SP_PHW2_Code_AmirPourmand_99210259

February 13, 2022

Please fill in your information and attach all the needed files in your final zip file uploaded.
Name: Amir Pourmand
Student Number: 99210259

1 0 Imports and Setups

[1]:
```
import math
import numpy as np
from scipy.special import factorial
import matplotlib.pyplot as plt
import scipy.stats as st
```

Set the last three digits of your student number as input to the function below:

[2]:
```
np.random.seed(259) #replace 123 with the last three digits of your student␣
↪number
np.set_printoptions(suppress=True)
```

2 1 Estimation

3 1.1 Generating Random Samples

Assume that n i.i.d samples are drawn from an exponential distribution with the following PDF:

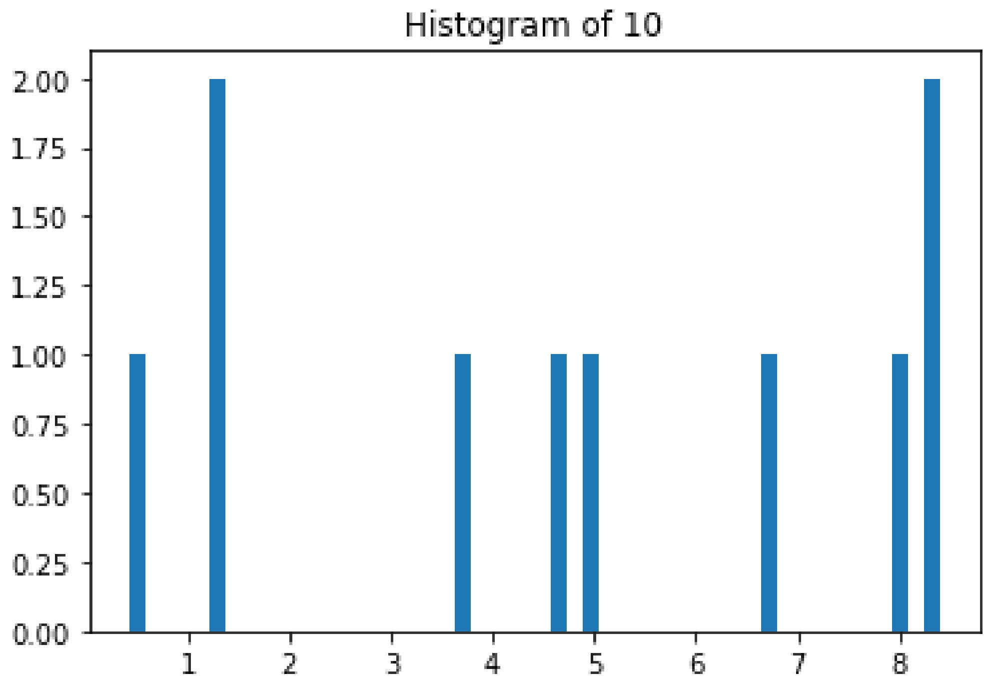
$$f(x_i|\lambda) = \lambda e^{-\lambda x_i}$$

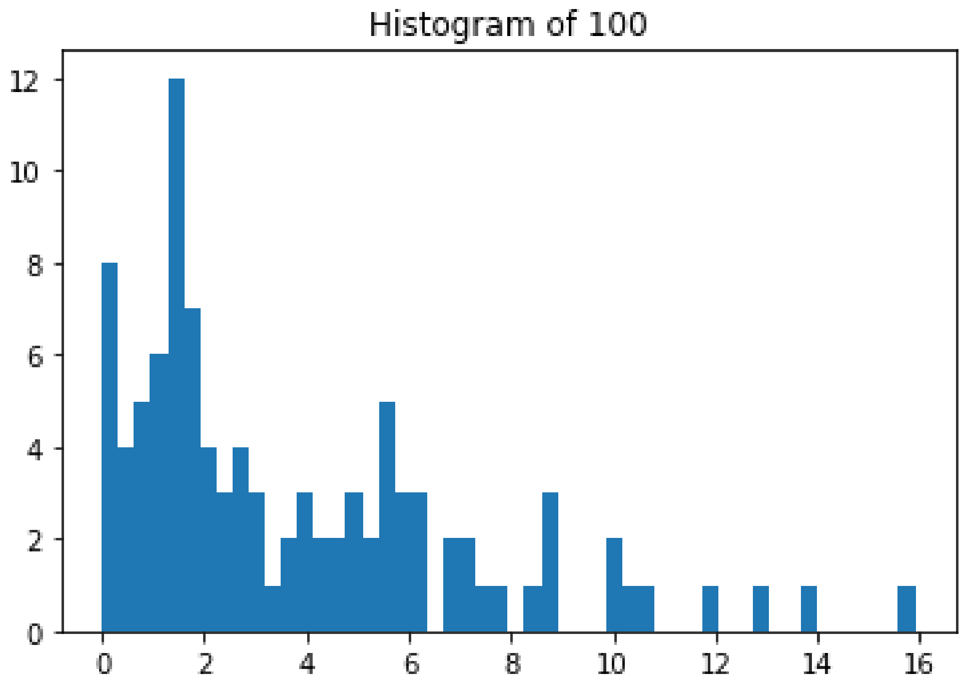
* For fixed $\lambda = 0.25$, generate random sequences of sizes 10, 100, 500, and 1000, and store them in seperate files named *exp_{size}.npy* attatched in your final zip file. * Plot each of the sequqnces generated above.

