Assignment 1

Name: Vuong Minh Phu

Student ID: 202050798

Class: Computer Engineering

Instructor: Prof. Seung-Hoon Na

Problem 3.1

We have

$$p(D|\theta) = \theta^{N_1}(1-\theta)^{N_0}$$

then we apply log to the equation

$$log(p) = N_1 log(\theta) + N_0 log(1-\theta)$$

in order to maximize this function we need to differentiate it and equal it to zero

$$\frac{d(log(p))}{d\theta} = 0$$

$$N_1 \frac{1}{\theta} - N_0 \frac{1}{1-\theta} = 0$$

from this we get the result:

$$\theta_{MLE} = \frac{N_1}{N_1 + N_0} = \frac{N_1}{N}$$

Problem 3.13

We have:

$$N = N^{old} + N^{new}$$
$$N_j = N_j^{old} + N_j^{new}$$

Posterior predictive for single trial:

$$p(X = j|D, \alpha) = \frac{\alpha_j + N_j}{\alpha + N} \tag{3.51}$$

express the batch of data as a series of single trials, we get:

$$p(\tilde{D}|D, \alpha) = p(\tilde{x_1}|D)p(\tilde{x_2}|D, \tilde{x_1})p(\tilde{x_3}|D, \tilde{x_1}, \tilde{x_2})\dots \tag{2}$$

Subsitute (3.51) into (2), we get:

$$p(\tilde{D}|D, \alpha) = \frac{1}{\prod_{i=0}^{N-1}(\alpha + N^{old} + i)} \prod_{j=1}^{k} \prod_{i=1}^{N_j^{new}-1} (\alpha_j + N_j^{old} + i)$$
$$= \frac{\Gamma(\alpha + N^{old})}{\Gamma(\alpha + N)} \prod_{j=1}^{k} \frac{\Gamma(\alpha_j + N_j)}{\Gamma(\alpha_j + N_j^{old})}$$

Problem 3.15

$$mean = m = \frac{a}{a+b}$$
$$\Rightarrow b = a\frac{1-m}{m}$$
$$var = v = \frac{ab}{(a+b)^2(a+b+1)} = \frac{a^2(\frac{1-m}{m})}{a^2(1+\frac{1-m}{n})^2(a+b+1)}$$
$$\Rightarrow (a+b+1) = \frac{1}{v}(\frac{1-m}{m})(\frac{m^2}{m^2 + 2m(1-m) + (1-m)^2}) = \frac{m(1-m)}{v}$$
$$\Leftrightarrow a(1 + \frac{1-m}{m}) = \frac{m(1-m)}{v} - 1$$
$$\Rightarrow a = m(\frac{m(1-m)}{v} - 1)$$
$$\Rightarrow b = (\frac{m(1-m)}{v} - 1)(1-m)$$

For m = 0.7,

$$v = 0.2^2 = 0.04$$

, we get:

$$a = 0.7(\frac{0.7 \times 0.3}{0.04} - 1) = 2.975$$
$$b = (\frac{0.7 \times 0.3}{0.04} - 1)(0.3) = 1.275$$

Problem 3.21

Expression for mutal information between a feature j and the output is:

$$I_j = \sum_{x_j}\sum_{y} p(x_j, y) log\frac{p(x_j, y)}{p(x_j)p(y)} \tag{3.75}$$

For binary features this becomes:

$$I_j = \sum_{y}(p(x_j = 0, y) log\frac{p(x_j = 0, y)}{p(x_j = 0)p(y)} + p(x_j = 1, y) log\frac{p(x_j = 1, y)}{p(x_j = 1)p(y)}) \tag{1}$$

Using chain rule

$$p(x_j = a, y) = p(y)p(x_j = a|y)$$

we can rewrite (1) as follow:

$$I_j = \sum_{y}(p(y)p(x_j = 0|y) log\frac{p(x_j = 0|y)}{p(x_j = 0)} + p(y)p(x_j = 1|y) log\frac{p(x_j = 1|y)}{p(x_j = 1)}) \tag{2}$$

From (3.76) we know that:

$$\pi_c = p(y = c)$$
$$\theta_{jc} = p(x_j = 1|y = c)$$
$$\theta_j = p(x_j = 1) = \sum_{c} \pi_c\theta_{jc}$$

subsitute these into (2) , we get the desired result.

$$I_j = \sum_{y}(\pi_c(1 - \theta jc) log\frac{1 - \theta_{jc}}{1 - \theta_j} + \pi_c\theta_{jc} log\frac{\theta_{jc}}{\theta_j})$$

Problem 3.16

Setting the beta hyper-parameters
User inputs the mean m and the interval [l, u] where

$$p(l < \theta < u) = 0,95.$$

Then, the program determine the hyper parameters of the beta distribution

$$B(\alpha_1, \alpha_2)$$

We will rewrite α_2 as α_1

$$m = \frac{\alpha_1}{\alpha_1 + \alpha_2}$$
$$\alpha_2 = \alpha_1(1/m - 1)$$

Second, we will minimize the squared difference between the probability mass in the interval [l, u] and 0.95 to arrive at the desired result:

$$\left(\int_l^u p(\theta) - 0.95\right)^2$$

We will create some utility functions to compute the integrand expression

$$expr = \frac{1}{B(\alpha_1, \alpha_2)}\theta^{(\alpha_1 - 1)}(1-\theta)^{(\alpha_1(\frac{1}{m} - 1) - 1)}$$

and the objective function

$$(\int_l^u expr d\theta - 0.95)^2$$

note that we have to choose the upper bound greater than lower bound, and the upper and lower bounds should be in the range (0,1)

In [18]:
```
import numpy as np
from scipy import special, integrate
from scipy.optimize import minimize
```

In [20]:
```
def expression(theta, alpha1, m):
    # Expression for the integrand

    beta = special.beta(alpha1, alpha1*(1/m-1))
    expr = (theta ** (alpha1 - 1)) * (1 - theta) ** (alpha1 * (1/m - 1) - 1)
    expr /= beta
    return expr
def func_value(alpha1, m, l, u):
    # Objective function

    integral = (integrate.quad(expression, l, u, args=(alpha1, m)))
    final_expression = (integral[0] - 0.95) ** 2
    return final_expression
# initialize parameters for the function
alpha = 0.3
m = 0.4
u = 0.8
l = 0.1
```

In [21]:
```
# minimize the function value using scipy module
opt_result = minimize(
    func_value,
    x0 = 0.25,
    args=(m, l, u),
    method='L-BFGS-B',
    bounds=((0, None),),
    options={
        'disp': None,
        'maxls': 20,
        'iprint': -1,
        'gtol': 1e-05,
        'eps': 1e-08,
        'maxiter': 15000,
        'ftol': 2.220446049250313e-09,
        'maxcor': 10,
        'maxfun': 15000}
)
alpha1_opt = opt_result.x
opt_value = func_value(alpha1_opt, m, l, u)
print("optimum argument: {0}\toptimum value: {1}".format(alpha1_opt, opt_value))
```

```
optimum argument: [2.45852243]  optimum value: 9.56921267241844e-11
```

Calculate the hypermeters

In [22]:
```
alpha1 = opt_result.x
alpha2 = alpha1 * (1/m - 1)
alpha = b"\xce\xb1".decode('utf-8')
print("{}1 = {}".format(alpha, alpha1))
print("{}2 = {}".format(alpha, alpha2))
```

```
α1 = [2.45852243]
α2 = [3.68778365]
```