نام درس : شناسایی آماری الگوریتم

تمرین شماره ۲

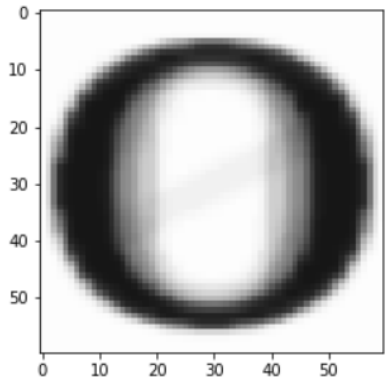
نام و شماره دانشجویی :فاطمه توکلی ، ۴۰۰۱۳۱۰۱۶

.۲

a. با استفاده از میانگین عکس ها هرکلاس را بازسازی کرده ونمایش می دهیم:

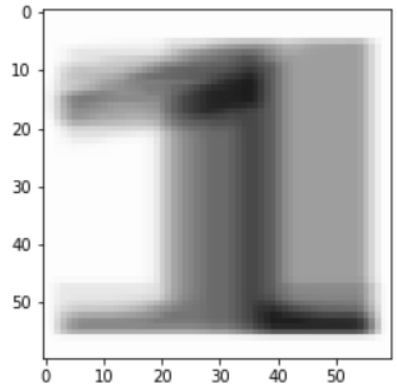
```
def get_photo(i):

    images = glob.glob (r"C:\\Users\\Fateme\\Desktop\\inputs\\P2\\dataset\\a\\Train\\/" + str(i) + '*')


    image_data = []
    for img in images:
        this_image = cv2.imread(img)
        image_data.append(this_image)

    avg_image = image_data[0]
    for i in range(len(image_data)):
        if i == 0:
            pass
        else:
            alpha = 1.0/(i + 1)
            beta = 1.0 - alpha
            avg_image = cv2.addWeighted(image_data[i], alpha, avg_image, beta, 0.0)

    cv2.imwrite('avg_zero.png', avg_image)
    plt.imshow(avg_image)
    plt.show()
    PHOTO_ARR = np.array(avg_image)
    return PHOTO_ARR
```

```
zero = get_photo(0)
cv2.imwrite('avg_zero.png', zero)
zero.mean()
```




```
170.12444444444444
```

```
one = get_photo(1)
cv2.imwrite('avg_one.png', one)
one.mean()
```




```
179.09166666666667
```

b.

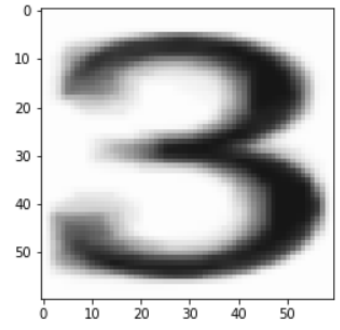
```
images = glob.glob (r"C:\\Users\\Fateme\\Desktop\\inputs\\P2\\dataset\\a\\Test\\/*")

test_data = []
test_label = []
for img in images:
    this_image = cv2.imread(img)
    test_data.append(this_image)

for i in range(0,10):
    test_label.append(0)
for i in range(10,20):
    test_label.append(1)

predict =[]
for i in range(len(test_data)):
    a = distance(test_data[i] , zero)
    b = distance(test_data[i], one)
    if a < b:
        predict.append(0)
    else:
        predict.append(1)
```

```
from sklearn.metrics import accuracy_score
score = accuracy_score(test_label, predict)
print('the accuracy of MDC : '+str(score))
```

```
the accuracy of MDC : 0.8
```

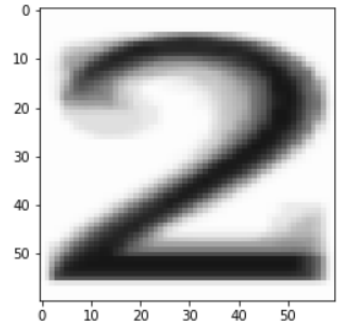
c.

```
tree = get_photo(3)
cv2.imwrite('avg_tree.png', tree)
tree.mean()
```



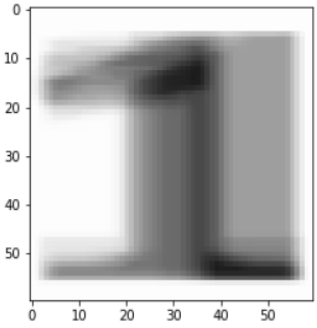
180.3336111111111

```
two = get_photo(2)
cv2.imwrite('avg_two.png', two)
two.mean()
```