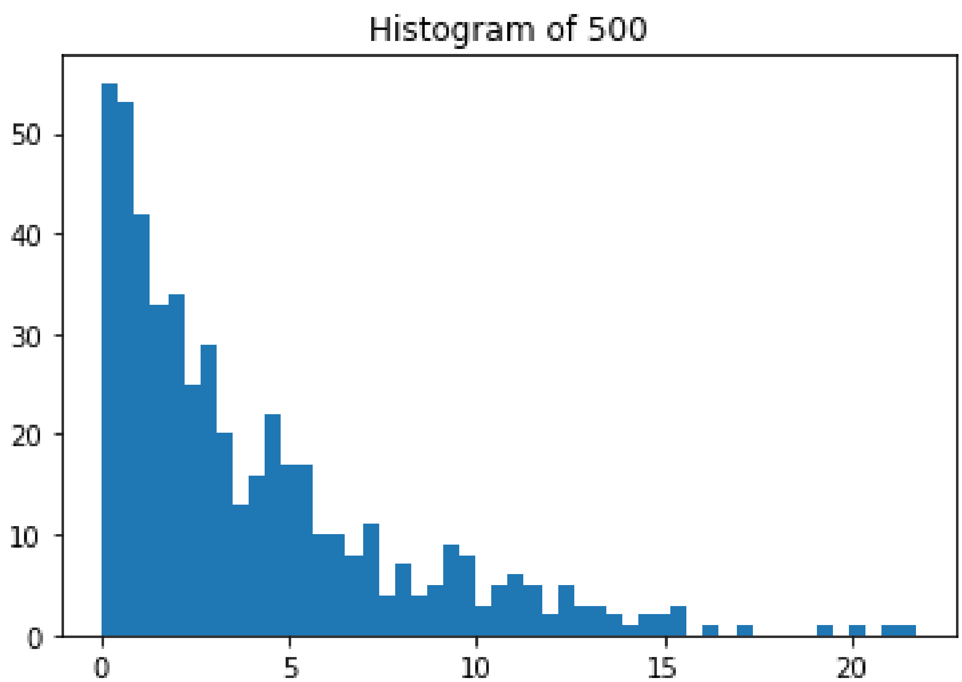
[3]:
```
#TODO
l = 0.25
size = [10,100,500,1000]
for item in size:
    sequence = np.random.exponential(scale=1/l,size=item)
```

