

**Sayyed Mohsen Vazirizade**

**23398312**

[**smvazirizade@email.arizona.edu**](mailto:smvazirizade@email.arizona.edu)

**INFO 521**

**Introduction to Machine Learning**

Assignment 4

# Problem 1

Therefore, the new value of is calculated as follow:

# Problem 2

From chapter 3 we know that

0

# Problem 3

The formulation for calculation of the required parameters for normal distribution are provided in the previous example. In this regard, having the values for each parameter of alpha, beta, N and y we can calculate sigma and mean. The following table summarized the calculated values for each state.

Table 1 Summarized calculated value

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| State | Alpha | Beta | N | y | Variance | Mean |
| 1 | 5 | 5 | 20 | 10 | 0.008928571428571428 | 0.5 |
| 2 | 3 | 15 | 10 | 3 | 0.005974055530268548 | 0.19230769230769232 |
| 3 | 1 | 30 | 10 | 3 | 0.0018206645425580337 | 0.07692307692307693 |

The following figures compare Normal and Beta Dist. for posterior for state 1 through 3, respectively. As it can be seen, because Normal Dist. is symmetric inherently, it provides a better estimation when the real posterior function is symmetric. Furthermore, it is obvious that the estimated curve, normal pdf is not exactly as the true posterior function; however, it is a trade of between computing cost and precision.

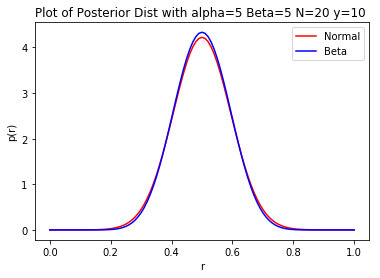


Figure 1Comaring Normal and Beta Dist. for posterior for state 1

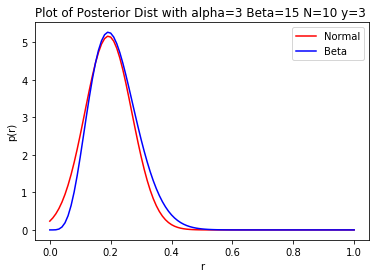


Figure 2 Comparing Normal and Beta Dist. for posterior for state 2

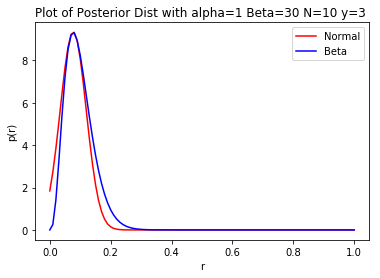


Figure 3 Comparing Normal and Beta Dist. for posterior for state 3

The developed script is as follow:

# -\*- coding: utf-8 -\*-

"""

Created on Sun Nov 5 09:53:08 2017

@author: smvaz

"""

def clear\_all():

"""Clears all the variables from the workspace of the spyder application."""

gl = globals().copy()

for var in gl:

if var[0] == '\_': continue

if 'func' in str(globals()[var]): continue

if 'module' in str(globals()[var]): continue

del globals()[var]

if \_\_name\_\_ == "\_\_main\_\_":

clear\_all()

import numpy as np

import matplotlib.pyplot as plt

import scipy.stats as stats

#defining each of three states

for i in range(0,3):

if i==0:

Alpha=5

Beta=5

N=20

y=10

if i==1:

Alpha=3

Beta=15

N=10

y=3

if i==2:

Alpha=1

Beta=30

N=10

y=3

#caclulating Variance (Standard Deviation) and mean based on previous problem

Variance=(y+Alpha-1)\*(N-y+Beta-1)/(Alpha+N+Beta-2)\*\*2/(Alpha+N+Beta-2)

print( 'Variance', Variance)

SD=Variance\*\*0.5

r=(y+Alpha-1)/(Alpha+N+Beta-2)

print('r',r)

R=np.linspace(0, 1, 101)