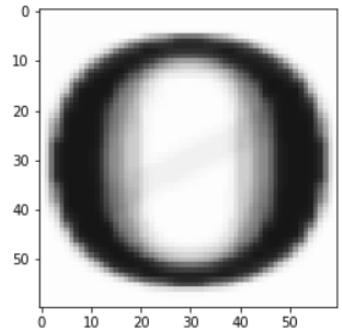
178.82805555555555

```
one = get_photo(1)
cv2.imwrite('avg_one.png', one)
one.mean()
```



179.09166666666667

```
zero = get_photo(0)
cv2.imwrite('avg_zero.png', zero)
zero.mean()
```



170.12444444444444

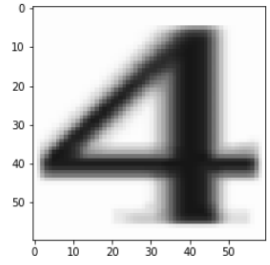
```
images = glob.glob (r"C:\\Users\\Fateme\\Desktop\\inputs\\P2\\dataset\\b\\Test\\/*")

test_data = []
test_label = []
for img in images:
    this_image = cv2.imread(img)
    test_data.append(this_image)

for i in range(0,10):
    test_label.append(0)
for i in range(10,20):
    test_label.append(1)
for i in range(20,30):
    test_label.append(2)
for i in range(30,40):
    test_label.append(3)

predict =[]
for i in range(len(test_data)):
    a = distance(test_data[i] , zero)
    b = distance(test_data[i], one)
    c = distance(test_data[i] , two)
    d = distance(test_data[i], tree)

    if min(a,b,c,d) == a:
        predict.append(0)
    elif min(a,b,c,d) == b:
        predict.append(1)
    elif min(a,b,c,d) == c:
        predict.append(2)
    else:
        predict.append(3)
```

```
predict = np.array(predict)
from sklearn.metrics import accuracy_score
score = accuracy_score(test_label, predict)
print('the accuracy of MDC : '+str(score))
```

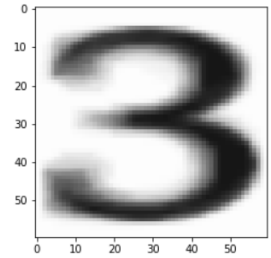
```
the accuracy of MDC : 0.65
```

d.

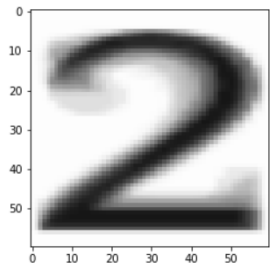
```
four = get_photo(4)
cv2.imwrite('avg_four.png', four)
four.mean()
```



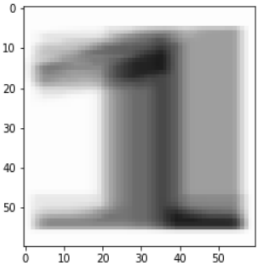
```
tree = get_photo(3)
cv2.imwrite('avg_tree.png', tree)
tree.mean()
```



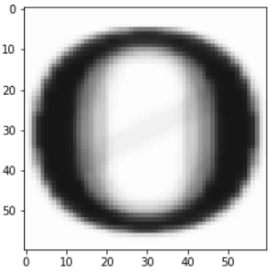
```
two = get_photo(2)
cv2.imwrite('avg_two.png', two)
two.mean()
```



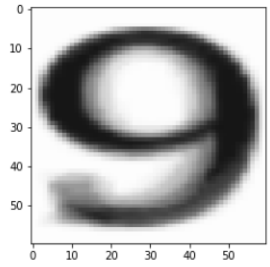
```
one = get_photo(1)
cv2.imwrite('avg_one.png', one)
one.mean()
```



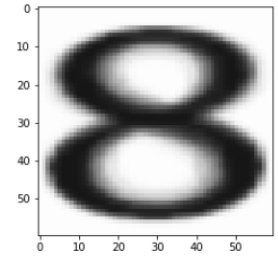
```
zero = get_photo(0)
cv2.imwrite('avg_zero.png', zero)
zero.mean()
```



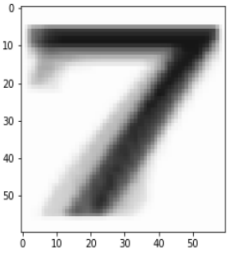
```
nine = get_photo(9)
cv2.imwrite('avg_nine.png', nine)
nine.mean()
z = nine
```