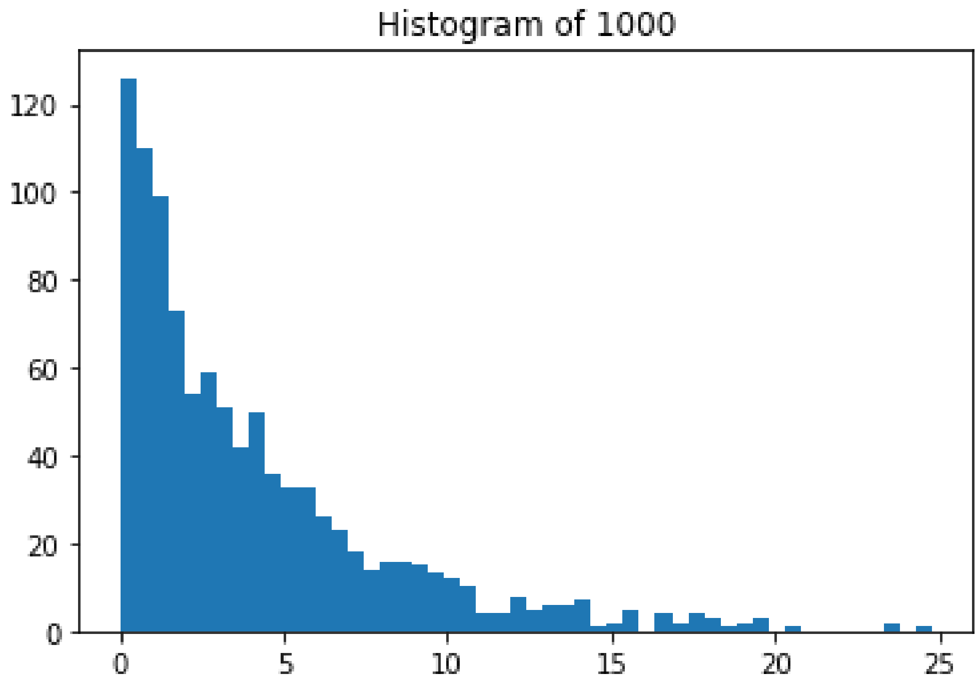
```
    np.save(f'exp_{item}.npy',sequence)

    plt.figure()
    plt.hist(sequence,bins=50)
    plt.title(f'Histogram of {item}')

np.load('exp_10.npy')
```

[3]:
```
array([4.68757046, 0.42520204, 1.23772355, 8.33141858, 4.93127457,
       7.95990286, 6.66258358, 3.71598807, 8.40292105, 1.3232411 ])
```




Histogram of 100
0
2
4
6
8
