GMDL212, HW2

Yuval Margalit

problem1:

הבאג שבו הסטודנט נתקל הינו trunction כתוצאה מכך בשפייתון 2 חילוק בין 2 שלמים מחזיר אך ורק את החלק השלם
של התוצאה. בניגוד לפייתון 3 שבו מוחזר float בצורה דיפולטיבית לחילוק, כתוצאה מכך הסטודנט דגם מההתפלגות האחידה על כל התומונת הבינאריות
בגודל 8×8.

problem 2:

התוצאות הן:

temp	$\hat{E_{temp}}(X_{1,1}, X_{2,2})$	$\hat{E_{temp}}(X_{1,1}, X_{8,8})$
1	0.9506	0.9066
1.5	0.7692	0.5462
2	0.5068	0.1312

ניתן לראות כי בכל הטמפרטורות ככל שהמרחק בין שני הפקיסלים גדל כך גם הסיכוי שיקבלו ערכים שונים גדל.
כאשר הטמפרטורה הינה 1 ההתפלגות נוטה לטובת דגימות אחידות, כלומר הסיכוי לקבל שני פיקסלים שכנים
בעלי ערך זהה הינו גבוהה מאוד ולכן גם במרחק ניכר בין הפיקסלים הסיכוי שהם יהיו בעלי אותו ערך נשמר גבוהה.
כאשר הטמפרטורה הינה 1.5 הסיכוי לקבל ערך שונה בשני פיקסלים קרובים עולה משמעותית ולכן כבר במרחק של 2
בשריג אנו מקבלים כי מספר הדגימות שבו קיבלו את אותה הערך ב$X_{1,1}$ו $X_{2,2}$קטן משמעותית.
באופן דומה התופעה נהיית חזקה אף יותר כאשר הטמפרטורה הינה 2 ואנו רואים שהסיכוי של שני פקיסלים במרחק 2 להיות
בעלי אותו ערך הוא כ$\frac{3}{4}$, באופן ישיר כתוצאה מכך ככל שהמרחק עולה הסיכוי שלהם להיות זהים קטן משמעותית והינו כמעט חצי.

problem 3:

תוצאות שיטה 1 (דגימות בלתי תלויות):

temp	$\hat{E}_{temp}(X_{1,1}, X_{2,2})$	$\hat{E}_{temp}(X_{1,1}, X_{8,8})$
1	0.9332	0.5638
1.5	0.7304	0.3588
2	0.5032	0.0950

תוצאות שיטה 2 (ארגודיות):

temp	$\hat{E}_{temp}(X_{1,1}, X_{2,2})$	$\hat{E}_{temp}(X_{1,1}, X_{8,8})$
1	0.9500	0.90157
1.5	0.7647	0.5316
2	0.5089	0.1287

נשים לב כי ההפרשים בין שתי השיטות הינם גדולים בתוחלת של $X_{1,1}, X_{8,8}$ בכל הטמפרטורות וקיים גם
הפרש ניכר בין השיטה הראשונה (דגימות בלתי תלויות) לבין התוחלת האמפירית שמדדנו,
אני משער שהפרש זה נובע מהעובדה כי 25 איטרציות אינן מספיקות בשביל שהסתברות הדגימה תתכנס להסתברות האמיתית
ובעקבות חוסר ההתכנסות לא מתבצע פעפוע של תוצאת הדגימה ב$X_{1,1}$אל עבר $X_{8,8}$.
כאשר הטמפ' היא 2 גם ההסתברות האמיתית של ערכים זהים ב $X_{1,1}, X_{8,8}$ מתקרבת לחצי ולכן ההפרש קטן יותר.
בנוסף ניתן לראות כי השיטה הארגודית קרובה להתכנסות ואולי אפילו מתכנסת להסתברות האמיתי כיוון והתוצאות כמעט זהות.

Computer Exercises:

HW2 notebook

April 10, 2021

[1]:
```
#Computer Exercise 1
import numpy as np
def G(row_s:np.ndarray,temp:float)->float:
    return np.exp(np.dot(row_s,np.pad(row_s[1:],(0,1)))/temp)
```

[2]:
```
#Computer Exercise 2
def F(row_s:np.ndarray,row_t:np.ndarray,temp:float)->float:
    return np.exp((np.dot(row_s,row_t))/temp)
```

[3]:
```
#Computer Exercise 3
import itertools
temps=np.array([1,1.5,2])
Ztemp=np.zeros(3)
i=0
for temp in temps:
    for prod in itertools.product([-1,1],repeat=4):
        grid=np.array(prod).reshape((2,2))
        Ztemp[i]+=G(grid[0],temp)*G(grid[1],temp)*F(grid[0],grid[1],temp)
    i+=1

Ztemp
```