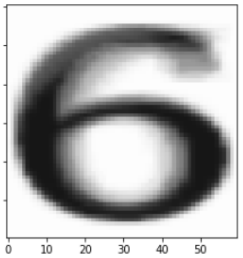
```
eight = get_photo(8)
cv2.imwrite('avg_eight.png', eight)
eight.mean()
w = eight
```



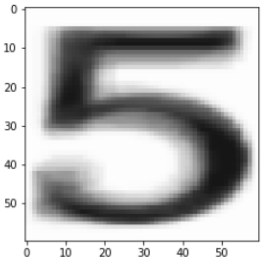
```
seven = get_photo(7)
cv2.imwrite('avg_seven.png', seven)
seven.mean()
```



```
six = get_photo(6)
cv2.imwrite('avg_six.png', six)
six.mean()
```



```
five = get_photo(5)
cv2.imwrite('avg_five.png', five)
five.mean()
```



```
from sklearn.metrics import accuracy_score
score = accuracy_score(test_label, predict)
print('the accuracy of MDC : '+str(score))
```

```
the accuracy of MDC : 0.41
```

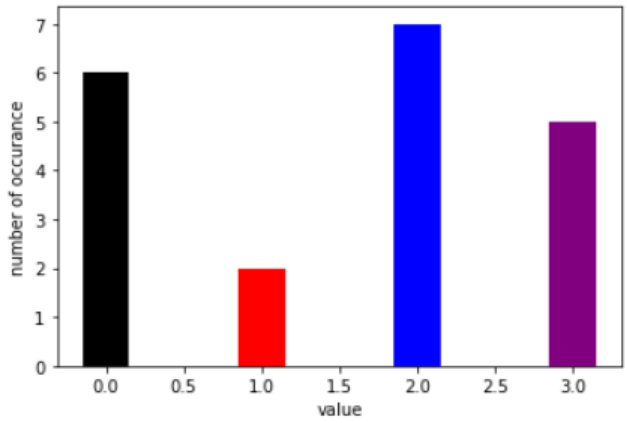
e. as the number of classes increases, our accuracy decreases and the minimum distance does not work well and the model get high bias

.2

A.

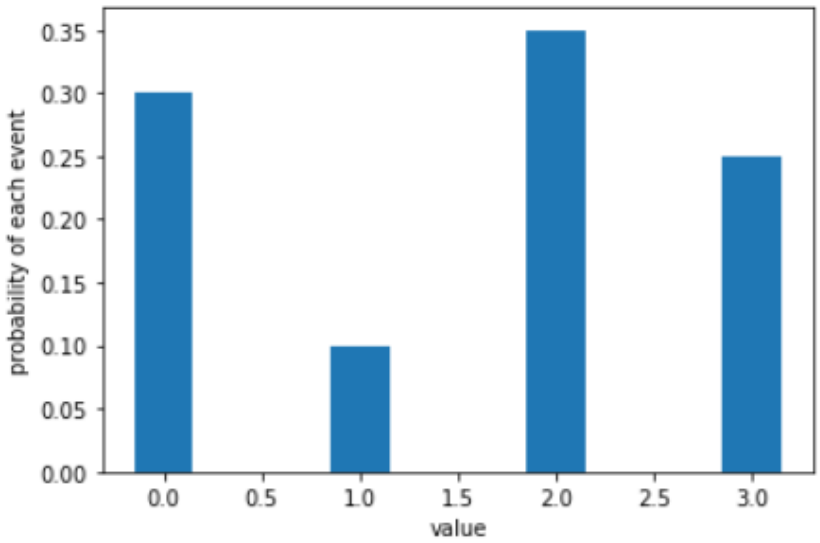
```
df = pd.DataFrame([0,2,0,3,1,2,2,2,0,2,3,1,0,0,2,3,3,2,3,0])

#bar plot
val, cnt = np.unique(df, return_counts=True)
plt.bar(val, cnt , width = 0.3 , color=['black', 'red', 'blue', 'purple'])
plt.xlabel('value')
plt.ylabel('number of occurance')
```

```
Text(0, 0.5, 'number of occurance')
```



B.