10
12
0
2
4
6
8
10
12
14
16


Histogram of 500
0
10
20
30
40
50
0
5
10
15
20



4 1.2 MLE Estimation

- Find the log likelihood equation for the given distribution. Include your solution in your attached pdf file.
- Suppose we have 40 candidates for λ which are in the following form:

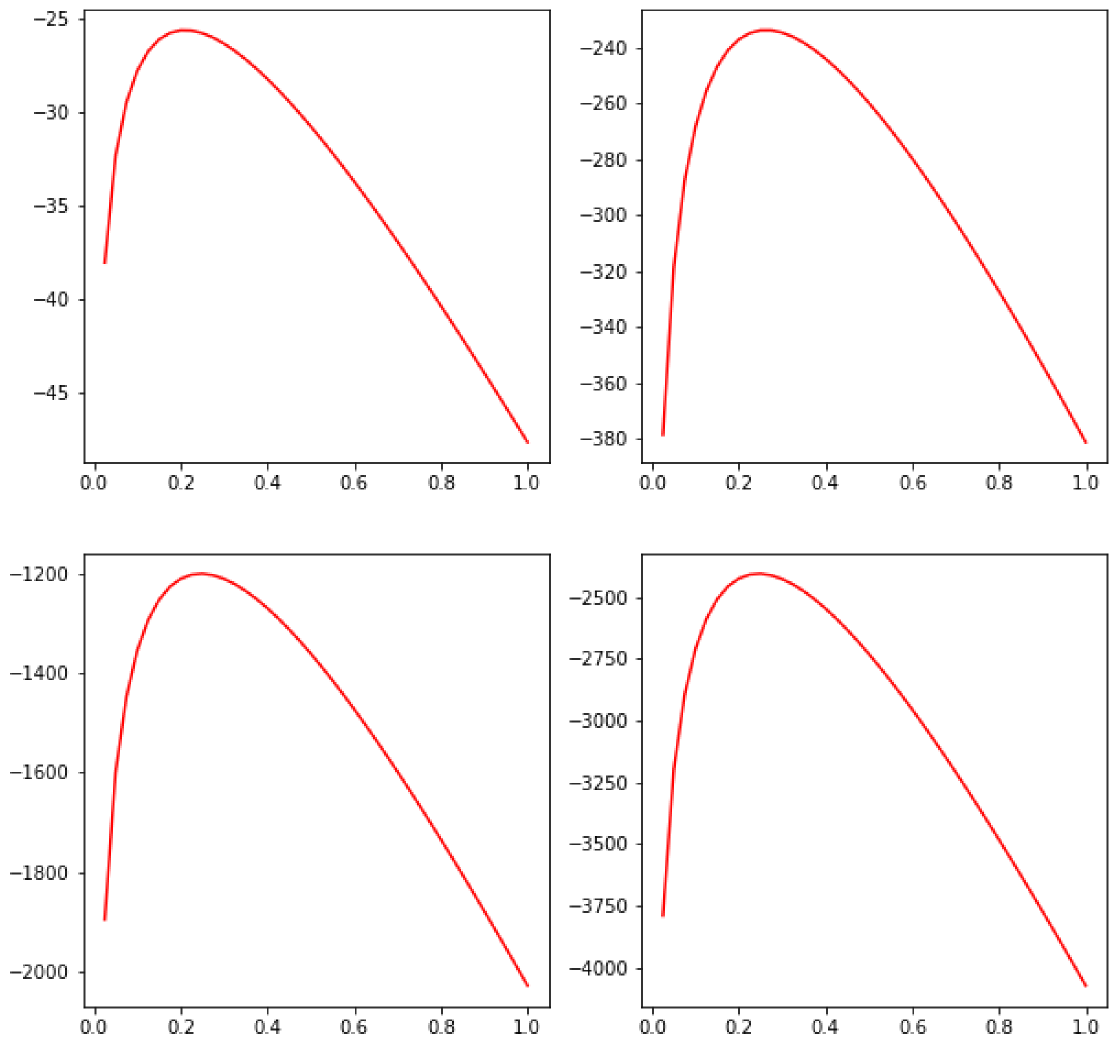
$$\lambda_{candidate} = 0.025i, (1 \leq i \leq 40)$$

