta) = \frac{\theta^x}{x!}e^{-\theta}, x \in \{0,1,2,...\}; \pi_1(\theta) = gamma(1,1)\}$ Poisson
$M_2 = \{g(x|\theta) = \theta(1-\theta)^x, x \in \{0,1,2,..\}; \pi_2(\theta) = beta(1,1)\}$ Geometric

prior model probabilities $f(M_1) = f(M_2) = \frac{1}{2}$

Find $f(M_1|data)$:
To find the posterior probability of using model 1 given the data, we need to construct a Markov chain that jumps between models. We can do so using a properly constructed MH algorithm that is similar to the algorithm used for estimating the number of components in a mixture model. The MH algorithm would jump between M_1 and M_2, and after sufficient iterations of the algorithm, we can find $f(M_1|data)$ as the proportion of time the chain was at model 1.

MH algorithm & required latent variables:
A simple proposal for jumping between models could be that we always propose to move to the other model. Therefore,
$q(M'|M) = 1$ for $M' \neq M$. The α for the MH algorithm is then given by:
$\alpha(M', M) = \min\{1, \frac{f(M',\theta_1,\theta_2|data)}{f(M,\theta_1,\theta_2|data)} \frac{q(M|M')}{q(M'|M)}\} = \min\{1, \frac{f(M',\theta_1,\theta_2|data)}{f(M,\theta_1,\theta_2|data)}\}$

$f(M_1, \theta_1|data) \propto f(data|M_1, \theta_1)f(\theta_1|M_1)f(M_1)$
However, since we propose to move to the other model at a particular time step, to do this using a Metropolis step, we need to augment the posterior for M_1 to include the parameters for M_2. Therefore, we introduce a latent variable corresponding to the parameters of model 2 in the posterior for M_1
This results in:
$f(M_1, \theta_1, \theta_2|data) \propto f(data|M_1, \theta_1)f(\theta_1|M_1)f(M_1)P(\theta_2|\theta_1)$
Similarly,
$f(M_2, \theta_1, \theta_2|data) \propto f(data|M_2, \theta_2)f(\theta_2|M_2)f(M_2)P(\theta_1|\theta_2)$
At any step in the MH algorithm, in anticipation of moving to the other model, sample the parameters θ' of the other model M' based on the parameters θ of the current model M. One easy way to sample the parameter for the Geometric distribution is to use a uniform random variable on (0,1). Therefore, $P(\theta_2|\theta_1) = 1$. As for sampling θ_1 for the Poisson, we can use a lognormal distribution with mean zero and variance 1. Therefore, $P(\theta_1|\theta_2) = \frac{1}{\theta_1\sqrt{2\pi}}e^{-\frac{(log(\theta_1))^2}{2}}$

A few more things should be defined before we outline the algorithm:
$\star$ $f(data|M_1, \theta_1) = \prod_{i=1}^{100} \theta_1^{x_i} e^{-\theta_1} (1/x_i!)$
$\star$ $f(\theta_1|M_1) = e^{-\theta_1}$
$\star$ $f(M_1) = \frac{1}{2}$
$\star$ $f(data|M_2, \theta_2) = \prod_{i=1}^{100} \theta_2 (1-\theta_2)^{x_i}$
$\star$ $f(\theta_2|M_2) = 1$
$\star$ $f(M_2) = \frac{1}{2}$