[3]:
```
array([121.23293134,  40.92279909,  27.04878276])
```

[4]:
```
#Computer Exercise 4
import itertools
Ztemp=np.zeros(3)
i=0
for temp in temps:
    for prod in itertools.product([-1,1],repeat=9):
        grid=np.array(prod).reshape((3,3))
        Ztemp[i]+=G(grid[0],temp)*G(grid[1],temp)*G(grid[2],temp)*\
        F(grid[0],grid[1],temp)*F(grid[1],grid[2],temp)
    i+=1

Ztemp
```

[4]:
```
array([365645.74913577,  10565.42198351,   2674.51812306])
```

[5]:
```
def y2row(y,width=8):
    """
    y: an integer in (0,...,(2**width)-1)
    """
    if not 0<=y<=(2**width)-1:
        raise ValueError(y)
    my_str=np.binary_repr(y,width=width)
    # my_list = map(int,my_str) # Python 2
    my_list = list(map(int,my_str)) # Python 3
    my_array = np.asarray(my_list)
    my_array[my_array==0]=-1
    row=my_array
    return row
```

[6]:
```
#Computer Exercise 5
Ztemp=np.zeros(3)
i=0
for temp in temps:
    for y_1 in range(4):
        for y_2 in range(4):
            Ztemp[i]+=G(y2row(y_1,width=2),temp)*G(y2row(y_2,width=2),temp)*\
            F(y2row(y_1,width=2),y2row(y_2,width=2),temp)
    i+=1

Ztemp
```

[6]: array([121.23293134, 40.92279909, 27.04878276])

[7]:
```
#Computer Exercise 6
Ztemp=np.zeros(3)
i=0
for temp in temps:
    for y_1 in range(8):
        for y_2 in range(8):
            for y_3 in range(8):
                Ztemp[i]+=G(y2row(y_1,width=3),temp)*G(y2row(y_2,width=3),temp)*\
                G(y2row(y_3,width=3),temp)*\
                F(y2row(y_1,width=3),y2row(y_2,width=3),temp)*\
                F(y2row(y_3,width=3),y2row(y_2,width=3),temp)
    i+=1
Ztemp
```

[7]: array([365645.74913577, 10565.42198351,   2674.51812306])

[8]:
```
#Computer Exercise 7
def forwardpass(temp:float,width=8):
    ts=[]
```

```
    length=2**width
    t1=np.zeros(length)
    """
    claculatin T1(y2)
    """
    for i in range(length):
        for j in range(length):
            t1[i]+=G(y2row(j,width),temp)*F(y2row(j,width),y2row(i,width),temp)
    ts.append(t1)
    """
    claculatin Tk(yk+1);k<width
    """
    for k in range (1,width -1):
        t=np.zeros(length)
        for i in range(length):
            for j in range(length):
                t[i]+=ts[k-1][j]*G(y2row(j,width),temp)*\
                F(y2row(j,width),y2row(i,width),temp)
        ts.append(t)
    t_last=0
    """
    claculatin Tk(yk);k=width
    """
    for i in range(length):
            t_last+=ts[width-2][i]*G(y2row(i,width),temp)
    ts.append(t_last)
    return ts
```

[9]:
```
def compute_probs(temp:float,ts:list,width=8):
    ps=[]
    length=2**width
    p_last=np.zeros(length)
    """
    calculta pk(yk);k=width
    """
    for i in range (length):
        p_last[i]=(ts[width-2][i]*G(y2row(i,width),temp))/ts[width-1]
    ps.append(p_last)
    """
    calculta p(k|k+1)(yk|yk+1);2<=k<width
    """
    for k in range (width-1,1,-1):
        pk=np.zeros((length,length))
        for i in range(length):
            for j in range(length):
```

```
                pk[i][j]=(ts[k-2][j]*G(y2row(j,width),temp)*\
                          F(y2row(j,width),y2row(i,width),temp))/ts[k-1][i]
        ps.insert(0,pk)
    """
    calculat p1/2
    """
    p_last=np.zeros((length,length))
    for i in range(length):
            for j in range(length):
                p_last[i][j]=G(y2row(j,width),temp)*\
                F(y2row(j,width),y2row(i,width),temp)/ts[0][i]
    ps.insert(0,p_last)
    return ps
```

