Bayesian Models HW1

Problem 1

Suppose these three doors are A, B, C. My friend picked door A, and the gameshow host opened door B which contains no prize.

For all the doors,
their prior probabilities are all the same;

$$P(\text{prize@A}) = P(\text{prize@B}) = P(\text{prize@C}) = \frac{1}{3}$$

(Note: prize@A means prize is at A)

After my friend chose door A:
chances for the host to open door B if the prize is at door A:

$$P(\text{open B} \mid \text{prize@A}) = \frac{1}{2}$$

Chances for the host to open door B if prize is at door B:

$$P(\text{open B} \mid \text{prize@B}) = 0$$

Also,

$$P(\text{open B} \mid \text{prize@C}) = 1$$

Then the posterior probabilities :

$$P(\cancel{\text{car}}\ \text{prize @ A} \mid \text{Open B}) = \frac{X}{X+Y+Z}$$

$$= \frac{\frac{1}{2}\times\frac{1}{3}}{\frac{1}{2}\times\frac{1}{3}+0\times\frac{1}{3}+1\times\frac{1}{3}}$$

$$= \frac{1}{3}$$

where $X = P(\text{Open B} \mid \text{prize @ A}) \times P(\text{prize @ A})$

$= \frac{1}{2}\times\frac{1}{3}$

$Y = P(\text{open B} \mid \text{prize @ B}) \times P(\cancel{\ }\ \text{prize @ B})$

$= 0\times\frac{1}{3}$

$Z = P(\text{open B} \mid \text{prize @ c}) \times P(\text{prize @ c})$

$= 1\times\frac{1}{3}$

$$P(\text{prize @ B} \mid \text{open B}) = \frac{P(\text{open B} \mid \text{prize @ B})\, P(\text{prize @ B})}{P(\text{open B})}$$

$$= \frac{0\times\frac{1}{3}}{\frac{1}{2}\times\frac{1}{3}+0\times\frac{1}{3}+1\times\frac{1}{3}}$$

$$= 0$$

$$P(\text{prize @ c} \mid \text{open B}) = \frac{P(\text{open B} \mid \text{prize @ c})\, P(\text{prize @ c})}{P(\text{open B})}$$

$$= \frac{1\times\frac{1}{3}}{\frac{1}{2}\times\frac{1}{3}+0\times\frac{1}{3}+1\times\frac{1}{3}}$$

$$= \frac{2}{3}$$

So she should switch doors because the chances of winning are 2 times higher ($\frac{2}{3}$ compared to $\frac{1}{3}$)

Problem 2

A conjugate prior for π is a Dirichlet Distribution:

Given $\pi = (\pi_1, \ldots \pi_k)$, $\pi_j \geq 0$, $\sum_j \pi_j = 1$, we need hyperparameters $\vec{a} = \{a_i \mid i=1, \ldots, k\}$ $a_i > 0 \ \forall i$

Then the prior for π is:

$$P(\pi \mid a_1, \ldots a_k) = \frac{1}{B(a)} \prod_{i=1}^{k} x_i^{a^i - 1}$$

where $B(a) = \dfrac{\prod_{i=1}^{k} \Gamma(a^i)}{\Gamma(\sum_{i=1}^{k} a^i)}$

The likelihood function is:

$$P(X \mid \pi) = \frac{n!}{n_1! \cdots n_k!} \pi_1^{n_1} \cdots \pi_k^{n_k}$$

$\therefore$ its posterior distribution is:

$$P(\pi \mid X) = \frac{1}{Z} \prod_{i=1}^{k} \pi_i^{a^i + n_i - 1}$$

Feature:
Which has the same form as Dirichlet Distribution.

So they are the conjugate pair.

Problem 3

a) Likelihood

$$L(\cancel{\lambda|x}) = \prod_{i=1}^{n} \frac{e^{-\lambda}\lambda^{x_i}}{x_i!} = \frac{e^{-n\lambda}\lambda^{\Sigma x_i}}{\Pi_{i=1}^{n}(x_i!)}$$

$x|\lambda$

prior

$$P(\lambda) = \frac{b^a}{\Gamma(a)}\lambda^{a-1} e^{-b\lambda}, \quad \lambda > 0$$