$\star$ $\alpha(M_2, M_1) = \min\{1, \frac{f(data|M_2,\theta_2) f(\theta_2|M_2) f(M_2) P(\theta_1|\theta_2)}{f(data|M_1,\theta_1) f(\theta_1|M_1) f(M_1) P(\theta_2|\theta_1)}\} = \min\{1, \frac{\prod_{i=1}^{100} \theta_2(1-\theta_2)^{x_i} (1/2) \frac{1}{\theta_1\sqrt{2\pi}} e^{-\frac{(log(\theta_1))^2}{2}}}{\prod_{i=1}^{100} \theta_1^{x_i} e^{-\theta_1} (1/x_i!) e^{-\theta_1} (1/2)}\} =$
$\min\{1, \frac{(\prod_{i=1}^{1} \theta_2(1-\theta_2)^{x_i}) \frac{1}{\theta_1\sqrt{2\pi}} e^{-\frac{(log(\theta_1))^2}{2}}}{(\prod_{i=100}^{100} \theta_1^{x_i} e^{-\theta_1} (1/x_i!)) e^{-\theta_1}}\}$

$\star$ $\alpha(M_1, M_2) = \min\{1, \frac{(\prod_{i=100}^{100} \theta_1^{x_i} e^{-\theta_1} (1/x_i!)) e^{-\theta_1}}{(\prod_{i=1}^{1} \theta_2(1-\theta_2)^{x_i}) \frac{1}{\theta_1\sqrt{2\pi}} e^{-\frac{(log(\theta_1))^2}{2}}}\}$

The MH procedure is then given as follows:
considering you are at model M_1 at the current iteration
(1) sample u from $U(0,1)$
(2) sample θ_2 from $P(\theta_2|\theta_1)$
(3) evaluate $\alpha(M_2, M_1) = \min\{1, \frac{(\prod_{i=1}^{100} \theta_2(1-\theta_2)^{x_i}) \frac{1}{\theta_1\sqrt{2\pi}} e^{-\frac{(log(\theta_1))^2}{2}}}{(\prod_{i=100}^{100} \theta_1^{x_i} e^{-\theta_1} (1/x_i!)) e^{-\theta_1}}\}$
(4) if $u < \alpha(M_2, M_1)$, then $M' = M_2$, else $M' = M_1$, where M' is the model at the next iteration

Similarly, if you are at M_2 at the current iteration
(1) sample u from $U(0,1)$
(2) sample θ_1 from $P(\theta_1|\theta_2)$
(3) evaluate $\alpha(M_1, M_2) = \min\{1, \frac{(\prod_{i=100}^{100} \theta_1^{x_i} e^{-\theta_1} (1/x_i!)) e^{-\theta_1}}{(\prod_{i=1}^{1} \theta_2(1-\theta_2)^{x_i}) \frac{1}{\theta_1\sqrt{2\pi}} e^{-\frac{(log(\theta_1))^2}{2}}}\}$
(4) if $u < \alpha(M_1, M_2)$, then $M' = M_1$, else $M' = M_2$

Across iterations, count the number of times you visit M_1 (n_{M_1}) and the total number of iterations n. Then $f(M_1|data) = \frac{n_{M_1}}{n}$.