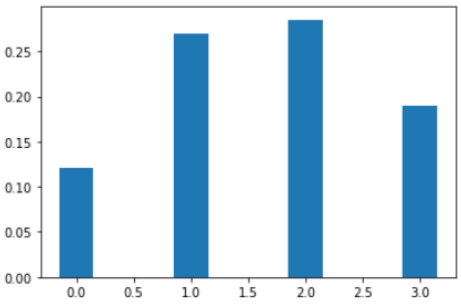
```
The probability of each number of success[0.3  0.1  0.35 0.25]
The normalize occurance of each value[0.85714286 0.28571429 1.         0.71428571]
```



C.

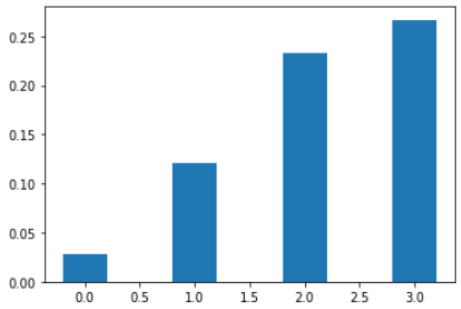
N= 20

```
n = 20
p = 0.1
r_values = [0,2,0,3,1,2,2,2,0,2,3,1,0,0,2,3,3,2,3,0]
dist = [binom.pmf(r, n, p) for r in r_values ]
plt.bar(r_values, dist ,width=0.3)
plt.show()
```



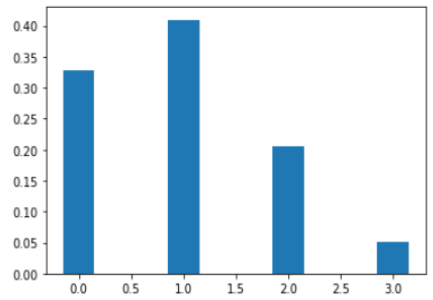
N=10

```
n = 10
p = 0.3
r_values = [0,2,0,3,1,2,2,2,0,2,3,1,0,0,2,3,3,2,3,0]
dist = [binom.pmf(r, n, p) for r in r_values ]
plt.bar(r_values, dist , width = 0.4)
plt.show()
```



N=5

```
n = 5
p = 0.2
r_values = [0,2,0,3,1,2,2,2,0,2,3,1,0,0,2,3,3,2,3,0]
dist = [binom.pmf(r, n, p) for r in r_values ]
plt.bar(r_values, dist , width = 0.3)
plt.show()
```



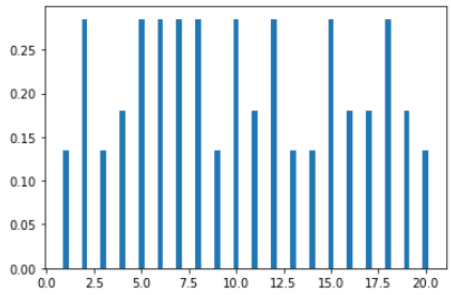
D.

سومی یعنی N=20 به احتمال بیشتری داده مطلوب مارو تولید میکند چون فرم نرمال گاوسی دارد

E.

```
# conside N = 5,20
r_result = []
N = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
for j in range(1,20):
    dist = [binom.pmf(r, j, p) for r in r_values ]
plt.bar(N,dist, width = 0.3)
```

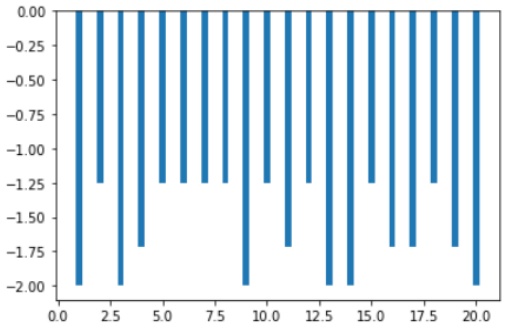
```
<BarContainer object of 20 artists>
```



F.

```
for j in range(1,20):
    dist = [binom.logpmf(r, j, p) for r in r_values ]
plt.bar(N,dist, width = 0.3)
```

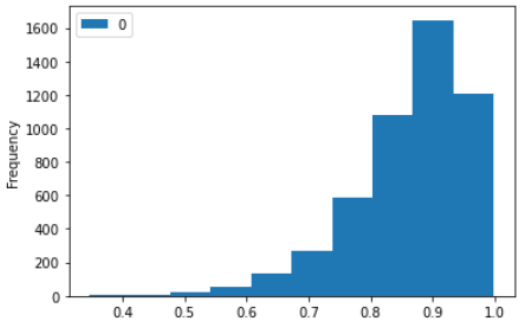
```
<BarContainer object of 20 artists>
```



G.

```
import pandas as pd
data = pd.read_csv('Kumaraswamy.txt', header = None)
data.plot.hist()
```

```
<AxesSubplot:ylabel='Frequency'>
```



5.