#drawing the figures

plt.figure()

plt.plot(R,stats.norm.pdf(R, loc=r, scale=SD),color='r',label='Normal')

plt.plot(R,stats.beta.pdf(R, Alpha, Beta),color='b',label='Beta')

plt.xlabel('r')

plt.ylabel('p(r)')

plt.legend()

ti = 'Plot of Posterior Dist with alpha={} Beta={} N={} y={} '.format(Alpha,Beta,N,y)

plt.title(ti)

plt.show()

# Problem 4

In order to estimate the value of we generate a 2D random numbers. Actually, we fill a square with total area of 4 with dots. As we know the formula for distance is and if we consider the center as the second point it changes to . If the distance of point to the center in that square is more than, it means it can be confined by a circle with radius of 1. As the following figure shows, the red dots are the dots with the distance to the center less than one, <1.

Table 2 Comparing the influence of different values of iterations

|  |  |
| --- | --- |
| Total Number of the Points | Calculated PI |
| 1e7 | 3.142661 |
| 1e6 | 3.142660 |
| 1e5 | 3.141640 |
| 1e4 | 3.143600 |
| 1e3 | 3.128000 |
| 1e2 | 2.920000 |

As it can be seen, increasing the number of the total points reduces the error compared to the exact number.

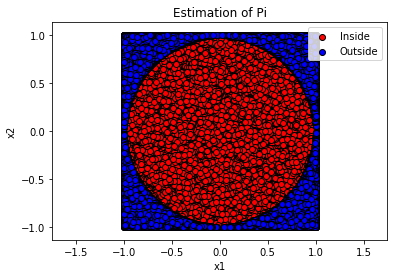


Figure 4Dots inside and outside of the circle

The script associated with this problem is:

# -\*- coding: utf-8 -\*-

"""

Created on Sat Nov 4 14:45:00 2017

@author: smvaz

"""

def clear\_all():

"""Clears all the variables from the workspace of the spyder application."""

gl = globals().copy()

for var in gl:

if var[0] == '\_': continue

if 'func' in str(globals()[var]): continue

if 'module' in str(globals()[var]): continue

del globals()[var]

if \_\_name\_\_ == "\_\_main\_\_":

clear\_all()

import numpy as np

import matplotlib.pyplot as plt

a=10000000

np.random.seed(seed=1)

Random=np.random.uniform(-1,1,(a,2))

Inside=[0,0]

Outside=[0,0]

print('All sample pairs \n{}'.format(Random))

for j in range(0,a):

Radius=(Random[j,0]\*\*2+Random[j,1]\*\*2)

if Radius<1:

#print('Inside point \n {} {}'.format(Random[j,:],Radius ))

Inside=np.vstack((Inside,Random[j,:]))

#print('Inside \n {}'.format(Inside))

else:

#print('Outside point \n {} {}'.format(Random[j,:],a ))

Outside=np.vstack((Outside,Random[j,:]))

#Outside=(Random[j,:])

Inside=np.delete(Inside,0,0)

print('Inside \n {}'.format(Inside))

Outside=np.delete(Outside,0,0)

print('Outside \n {}'.format(Outside))

#print(Inside.shape[0])

NumberofInside=Inside.shape[0]

NumberofOutside=Outside.shape[0]

PI=NumberofInside/(NumberofInside+NumberofOutside)

print('Calculated value for Pi is {:01.6f}'.format(4\*PI))

plt.figure()

plt.scatter(Inside[:,0], Inside[:,1], color='r', edgecolor='k',label='Inside')

plt.scatter(Outside[:,0], Outside[:,1], color='b', edgecolor='k',label='Outside')

plt.xlabel('x1')

plt.ylabel('x2')

plt.legend()

plt.title('Estimation of Pi')

plt.axis('equal')

plt.show()

# Problem 5



P=

)=

)=

So, we have for first and second derivative to w:

First derivative:



Second derivate (hessian Matrix)



Newtom-Raphson:

()

Widrow\_hoff

:()