[10]:
```
def sample_pic(ps,width=8):
    length=2**width
    values=np.array(range(2**width))
    ys=np.zeros((width,width))
    yk2=np.random.choice(values,p=ps[width-1])
    ys[7]=y2row(yk2,width)
    for k in range (width-1,0,-1):
        yk1=np.random.choice(values,p=ps[k-1][yk2])
        ys[k-1]=y2row(yk1,width)
        yk2=yk1
    return ys
```

[11]:
```
def sampler(temp):
    ts=forwardpass(temp)
    ps=compute_probs(temp,ts)
    return lambda :sample_pic(ps)
```

[12]:
```
sampler1=sampler(1)
sampler15=sampler(1.5)
sampler2=sampler(2)
```

[13]:
```
import pylab
pylab.ion()
from pylab import plt
fig, axes = plt.subplots(nrows=3, ncols=10, figsize=(20, 20))
rows = ['temp {}'.format(row) for row in [1, 1.5, 2]]
for ax, row in zip(axes[:,0], rows):
    ax.set_ylabel("%s"%row, rotation=0, size='large')
index=1
for sampler_t in [sampler1,sampler15,sampler2]:
```

```
    for i in range(10):
        plt.subplot(3,10,index)
        plt.imshow(sampler_t(),interpolation="None",vmin=-1,vmax=+1)
        plt.axis('off')
        plt.title("temp =%s"%temps[(index-1)//10])
        index+=1
```

[14]:
```
#Computer Exercise
import pandas as pd
results=pd.DataFrame(index=pd.Series(temps))
results.index.name='temperature'
j=0
for sampler_t in [sampler1,sampler15,sampler2]:
    sum=np.zeros(2)
    for i in range(10000):
        pic=sampler_t()
        sum[0]+=pic[0][0]*pic[1][1]
        sum[1]+=pic[0][0]*pic[7][7]
    sum=sum/10000
```

```
    results.loc[temps[j],"$$\hat{E}_{Temp}(X_{(1,1)}X_{(2,2)})$$"]=sum[0]
    results.loc[temps[j],"$$\hat{E}_{Temp}(X_{(1,1)}X_{(8,8)})$$"]=sum[1]
    j+=1
results
```

temperature	$\hat{E}_{Temp}(X_{(1,1)}X_{(2,2)})$	$\hat{E}_{Temp}(X_{(1,1)}X_{(8,8)})$
1.0	0.9508	0.9046
1.5	0.7674	0.5448
2.0	0.5066	0.1080

[15]:
```
#Computer Exercise 9
def single_chain(size:int,iterations:int,temp:float)->np.ndarray:
    sample=np.random.randint(low=0,high=2,size=(size+2,size+2))*2-1
    sample[0]=0
    sample[size+1]=0
    sample[:,0]=0
    sample[:,size+1]=0
    for k in range(iterations):
        for i in range(1,size+1):
            for j in range(1,size+1):
                nsum=sample[i-1][j]+sample[i+1][j]+sample[i][j-1]+sample[i][j+1]
                prob=np.array([np.exp(nsum/temp),np.exp(-nsum/temp)])
                prob=prob/np.sum(prob)
                sample[i][j]=np.random.choice([1,-1],p=prob)
    sample=np.delete(sample,[0,size+1],1)
    sample=np.delete(sample,[0,size+1],0)
    return sample
```

[16]:
```
for temp in [1,1.5,2]:
    sum=np.zeros(2)
    for i in range(10000):
        sample=single_chain(8,25,temp)
        sum[0]+=sample[0][0]*sample[1][1]
        sum[1]+=sample[0][0]*sample[7][7]
    sum=sum/10000
    results.loc[temp,"$$\hat{E}_{Temp}(X_{(1,1)}X_{(2,2)})$$"]=sum[0]
    results.loc[temp,"$$\hat{E}_{Temp}(X_{(1,1)}X_{(8,8)})$$"]=sum[1]
results
```

temperature	$\hat{E}_{Temp}(X_{(1,1)}X_{(2,2)})$	$\hat{E}_{Temp}(X_{(1,1)}X_{(8,8)})$
1.0	0.9332	0.5638
1.5	0.7304	0.7340
2.0	0.5032	0.0950