  Calculate and plot the log likelihood values for the candidates of λ for each of the four sequences generated in secion 1.1.
- Find the MLE estimator and include it in your solution. Also find the best estimator of λ from the log likelihood values of the previous part for each of the four sequences generated in secion 1.1.

[4]:
```
#TODO
fig,axes=plt.subplots(figsize=(10,10),nrows=2,ncols=2)

lrange = np.arange(1,41,1)*0.025
i=0
mle_estimates = []
for item in size:
    x_values=np.load(f'exp_{item}.npy')
    n=len(x_values)
```

```
    exponential_likelihood = n*np.log(lrange)-np.sum(lrange[:
↪,None]@x_values[None,:],axis=1)

    current_ax =axes[int(i>1),int(i%2)]
    current_ax.plot(lrange,exponential_likelihood,'-r')
    i = i+1

    mle_estimate = n/np.sum(x_values)
    mle_estimates.append(mle_estimate)
    print(f'size = {item}\t lambda = ',n/np.sum(x_values))
```

```
size = 10        lambda =  0.20974110759800615
size = 100       lambda =  0.2624507583700247
size = 500       lambda =  0.24656359010501716
size = 1000      lambda =  0.24547164102849842
```



[ ]:

```

```

5 1.3 MAP Estimation

Gamma distrutione with the shape k and the scale θ, noted by $G(k, \theta)$, has the following PDF function:

$$f(x|k, \theta) = \frac{1}{\Gamma(k)\theta^k} x^{k-1} e^{-\frac{x}{\theta}},$$

where for any integer n:

$$\Gamma(n) = (n-1)!$$

* Suppose that the parameter λ of the exponential distribution is itself originated from the gamma distribution $G(2, 0.5)$. Plot the prior probabilities for the candidates of λ introduced in section 1.2, given this prior distribution. * Find the posterior distribution. Include your solution in your attatched pdf file. * Plot the posterior distribution for the candidates of λ for each of the four sequences generated in secion 1.1. * Find the MAP estimator and include it in your solution. Also find the best MAP estimator of λ using the probabilities obtained for the posterior distribution for each of the four sequences generated in secion 1.1.