A,B.

میانگین و کواریانس را می یابیم و با استفاده از ان ها میتوانیم نمونه برداری انجام دهیم:

10 sample:

ERROR = 0.2

```
Mean of features :
[91.9, 69.5, 26.5, 21.8, 64.3, 3.85]
Variance Matrix features :
[[18602.6625]]
Mean feature012 =[91.9, 69.5, 26.5, 21.8, 64.3, 3.85]

Covariance feature012(assumed diagonal) =
[[8445.61      0.        0.        0.        0.        0.     ]
 [   0.     4830.25      0.        0.        0.        0.     ]
 [   0.        0.      702.25      0.        0.        0.     ]
 [   0.        0.        0.      475.24      0.        0.     ]
 [   0.        0.        0.        0.     4134.49      0.     ]
 [   0.        0.        0.        0.        0.       14.8225]]
```

50 sample:

ERROR = 0.32

```
Mean of features :
[89.44, 70.58, 26.92, 23.86, 27.82, 3.05]
Variance Matrix features :
[[15058.2909]]
Mean feature012 =[89.44, 70.58, 26.92, 23.86, 27.82, 3.05]

Covariance feature012(assumed diagonal) =
[[7999.5136    0.        0.        0.        0.        0.    ]
 [   0.     4981.5364    0.        0.        0.        0.    ]
 [   0.        0.      724.6864    0.        0.        0.    ]
 [   0.        0.        0.      569.2996    0.        0.    ]
 [   0.        0.        0.        0.      773.9524    0.    ]
 [   0.        0.        0.        0.        0.        9.3025]]
```

100 sample:

ERROR = 0.38

```
Mean of features :
[89.8, 69.84, 29.68, 24.48, 33.32, 3.29]
Variance Matrix features :
[[15542.8849]]
Mean feature012 =[89.8, 69.84, 29.68, 24.48, 33.32, 3.29]

Covariance feature012(assumed diagonal) =
[[8064.04      0.        0.        0.        0.        0.    ]
 [   0.     4877.6256    0.        0.        0.        0.    ]
 [   0.        0.      880.9024    0.        0.        0.    ]
 [   0.        0.        0.      599.2704    0.        0.    ]
 [   0.        0.        0.        0.     1110.2224    0.    ]
 [   0.        0.        0.        0.        0.       10.8241]]
```

C.

MLE و MAP یک مقدار ثابت را برمی‌گردانند، اما استنتاج بیزی تابع چگالی (یا جرم) احتمال را برمی‌گرداند. در نتیجه خروجی متفاوت است

```python
def mle_mean(M):

    Matrixs= np.array(M)
    sum_all = np.array([0]*len(Matrixs[0]))

    for element in Matrixs:
        sum_all = sum_all + element
    sum_all /= len(M)

    return sum_all.tolist()
```

```python
def mle_var(M, mean):

    Matrixs= np.array(M)
    mean_np = np.matrix(mean)
    sum_var = np.array([0]*len(Matrixs[0]))
    for element in Matrixs:
        sum_var = sum_var + \
                  (np.transpose(mean_np)*(mean_np))
    sum_var /= len(M)

    return sum_var
```

```python
def p_D(mu0, sigma0, sigma, D):
    import numpy as np

    D = [d[0] for d in D]
    sigmaN_pow2 = (sigma0**2 * sigma**2)/((len(D)*sigma0**2)+sigma**2)

    muN = ((len(D)*sigma0**2)/(len(D)*sigma0**2 + sigma**2))*(sum(D)/len(D)) +\
          (((sigma**2)/(len(D)*(sigma0**2)+sigma**2)))*mu0

    return muN,sigmaN_pow2
```

D.

دو داده با بیشترین خطا : sgpt,sgot که هرکدام به ترتیب خطای 23.0, 47.0 دارند

.٦

A.

```
prior of skin class is : 0.27507549132497106
prior of non-skin class is : 0.724924508675029
```

B.

```
 mean of skin class is : 151.9870035344449
 mean of non-skin class is : 112.2040278150894

variance of skin class is : 2171.8591745582553
variance non-skin class is : 6937.6443903786185
```

C.