Physics 129L: Problem Set 4

Problem Set Submission Guideline

A) Github Submissions We will use GitHub for problem set submissions. To access the problem set, please **fork** and **clone** the **forked** repository to your local virtual machine. **Please complete the problem set in this forked directory.** Submit **a pull request** for merging into the main branch before the problem set due date.

B) .tar.gz File compression and submission on Github For each problem set, you are asked to submit the compressed version of the problem set to GitHub via git operation. Here is a step-by-step guideline:

1. Use the **tar** command to compress the problem set directory into a **single** ".tar.gz" file.
2. Obtain the sha256sum by running "sha256sum P2.tar.gz".
3. Echo the **full sha256sum** to a text file named "sha25sum_problem_set.txt".
4. Initialize a git repository named "Archive_P# (#: problem set number) on your local machine, and move both the "tar.gz" file and the "sha25sum_problem_set.txt" file to the repository.
5. Create an empty **public** directory under the **same name** in **your own GitHub account**.
6. **Push** this local repository to the remote repository.

Problem 1: Statistical Inference on Biased Coins

Let's consider a problem similar to the one we discussed in class. You are given three datasets, each containing 500 coin-flip outcomes represented as **Boolean** values (True means Head, and False means Tail). In this problem, your task is to investigate potential biases in the coin flips within these datasets. To load the dataset, please use the following code:

In [ ]:
```
import json
# Define the file path for the JSON file
json_file_path = 'problem_1/dataset_1.json'

# Read the boolean data from the JSON file into a Python list
with open(json_file_path, 'r') as json_file:
    boolean_list_out = json.load(json_file)
```

Bayesian Inference

As discussed in class, for a given p, the probability that $N = 500$ tosses result in M heads and $N - M$ tails is given by the binomial distribution, with the probability mass function, namely the likelyhood function:

$$P(M, N|p) = \binom{N}{M} p^M (1-p)^{N-M}.$$

Then the corresponding posterior is given by the Bayes' theorem,

$$P(p|M,N) \sim \frac{1}{B(M,N)} P(M,N|p)P(p),$$

where,

$$B(M,N) = \int_0^1 dp\, p^M (1-p)^{N-M} = \frac{\Gamma(M+1)\Gamma(N-M+1)}{\Gamma(N+2)}.$$

Note that the above integral defines the Beta function, expressed in terms of the Gamma function (factorial, $\Gamma(n+1) = n!$). For more details, please check out https://en.wikipedia.org/wiki/Beta_function).

A)

First, we assume that our initial model for p is a uniform distribution (prior), i.e., $P(p) \sim 1/N$.

With this prior distribution, apply the Bayesian inference methods discussed in class to calculate the likelihood functions for three datasets named 'problem_1/dataset_1.json,' 'problem_1/dataset_2.json,' and 'problem_1/dataset_3.json.' Subsequently, create a (3,1) plot to visualize the corresponding posterior distributions for each dataset. Additionally, numerically calculate the expectation value and variance of each posterior distribution. (hint: look at the stats.ipynb I provided on Canvas)

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt

'''--------------------Write your code below this line----------------------



'''--------------------Write your code above this line----------------------
```

Frequentist Inference: Stirling's approximation

As we discussed in class, the frequentist inference assumes a fixed p_{true}. Maximum Likelihood Estimation (MLE) is a technique used to estimate the parameter p_{true} of a presumed probability distribution (in this case, binomial) based on observed data. It involves maximizing a likelihood function,

$$P(M,N|p) = \binom{N}{M} p^M (1-p)^{N-M}.$$

with the objective of making the observed data most likely under the assumed model. In practice, working with the natural logarithm of the likelihood function $P(M,N|p)$, known as the log-likelihood, is often more convenient,

$$\mathcal{P}(M, N|p) = \log[P(M, N|p)] = \log[\binom{N}{M}] + M\log(p) + (N - M)\log(1 - p).$$

When N is large, we can approximate the factorial by the Stirling's approximation,

$$\log(n!) \approx n\log(n) - n + \frac{1}{}\log(2\pi n).$$

B)

Numerically check the Stirling's approximation by compute both sides of the above equation and make a (2,1) plot that shows the following: 1) for factorial, vary N from 1 to 10 and calculate each value then make a scatter plot. Plot a **smooth curve** that shows the Stirling's formula (remember, the Stirling's formula takes real values) and the Gamma function $\Gamma(N + 1)$. 2) plot the difference between the Stirling's formula and the Gamma function. Please label your plot.

In [ ]:

```
'''--------------------Write your code below this line----------------------



'''--------------------Write your code above this line----------------------
```

Frequentist Inference: Maximum Likelihood Estimation

Maximum Likelihood Estimation is given by the condition,

$$\frac{\partial}{\partial p}\mathcal{P}(M, N|p) = 0.$$

C)

Calculate the above condition **analytically**,
''**write you answer below**''

Steps:

'' **write you answer above**''

What is the $p_{\max}$ for the above three datasets? ''**write you answer below**''
Data set 1:
Data set 2:
Data set 3:
'' **write you answer above**''

D)

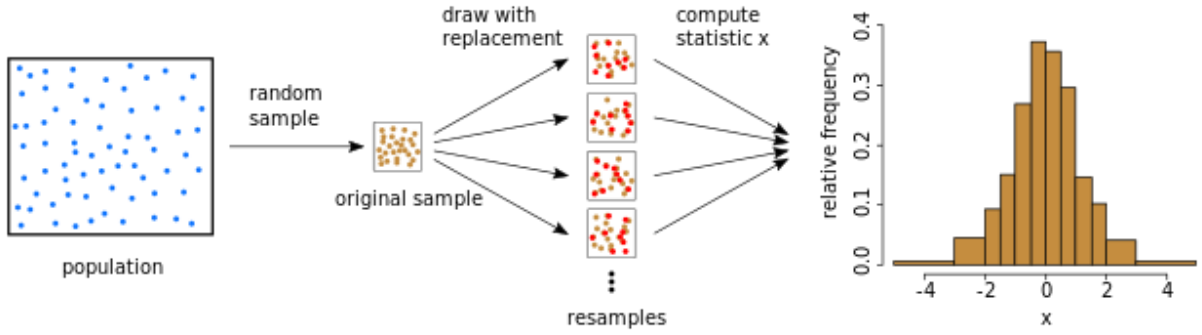
Using the method discussed in the class to find the maximum likelyhood function **numerically** for all three datasets. Plot three maximum likelyhood functions in three different subplots, and also plot the $p_{\max}$ above. (hint: look at the stats.ipynb). What do you find?

In [5]:
```
'''--------------------Write your code below this line----------------------


'''--------------------Write your code above this line----------------------
```

Out[5]:
```
'--------------------Write your code above this line----------------------
---------'
```

Bootstrapping

Bootstrapping belongs to the wider category of resampling methods and involves any test or metric that employs random sampling with replacement, effectively replicating the sampling process (https://en.wikipedia.org/wiki/Bootstrapping_(statistics)),



This process involves a substantial number of iterations, and for each of these bootstrap samples, we compute its expectation value or other statistical properties. The distribution of these expectation values or other statistical properties then provides insights into the characteristics of the underlying distribution.

E)

Bootstrap three datasets 100 times each with sample sizes of 5, 15, 40, 60, 90, 150, 210, 300, and 400, and create a (3,3) histogram (so there will be a total of three (3,3) histograms). Calculate the expectation value and variance for each and label the subplot title with the sample size.

In [ ]:
```
'''--------------------Write your code below this line----------------------



'''--------------------Write your code above this line----------------------
```

F)

Plot the expectation values and variances of the above in comparison with the one you have calculated in **C)**. Are there any difference? and is that what you expected? why?

In [ ]:
```
'''--------------------Write your code below this line----------------------



'''--------------------Write your code above this line----------------------
```

Particle Decay

Unstable particles are emitted from a source and undergo decay at a distance x, a continuous real number following an exponential probability distribution with a characteristic length λ. Decay events can be detected only if they happen within a range extending from $x = 1$ to $x = \infty$, (a.u.). A total of N decay events are observed at positions $\{x_1, \ldots, x_N\}$.

Decay Process: Probability density function

For a given decay parameter, λ, the probability of observing a particle at a distance x is given by an exponential function,

$$P(x \mid \lambda) = \frac{1}{Z(\lambda)} \frac{1}{\lambda} e^{-x/\lambda} \qquad 0 < x < \infty$$

where

$$Z(\lambda) = \int_0^\infty dx \frac{1}{\lambda} e^{-x/\lambda} = e^{-1/\lambda}.$$

A)

Define a particle class that takes a single input, named decay constant. Write the probability density function (given a fixed λ) showing above as a method ('pdf_decay') of the class.

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt


'''--------------------Write your code below this line----------------------

class Particle:
    def __init__():
        pass

    def pdf_decay():
        pass

'''--------------------Write your code above this line----------------------
```

B)

Define four particles as class objects with decay constants [0.4,1,2,3]. Plot the pdf curves $P(x \mid \lambda)$ and label them (in a single plot). (hint: you may want to define different class methods to output the value).