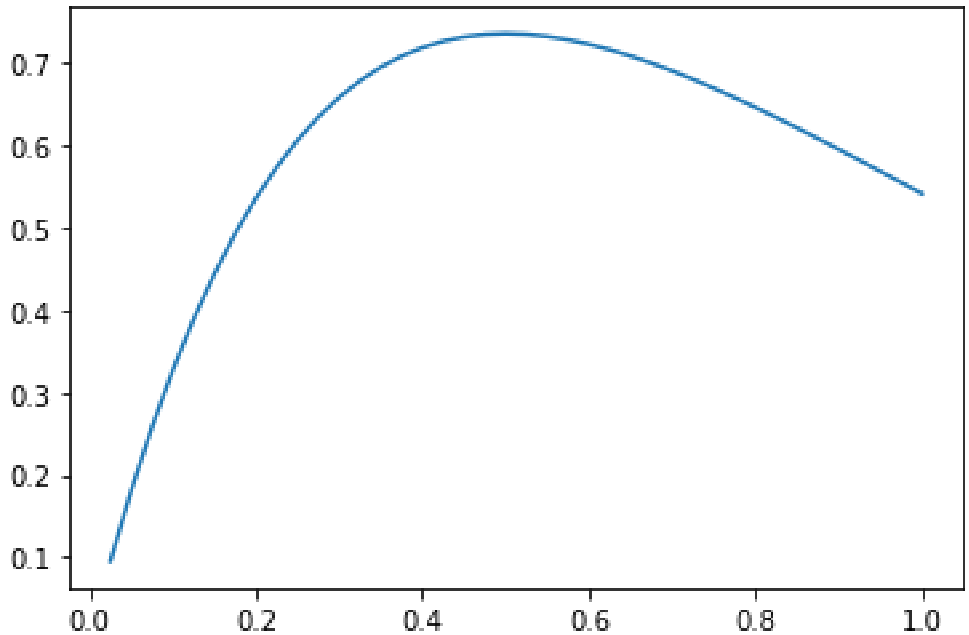
[5]:

```python
def gamma(x,alpha,beta):
    return x**(alpha-1) * np.exp(-1*x*beta) * beta**alpha / factorial(alpha-1)

def log_gamma(x,alpha,beta):
    return (alpha-1)*np.log(x) + (-1*x*beta) + alpha*np.log(beta) - np.sum(np.
 ↪arange(1,alpha))
```

[6]:

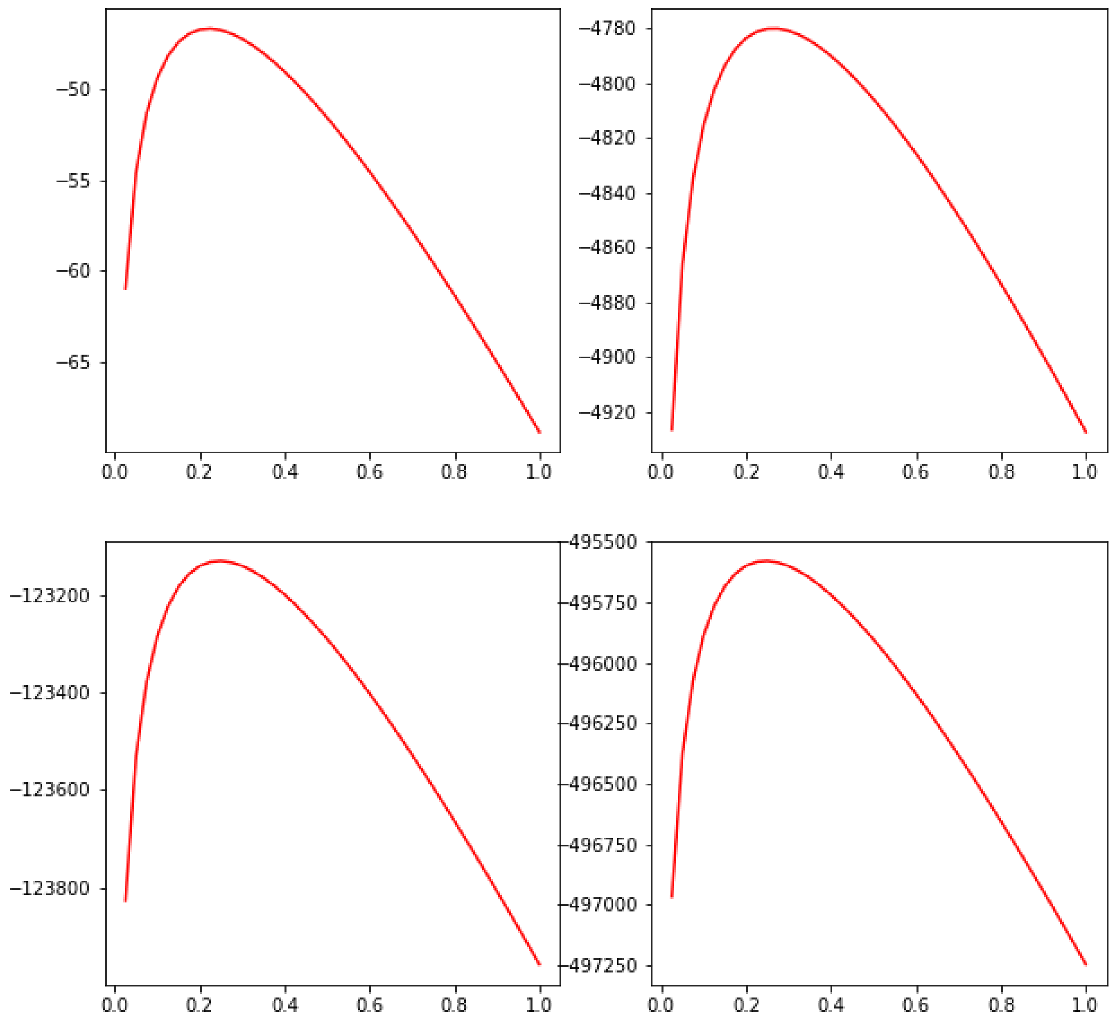
```python
#TODO
k,theta = 2,0.5
alpha = k
beta = 1/theta
prob=gamma(lrange,alpha,beta)
plt.plot(lrange,prob)
```