[17]:
```
def method_2(size:int,temp:float):
    sample=np.random.randint(low=0,high=2,size=(size+2,size+2))*2-1
    sample[0]=0
    sample[size+1]=0
    sample[:,0]=0
    sample[:,size+1]=0
    avg=np.zeros(2)
    for k in range(25000):
        for i in range(1,size+1):
            for j in range(1,size+1):
                nsum=sample[i-1][j]+sample[i+1][j]+sample[i][j-1]+sample[i][j+1]
                prob=np.array([np.exp(nsum/temp),np.exp(-nsum/temp)])
                prob=prob/np.sum(prob)
                sample[i][j]=np.random.choice([1,-1],p=prob)
        if k>=100:
            avg[0]=((k-99)*avg[0]+sample[1][1]*sample[2][2])/(k-98)
            avg[1]=((k-99)*avg[1]+sample[1][1]*sample[8][8])/(k-98)
        elif k==99:
            avg[0]=sample[0][0]*sample[1][1]
            avg[1]=sample[0][0]*sample[7][7]
    return avg
```

[18]:
```
for temp in [1,1.5,2]:
    avg=method_2(8,temp)
    results.loc[temp,"$$\hat{E}_{Temp}(X_{(1,1)}X_{(2,2)})$$"]=avg[0]
    results.loc[temp,"$$\hat{E}_{Temp}(X_{(1,1)}X_{(8,8)})$$"]=avg[1]
results
```

temperature	$\hat{E}_{Temp}(X_{(1,1)}X_{(2,2)})$	$\hat{E}_{Temp}(X_{(1,1)}X_{(8,8)})$
1.0	0.950002	0.901570
1.5	0.764789	0.531625
2.0	0.508895	0.128670

[26]:
```
#Computer Exercise 10 c
def post_sampler(temp:float,sigma_sq:float,y:np.ndarray,iterations:int,size:int):
    sample=np.random.randint(low=0,high=2,size=(size+2,size+2))*2-1
    sample[0]=0
    sample[size+1]=0
    sample[:,0]=0
    sample[:,size+1]=0
    for k in range(iterations):
        for i in range(1,size+1):
            for j in range(1,size+1):
                nsum=sample[i-1][j]+sample[i+1][j]+sample[i][j-1]+sample[i][j+1]
                nsum=sample[i-1][j]+sample[i+1][j]+sample[i][j-1]+sample[i][j+1]
```

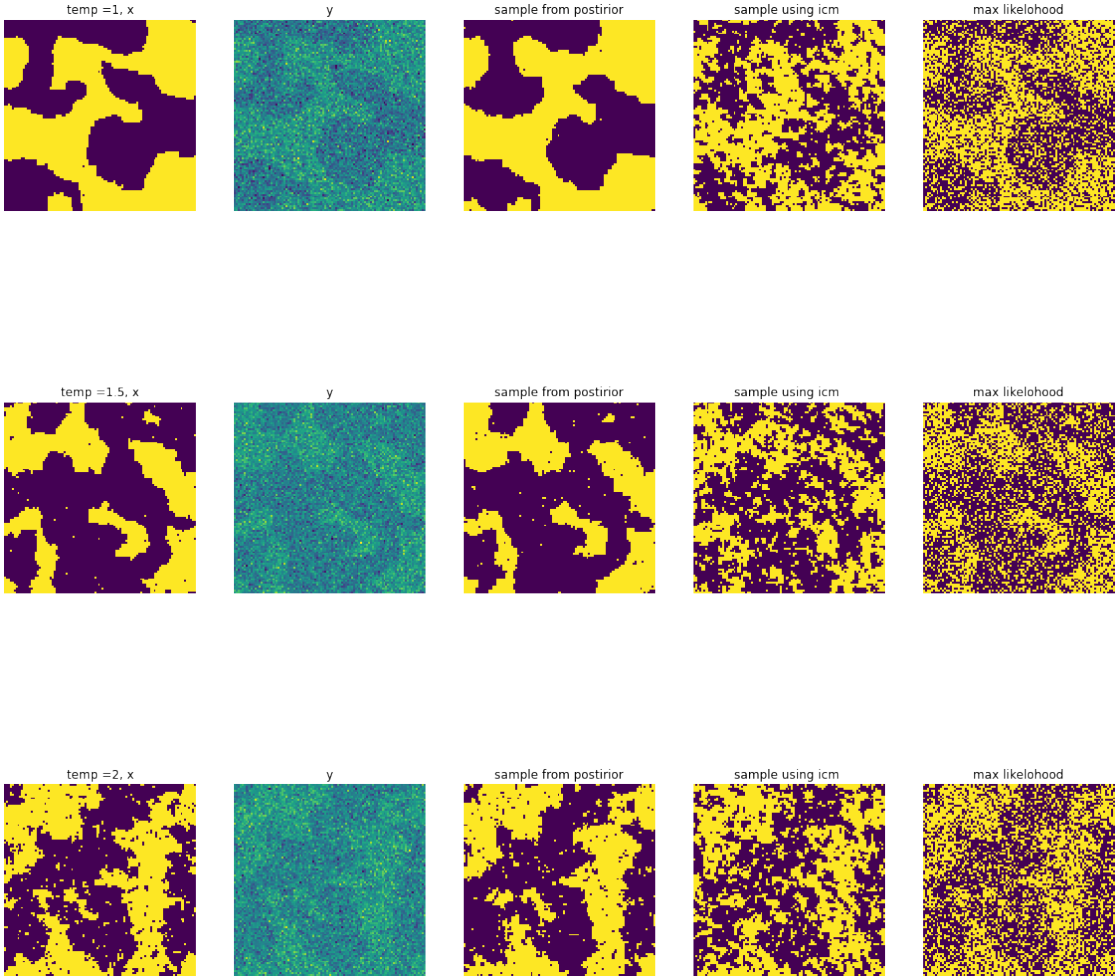
```
                prob=np.array([np.exp(nsum/temp-(y[i-1][j-1]-1)**2/
↪(2*sigma_sq)),#-1 because y is size*size
                               np.exp(-nsum/temp-(y[i-1][j-1]+1)**2/
↪(2*sigma_sq))])
                prob=prob/np.sum(prob)
                sample[i][j]=np.random.choice([1,-1],p=prob)
    sample=np.delete(sample,[0,size+1],1)
    sample=np.delete(sample,[0,size+1],0)
    return sample
```