The algorithm was implemented in Python as shown below:

```
"""
This code is for Model choice problem
on the end_of_term_exam for the
SDS386D exam

@cnyahia
"""

import numpy.random as nprand
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats
import math


# define sampling from Poisson
def sample_Poisson(lam, sample_size=1):
        """
        sample from a Poisson
        :param lam: Poisson rate lambda
        :param sample_size: number of sampled values
        :return: sampled values
        """
        if sample_size == 1:
        samples = nprand.poisson(lam=lam)
        else:
        samples = list(nprand.poisson(lam=lam, size=sample_size))
        return samples


# compute likelihood from Poisson M1
def like_M1(theta, data):
        """
        computes the likelihood for the Poisson model M1
        :param theta: poisson parameter
        :param data: list of data points
        :return: likelihood
        """
        likelihood = 1
        for data_point in data:
        likelihood = likelihood * (theta**data_point) * (np.exp(-theta))
                        * (1.0 / math.factorial(data_point))

        return likelihood


# compute likelihood from Geometric M2
```

```
def like_M2(theta, data):
        """
        computes the likelihood for the Geometric model M2
        :param theta: parameter of geometric dist
        :param data: list of data points
        :return: likelihood
        """
        likelihood = 1
        for data_point in data:
        likelihood = likelihood * theta * ((1 - theta)**data_point)

        return likelihood


# def evaluate lognormal
def eval_lognormal(point):
        """
        evaluates a standard log normal
        :param point: point of evaluation
        :return: value
        """
        eval = stats.lognorm.pdf(point, 1)
        return eval


# def sample lognormal
def sample_lognormal(mean=0.0, sigma=1.0):
        """
        samples a lognormal
        :param mean:
        :param sigma:
        :return: sample
        """
        sample = nprand.lognormal(mean, sigma)
        return sample


# implement algorithm
if __name__ == '__main__':
        # generate 100 samples from Poisson, lambda=1
        true_data = sample_Poisson(1, 100)

        # initialize algorithm
        theta1 = 6
        theta2 = 0.5
        current_model = 2

        # MH algorithm parameters
        num_of_iter = 1
```

```
total_iter = 5000

# number of times visit M1
visits = 0

while num_of_iter <= total_iter:
        u = nprand.uniform(low=0, high=1)
        if current_model == 1:
                visits += 1
                theta2 = nprand.uniform(low=0, high=1)
                numerator = like_M2(theta2, true_data) *
                        eval_lognormal(theta1)
                denom = like_M1(theta1, true_data) *
                        np.exp(-theta1)
                ratio = float(numerator / denom)
                alpha = min(1.0, ratio)
                if u < alpha:
                        current_model = 2

        elif current_model == 2:
                theta1 = sample_lognormal(mean=0.0, sigma=1.0)
                numerator = like_M1(theta1, true_data) *
                        np.exp(-theta1)
                denom = like_M2(theta2, true_data) *
                         eval_lognormal(theta1)
                ratio = float(numerator / denom)
                alpha = min(1.0, ratio)
                if u < alpha:
                        current_model = 1

        num_of_iter += 1

print('number of time visit M1:', visits)
print('total number of iterations', total_iter)
print('posterior probability of M1', float(visits) / total_iter)
```

Running the model for 5000 iterations, we observe that while the algorithms transitions to M_2, the likelihood corresponding to M_1 is significantly less than the likelihood corresponding to M_2 for different values of θ_1 and θ_2. While the likelihood $f(data|M_1, \theta_1)$ is in the range of E^{-50} for $\theta_1 = 1.01$ (which was sampled in the very first iterations), the likelihood $f(data|M_2, \theta_2)$ is in the range E^{-60} or more. The lowest the likelihood $f(data|M_2, \theta_2)$ got was in the range of E^{-64} for $\theta_2 = 0.47$. The result is that the probability $f(M_1|data) = 0.98$, since the algorithm spends about 98% of the time in model 1.

A strong reason for that I believe is due to the lognormal proposal $P(\theta_1|\theta_2)$ that I used. This proposal samples θ_1 from a lognormal with mean zero and variance 1 regardless of θ_2, using this proposal, a θ_1 very close to 1 is sampled in the very first iterations of the algorithm. The likelihood from using a Poisson and this θ_1 will be lower than any likelihood from the Geometric distribution since the true data comes from Poisson with $\theta_0 = 1$. Therefore, we would expect that the model spends most of it's time at M_1 as was observed.

A different way to create the proposals $P(\theta_1|\theta_2)$ and $P(\theta_2|\theta_1)$, is to choose the parameter of the distribution you are transitioning to such that the means of the Poisson and Geometric are the same. Then you can add some perturbation to that parameter. In this case, if you are at M_1, you would choose $\theta_2 = \frac{1}{1+\theta_1}$ with some perturbation. Equivalently, if you are at M_2, you would choose $\theta_1 = \frac{1-\theta_2}{\theta_2}$ with some perturbation. The challenge with this method is that you need to keep the sample parameters within their allowable range. Specifically, $0 \leq \theta_2 \leq 1$ and $\theta_1 > 0$