[6]: [<matplotlib.lines.Line2D at 0x7f8144295880>]



[7]:
```
fig,axes=plt.subplots(figsize=(10,10),nrows=2,ncols=2)

map_estimates = []
i=0
for item in size:
    x_values=np.load(f'exp_{item}.npy')
    n=x_values.shape[0]

    posterior = log_gamma(lrange,n+alpha,beta+np.sum(x_values))

    current_ax =axes[int(i>1),int(i%2)]
    current_ax.plot(lrange,posterior,'-r')
    i = i+1

    map_estimate = (n+alpha-1)/ (beta+np.sum(x_values))
    map_estimates.append(mle_estimate)
    print(f'size = {item}\t lambda=', map_estimate)
```

```
size = 10        lambda= 0.22142675959591004
size = 100       lambda= 0.2636911471229672
size = 500       lambda= 0.24681329659525048
size = 1000      lambda= 0.2455965386987563
```

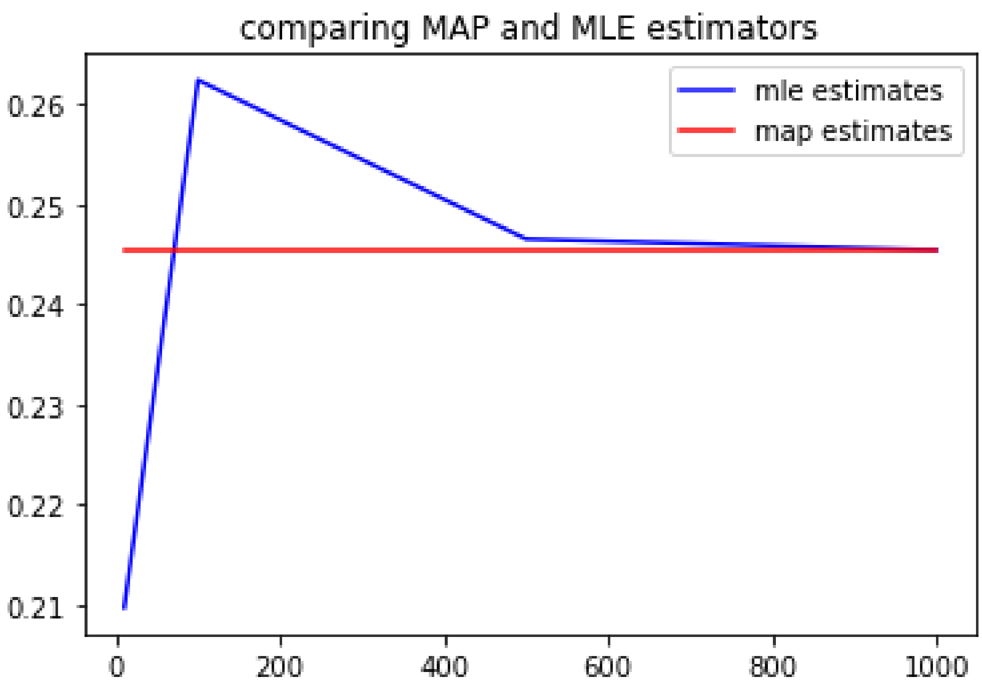


6 1.4 Conclusion and Analysis

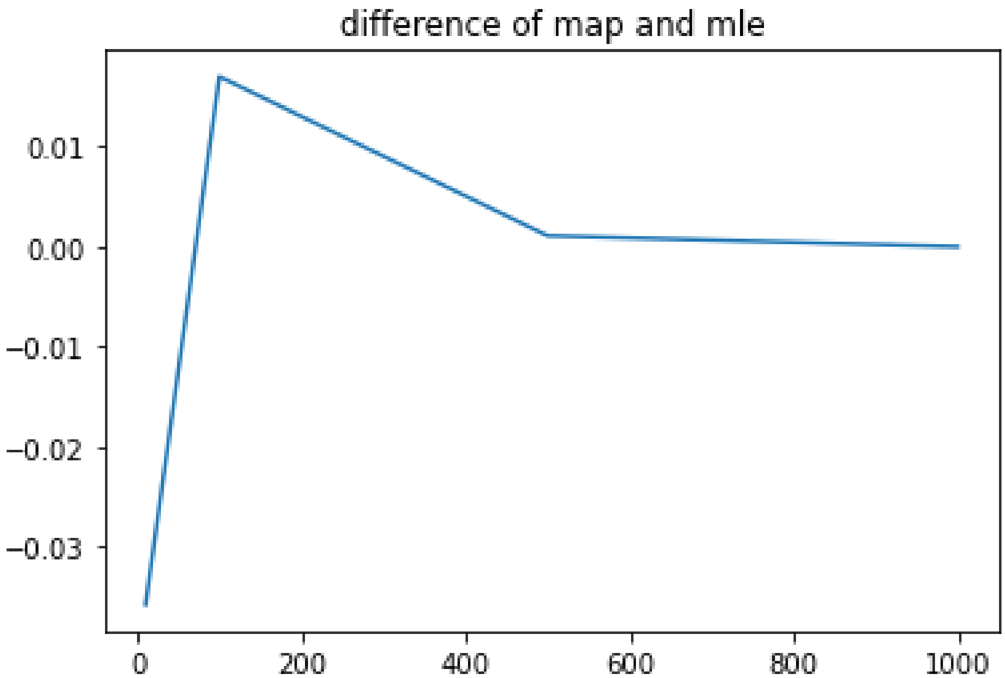
Include your answers of the below questions in your attatched pdf file or in the cell below: * Compare the MAP and MLE estimators in the previous sections for the given different four sequqnces. * Compare the difference of the two estimators with the actual parameter ($\lambda = 0.25$) with respect to the number of samples in each sequence. * Based on the results from part 1.2 and 1.3, conclude about the relation between these two estimators, as the number of samples increases.

[8]:
```
#TODO
plt.plot(size,mle_estimates,'b',label='mle estimates')
plt.plot(size,map_estimates,'r',label='map estimates')
plt.legend()
plt.title('comparing MAP and MLE estimators')
```

```
[8]: Text(0.5, 1.0, 'comparing MAP and MLE estimators')
```




```
[9]: diff=np.array(mle_estimates)-np.array(map_estimates)
     plt.plot(size,diff)
     plt.title('difference of map and mle')
     plt.show();
```



We conclude that difference between mle and map gets less when we increase sample size.

7 2 Hypothesis Testing

In this section, you are supposed to implement functions for performing Z-test and p-test. **You are not allowed to use the ready package implementations of these test functions.** Still, you may use built-in functions for converting standard normal distribution probablities to areas under the normal curve and vice versa.

8 2.1 Function Implementation

Implement the hypothesis_test function. You are supposed to implement it as the function described in the cell below. Description, inputs, and output are provided in the cell below.