[27]:
```
#Computer Exercise 10 e
def sign(y:np.ndarray)->np.ndarray:
    shape=y.shape
    mat=np.zeros(shape)
    for i in range(shape[0]):
        for j in range(shape[1]):
            if y[i][j]>0:
                mat[i][j]=1
            else:
                mat[i][j]=(-1)
    return mat
```

[28]:
```
#Computer Exercise 10 d
def icm(temp:float,sigma_sq:float,y:np.ndarray):
    sample=np.random.randint(low=0,high=2,size=(100+2,100+2))*2-1
    sample[0]=0
    sample[100+1]=0
    sample[:,0]=0
    sample[:,100+1]=0
    while True:
        prev_sample=sample
        for i in range(1,100+1):
            for j in range(1,100+1):
                nsum=sample[i-1][j]+sample[i+1][j]+sample[i][j-1]+sample[i][j+1]
                if (nsum/temp-(y[i-1][j-1]-1)**2/(2*sigma_sq))>=(-nsum/
↪temp-(y[i-1][j-1]+1)**2/(2*sigma_sq)):
                    sample[i][j]=1
                else:
                    sample[i][j]=-1
        if np.array_equal(prev_sample,sample):
            sample=np.delete(sample,[0,100+1],1)
            sample=np.delete(sample,[0,100+1],0)
            return sample
```

[29]:
```
#Computer Exercise 10
sigma_sq=4
fig, axes = plt.subplots(nrows=3, ncols=5,figsize=(20, 20))
```

```
index=1
for temp in [1,1.5,2]:
    x=single_chain(100,50,temp)
    plt.subplot(3,5,index)
    plt.imshow(x,interpolation="None",vmin=-1, vmax=+1)
    plt.axis('off')
    plt.title("temp =%s, x"%temp)
    index+=1
    eta = 2*np.random.standard_normal(size=(100,100))
    y = x + eta
    plt.subplot(3,5,index)
    plt.imshow(y,interpolation="None")
    plt.axis('off')
    plt.title("y")
    index+=1
    post_sample=post_sampler(temp,sigma_sq,y,50,100)
    plt.subplot(3,5,index)
    plt.imshow(post_sample,interpolation="None",vmin=-1, vmax=+1)
    plt.axis('off')
    plt.title("sample from postirior")
    index+=1
    icm_restoration=icm(temp,sigma_sq,y)
    plt.subplot(3,5,index)
    plt.imshow(icm_restoration,interpolation="None",vmin=-1, vmax=+1)
    plt.axis('off')
    plt.title("sample using icm ")
    index+=1
    max_likelihood=sign(y)
    plt.subplot(3,5,index)
    plt.imshow(max_likelihood,interpolation="None",vmin=-1, vmax=+1)
    plt.axis('off')
    plt.title("max likelohood")
    index+=1
```


temp =1, x
y
sample from postirior
sample using icm
max likelohood
temp =1.5, x
y
sample from postirior
sample using icm
max likelohood
temp =2, x
y
sample from postirior
sample using icm
max likelohood