HW3. You need to read Bayesian update before you can answer problem 1.

1. (30%)

In this problem, you are asked to add one Bin to the Bayesain update problem.

In total, there are three possible bins with probability

P(red |H=A) =0.7

P(green |H=A) =0.3

P(red |H=B) = 0.3

P(green |H=B) = 0.7

P(red |H=C) = 0.1

P(green |H=C) = 0.9

We start with the assumption that the prior probability is given by

P(H=A) =1/3

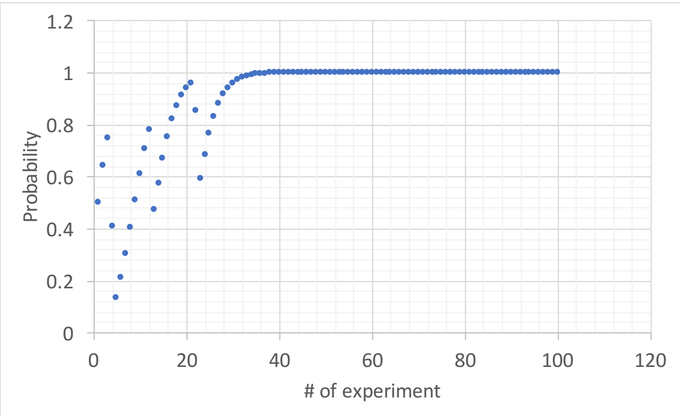
P(H=B)= 1/3

P(H=C)= 1/3

Now for simulation, we place Bin C in the black box and start to draw ball from the Bin C. Please run the Bayesian update for 100 iteration and obtain the posterior probability P(H=C | data) (data can be red or green) or equivalently the new prior P(H=C) because you need to reset the prior with posterior and update the prior probability accordingly. Plot it in excel or other grapher software.

Solution:

```python
import random

#initial prior probabity
priorA = 1/3    #prior probability
priorB = 1/3    # prior proability
priorC = 1/3    # prior proability

Pgreen_A= 4/10   # Probability of green given Bin A
Pred_A=6/10      # Probability of red given Bin A
Pgreen_B= 7/10   # Probability of green given Bin B
Pred_B= 3/10     # Probability of red given Bin B
Pgreen_C= 1/10   # Probability of green given Bin C
Pred_C= 9/10     # Probability of red given Bin C
#define these probabity and set to zero for convenience
P_A_red=0
P_A_green=0
P_B_red=0
P_B_green=0
P_C_red=0
P_C_green=0
num_seq=100

for seq in range(num_seq):
    x=random.uniform(0,1)

    if x>=0 and x<=Pgreen_C:  # need to modify here to place BIN C
        P_A_green= Pgreen_A*priorA/(Pgreen_A*priorA+Pgreen_B*priorB+Pgreen_C*priorC)
        P_B_green= Pgreen_B*priorB/(Pgreen_A*priorA+Pgreen_B*priorB+Pgreen_C*priorC)
        P_C_green= Pgreen_C*priorC/(Pgreen_A*priorA+Pgreen_B*priorB+Pgreen_C*priorC)
        priorA=P_A_green
        priorB=P_B_green
        priorC=P_C_green
        print( 'green', seq+1, round(P_C_green,4))

    else:
        P_A_red= Pred_A*priorA/(Pred_A*priorA+Pred_B*priorB+Pred_C*priorC)
        P_B_red= Pred_B*priorB/(Pred_A*priorA+Pred_B*priorB+Pred_C*priorC)
        P_C_red= Pred_C*priorC/(Pred_A*priorA+Pred_B*priorB+Pred_C*priorC)
        priorA=P_A_red
        priorB=P_B_red
        priorC=P_C_red
        print( 'red', seq+1, round(P_C_red,4))
```



3. A random sampling algorithm for quadratic programming

Consider the following quadratic programming problem

Objective function

Z= $(\boldsymbol{w}-\boldsymbol{b})^T(\boldsymbol{w}-\boldsymbol{b})$

The feasible region is cube of dimension d defined by

$0 \le w_i \le 1$ i=1,2,3.....d

(a) (30%) First, we set d=3 and consider a 3D problem with :

$$\boldsymbol{w} = (w_1, w_{2,} w_3)$$

$$\boldsymbol{b} = (3, 1/2, 1/2)$$

Now randomly and uniformly sample the feasible region for 10^6 sample point and find both *the minimal value of Z* and $\boldsymbol{w}$ such that Z(**w**) is the minimum of Z. Usually we write it as

$$\boldsymbol{w} = \operatorname{argmin}_{\boldsymbol{w} \in \text{Cube}} \boldsymbol{Z}$$

Repeat the experiment for 10 times and for each run, print out both minimal value of Z and the corresponding w. (Print out both the code and result.)

(b) (10%) Is the answer you find in close proximity to the corner point of cube? Corner point (0,0,0) (1,0,0) (0,1,0) (1,1,1) (0,1,0) (1,1,0).....

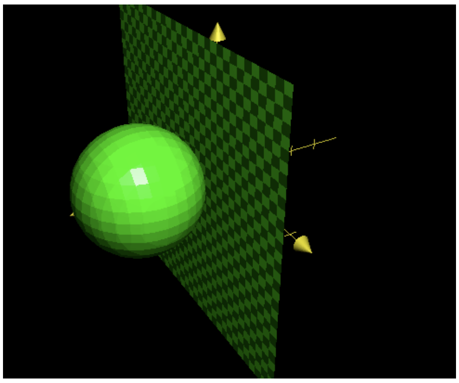
Can you justify $\boldsymbol{w}$ using geometrical interpretation?