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$$

$$\pi(\lambda|x) = \frac{L(\lambda|x)\,p(\lambda)}{\int L(\lambda|x)\,p(\lambda)\,d\lambda}$$

$$\propto \lambda^{\Sigma x_i + a - 1} e^{-(n+b)\lambda}, \quad \lambda > 0$$

So $\pi(\lambda|x)$ is $\text{gamma}(\Sigma x_i + a,\ n+b)$

problem 3

(b)

$$p(x^* \mid x_1, \dots, x_n) = \int_0^\infty p(x^* \mid \lambda)\, p(\lambda \mid x_1, \dots, x_N)\, d\lambda$$

$$= \int_0^\infty \left[\frac{\lambda^{x^*} e^{-\lambda}}{(x^*)!} \right] \left[\frac{(n+b)^{\sum x_i + a}}{\Gamma(\sum x_i + a)} \lambda^{\sum x_i + a - 1} e^{-(n+b)\lambda} \right] d\lambda$$

$$= \frac{(n+b)^{\sum x_i + a}}{\Gamma(\sum x_i + a)\Gamma(x^* + 1)} \times \int_0^\infty \lambda^{x^* + \sum x_i + a - 1} e^{-(n+\beta+1)\lambda} d\lambda$$

$$= \frac{(n+b)^{\sum x_i + a}}{\Gamma(\sum x_i + a)\Gamma(x^* + 1)} \frac{\Gamma(x^* + \sum x_i + a)}{(n+b+1)^{x^* + \sum x_i + a}}$$

$$= \frac{\Gamma(x^* + \sum x_i + a)}{\Gamma(\sum x_i + a)\Gamma(x^* + 1)} \left(\frac{n+b}{n+b+1} \right)^{\sum x_i + a} \left(\frac{1}{n+\beta+1} \right)^{x^*}$$

The posterior predictive distribution has the same mean as the posterior distribution, but a greater variance since a new data has been drawn.

Q4 (a)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Sep 23 22:51:42 2018

@author: rainsunny
"""

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from scipy.io import loadmat
from scipy.special import gammaln
Xtrain = pd.read_csv('Users/rainsunny/Desktop/data/X_train.csv')
Labeltrain =
pd.read_csv('Users/rainsunny/Desktop/data/label_train.csv')
a = b = c = e = f = 1.0
dim = Xtrain.shape[0]
n = Xtrain.shape[1]

Xtrain_P = []
Xtrain_N = []

for i in xrange(n):
    if [Labeltrain][:, i][0] == 1:
        Xtrain_P.append(data['Xtrain'][:, i])
    else:
        Xtrain_N.append(data['Xtrain'][:, i])

Xtrain_P = np.transpose(np.array(Xtrain_P))
Xtrain_N = np.transpose(np.array(Xtrain_N))

n_P = Xtrain_P.shape[1]
n_N = Xtrain_N.shape[1]

x_bar_P = []
x_bar_N = []
for i in xrange(15):
    x_bar_P.append(np.mean(Xtrain_P[i]))
    x_bar_N.append(np.mean(Xtrain_N[i]))

x_bar_P, x_bar_N

print (Xtrain_P.shape)
```

```
x_2_P = []
x_2_N = []
for i in xrange(54):
    x_2_P.append(sum(map(lambda x:x * x, Xtrain_P[i, :])))
    x_2_N.append(sum(map(lambda x:x * x, Xtrain_N[i, :])))

mu_n_P = map(lambda x: (a * n_P * x) / (a * n_P + 1), x_bar_P)
mu_n_N = map(lambda x: (a * n_N * x) / (a * n_N + 1), x_bar_N)

kappa_n_P = (a * n_P + 1) / a
kappa_n_N = (a * n_N + 1) / a

alpha_n_P = b + n_P * 0.5
alpha_n_N = b + n_N * 0.5

beta_n_P = []
beta_n_N = []
for i in xrange(54):
    beta_n_P.append(c + 0.5 * x_2_P[i] - 0.5 * ((a * n_P * x_bar_P[i])
** 2)/(a * (a * n_P + 1)))
    beta_n_N.append(c + 0.5 * x_2_N[i] - 0.5 * ((a * n_N * x_bar_N[i])
** 2)/(a * (a * n_N + 1)))

p_y_1_y_star = (e + n_P)/(n + e + f)
p_y_0_y_star = (f + n_N)/(n + e + f)
p_y_1_y_star, p_y_0_y_star

alpha_n_P_1 = alpha_n_P + 0.5
alpha_n_N_1 = alpha_n_N + 0.5

log_kappa_t_P = np.log((kappa_n_P / (kappa_n_P + 1)) ** 0.5)
log_kappa_t_N = np.log((kappa_n_N / (kappa_n_N + 1)) ** 0.5)

log_pi_t = np.log((2 * np.pi) ** -0.5)

log_gamma_t_P = gammaln(alpha_n_P + 0.5) - gammaln(alpha_n_P)
log_gamma_t_N = gammaln(alpha_n_N + 0.5) - gammaln(alpha_n_N)

log_beta_alpha_n_P = map(lambda x: alpha_n_P * np.log(x), beta_n_P)
log_beta_alpha_n_N = map(lambda x: alpha_n_N * np.log(x), beta_n_N)

pred_Y = []
true_P = 0
true_N = 0
false_P = 0
false_N = 0
```

```
print (data['ytest'].shape)

pred_Y = []
true_P = 0
true_N = 0
false_P = 0
false_N = 0
print (data['ytest']).shape
```