[10]:
```
def z_test(samples, hypothesis_mean, true_variance, alpha, condition):
    """
    Description:
    The function applies hypothesis testing z_test on the collected samples␣
↪given a null hypothesis.
```

```
    Inputs:
    samples -> collected data samples to be tested
    hypothesis_mean -> the mean value being tested
    true_variance -> the actual variance of the samples (given in advance)
    alpha -> level of significance of the test
    condition -> one of the following string values: "eq", "gte", "lte"␣
↪specifying the condition on the hypothesis mean
                         "eq": if the whole population mean is equal to the␣
↪hypothesis mean.
                         "gte": if the whole population mean is greater than or␣
↪equal to the hypothesis mean.
                         "lte": if the whole population mean is less than or equal␣
↪to the hypothesis mean.

    Outputs:
    True if the null hypothesis is not rejected under the z_test with the␣
↪specified conditions, False otherwise
    """
    #TODO
    z = (samples.mean() - hypothesis_mean) / math.sqrt(true_variance/
↪len(samples))

    if condition == "lte":
      t = st.norm.ppf(1-alpha)
      return z <= t

    elif condition == "gte":
      t = st.norm.ppf(alpha)
      return z >= t

    else:
      t = st.norm.ppf(alpha/2)
      return z >= t and z <= (-1 * t)
```

[21]:
```
def p_test(samples, hypothesis_mean, true_variance, alpha, rejection_side):
    """
    Description:
    The function applies hypothesis testing p_test on the collected samples␣
↪given a null hypothesis.

    Inputs:
    samples -> collected data samples to be tested
    hypothesis_mean -> the mean value being tested
    true_variance -> the actual variance of the samples (given in advance)
    alpha -> level of significance of the test
```

```
    condition -> one of the following string values: "eq", "gte", "lte"␣
↪specifying the condition on the hypothesis mean
                        "eq": if the whole population mean is equal to the␣
↪hypothesis mean.
                        "gte": if the whole population mean is greater than or␣
↪equal to the hypothesis mean.
                        "lte": if the whole population mean is less than or equal␣
↪to the hypothesis mean.
    Outputs:
    True if the null hypothesis is not rejected under the p_test with the␣
↪specified conditions, False otherwise
    """
    #TODO
    z = (samples.mean() - hypothesis_mean) / math.sqrt(true_variance/
↪len(samples))

    if rejection_side == "lte":

      p_value = st.norm.cdf(z)
      p_value = 1 - p_value

    elif rejection_side == "gte":
      p_value = st.norm.cdf(z)

    else:
      temp_z = (-1) * abs(z)
      p_value = st.norm.cdf(temp_z)
      p_value *= 2

    return p_value >= alpha
```

9 2.2 Applying the Tests

In the cell below, load the samples given to you. Samples are provided in the attatched file **samples.npy**.
Suppose the true variance for the sample space is 9. ($\sigma^2 = 9$)

[13]:
```
#TODO
#TODO
true_variance = 9

with open("samples.npy","rb") as f:
  samples = np.load(f)
```

Examine the following hypotheses on the data using **z__test**, seperately with $\alpha = 0.05$ and $\alpha = 0.01$:

* The mean of the whole population is 8.2. * The mean of the whole population is at most 10. * The mean of the whole population is at least 9.2

[19]:
```
#TODO
#TODO
alphas = [0.01,0.05]
for alpha in alphas:
    print('='*30)

    print(alpha,":")

    h1 = z_test(samples, 8.2, true_variance, alpha, "eq")
    print(h1)

    h2 = z_test(samples, 10, true_variance, alpha, "lte")
    print(h2)

    h3 = z_test(samples, 9.2, true_variance, alpha, "gte")
    print(h3)

    print('='*30)
```

```
==============================
0.01 :
False
True
True
==============================
==============================
0.05 :
False
True
False
==============================
```

Examine the following hypotheses on the data using **p__test**, seperately with $\alpha = 0.05$ and $\alpha = 0.01$:

- The mean of the whole population is 8.8.
- The mean of the whole population is at most 9.
- The mean of the whole population is at least 8.4.

[20]:
```
#TODO
#TODO.

for alpha in alphas:

    print('='*30)
    print(alpha,":")

```

```
    h1 = p_test(samples, 8.8, true_variance, alpha, "eq")
    print(h1)

    h2 = p_test(samples, 9, true_variance, alpha, "lte")
    print(h2)

    h3 = p_test(samples, 8.4, true_variance, alpha, "gte")
    print(h3)

    print('='*30)
```

```
==============================
0.01 :
True
True
True
==============================
==============================
0.05 :
False
True
True
==============================
```

10 2.3 Conclusion and Analysis

Overally, conclude about the probable value ranges for the mean of the whole population using the results from the previous part. Include your answers in your attatched pdf file or in the cell below.

#TODO

. .

. . . !

[ ]:

```

```