Solution:

```
import random
numNeedles=1000000 #use 1 million sample for each run
MIN=1000
iteration= 10
Z_current=0.0

b1=3
b2=1/2
b3=1/2
w1_min=0.0
w2_min=0.0
w3_min=0.0
# z is the obective fuction
def throwNeedles(numNeedles):
    Z_min=1000
    w1_min_local=0.0
    w2_min_local=0.0
    w3_min_local=0.0
    inSphere=0
    for Needles in range(1, numNeedles+1):
        w1= random.uniform(0,1)
        w2= random.uniform(0,1)
        w3= random.uniform(0,1)
        Z_current=(w1-b1)*(w1-b1)+(w2-b2)*(w2-b2)+(w3-b3)*(w3-b3)
        if Z_current <= Z_min:
            Z_min=Z_current
            w1_min_local=w1
            w2_min_local=w2
            w3_min_local=w3
    return Z_min,w1_min_local,w2_min_local,w3_min_local
for iteration in range(1,11):
    MIN, w1_min, w2_min, w3_min =throwNeedles(numNeedles)
    print('Est =',iteration, round(MIN,3), round(w1_min,3),round(w2_min,3),round(w3_min,3))
```

```
Est = 1 4.002 1.0 0.514 0.514
Est = 2 4.002 0.999 0.501 0.496
Est = 3 4.001 1.0 0.51 0.466
Est = 4 4.002 1.0 0.512 0.511
Est = 5 4.002 1.0 0.486 0.475
Est = 6 4.002 1.0 0.492 0.476
Est = 7 4.001 1.0 0.499 0.529
Est = 8 4.001 1.0 0.509 0.485
Est = 9 4.002 1.0 0.492 0.522
Est = 10 4.001 1.0 0.488 0.5
```

The w is not the corner point but it is on the face of the cube. The geometric reason is that the sphere touches the cube at this point and the plane corresponding to the face of the cube is tangent to the sphere. Below I plot the sphere and the plane X=1, which is the face of the cube.



(c) (30%) Now we change the problem to 5 dimension (d=5). Hence w is a five dimensional vector

$$\boldsymbol{w} = (w_1, w_{2,} w_3, w_4, w_5)$$

and set

$$\boldsymbol{b} = (3, 1/2, 1/2, 1/2, 1/2)$$

Now randomly sample the feasible region for 10^6 and find both $\boldsymbol{w}$ such that $Z(\boldsymbol{w})$ is minimum and the minimum of Z.

Repeat the experiment for 10 times and for each run, print out both minimal value of Z and the corresponding w. (Print out both the code and result.)

Solution: I run the experiment 10 times. You can need to run 1 time. OK.

```
import random
numNeedles=1000000 #use 1 million sample for each run
MIN=1000
iteration= 10
Z_current=0.0

b1=3
b2=1/2
b3=1/2
b4=1/2
b5=1/2
w1_min=0.0
w2_min=0.0
w3_min=0.0
w4_min=0.0
w5_min=0.0
# z is the obective fuction
def throwNeedles(numNeedles):
    Z_min=1000
    w1_min_local=0.0
    w2_min_local=0.0
    w3_min_local=0.0
    inSphere=0
    for Needles in range(1, numNeedles+1):
        w1= random.uniform(0,1)
        w2= random.uniform(0,1)
        w3= random.uniform(0,1)
        w4= random.uniform(0,1)
        w5= random.uniform(0,1)
        Z_current=(w1-b1)*(w1-b1)+(w2-b2)*(w2-b2)+(w3-b3)*(w3-b3)+(w4-b4)*(w4-b4)+(w5-b5)*(w5-b5)
        if Z_current <= Z_min:
            Z_min=Z_current
            w1_min_local=w1
            w2_min_local=w2
            w3_min_local=w3
            w4_min_local=w4
            w5_min_local=w5
    return Z_min,w1_min_local,w2_min_local,w3_min_local, w4_min_local, w5_min_local
for iteration in range(1,11):
    MIN, w1_min, w2_min, w3_min, w4_min , w5_min=throwNeedles(numNeedles)
    print('Est =',iteration, round(MIN,3), round(w1_min,3),round(w2_min,3),round(w3_min,3), round(w4_min,3), round(w5_min,3))
```

```
Est = 1 4.007 0.999 0.447 0.512 0.492 0.516
Est = 2 4.003 1.0 0.462 0.494 0.5 0.532
Est = 3 4.01 1.0 0.499 0.411 0.462 0.514
Est = 4 4.017 0.996 0.493 0.502 0.466 0.485
Est = 5 4.009 0.999 0.488 0.488 0.489 0.563
Est = 6 4.007 0.999 0.514 0.509 0.455 0.467
Est = 7 4.018 0.999 0.479 0.592 0.539 0.44
Est = 8 4.013 0.999 0.516 0.58 0.482 0.528
Est = 9 4.018 0.999 0.446 0.407 0.484 0.457
Est = 10 4.018 1.0 0.533 0.486 0.615 0.458
```

The minimal value of Z is 4.018 and the w is (1.0, 0.533, 0.486, 0.615, 0.458).

This point is close to the center of one face of cube. Interestingly, the minimal value of Z does not change with dimension.