In [6]:
```
'''--------------------Write your code below this line----------------------


'''--------------------Write your code above this line----------------------
```

Out[6]:
```
'--------------------Write your code above this line----------------------
---------'
```

C)

Write the probability density function $P(x \mid \lambda)$ (given a fixed position x) showing above as a **new method** in the particle class. Define four particles as class objects with fixed positions [2,5,7,10]. Plot the pdf curves and label them (in a single plot).

In [6]:
```
'''--------------------Write your code below this line----------------------


'''--------------------Write your code above this line----------------------
```

Out[6]:
```
'--------------------Write your code above this line----------------------
---------'
```

D)

Define a **new method** in the particle class that generate a 3D surface Plot $P(x \mid \lambda)$ as a function of x and λ. You should create the surface plot in '.png' format, and output the image to a user provided path.

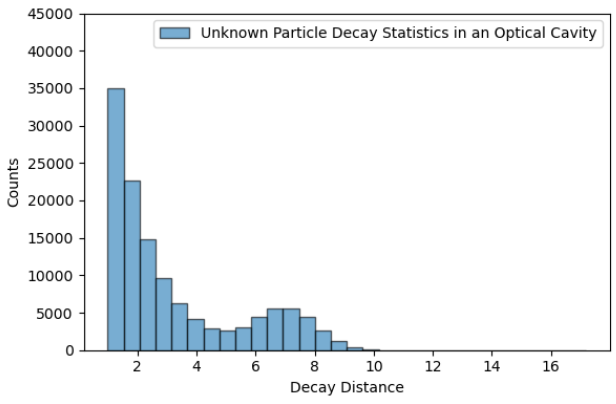
In [7]:
```
# Import packages for making a 3D plot
from mpl_toolkits.mplot3d import Axes3D


'''--------------------Write your code below this line----------------------



'''--------------------Write your code above this line----------------------
```

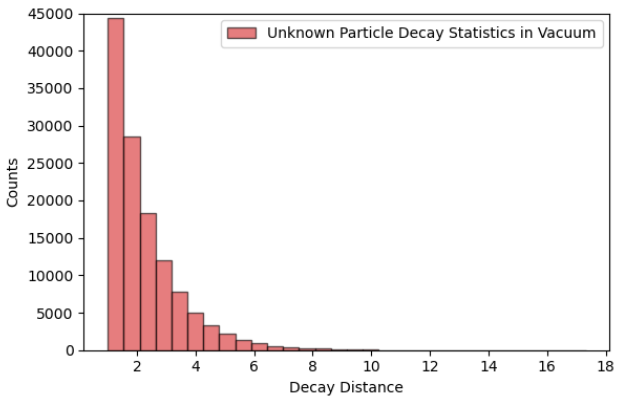
Out[7]:
```
'--------------------Write your code above this line----------------------
---------'
```

In [ ]:
```
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt


'''--------------------Write your code below this line----------------------



'''--------------------Write your code above this line----------------------
```

Unknown Particle Decay

You are given two datasets that record the decay distances of approximately 10^4 data points in both a vacuum and an optical cavity. In this question, you are asked to infer the values of the decay constants, λ, under both conditions. The measurement files are 'Vacuum_decay_dataset.json' and 'Cavity_decay_dataset.json.' Here are snapshots of the data.





E)

ou can observe that the optical cavity modifies a fraction of particles into a different type with distinct decay properties. What are the decay constants in both cases? How can you define the decay constant in the presence of an optical cavity? What additional structures have you observed? Let's assume the second particle type follows a Gaussian probability distribution function:

$$\mathcal{F}(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

For simplicity, you can consider that the observed decay counts in an optical cavity are proportional to: $\sim \mathcal{F}(x \mid \mu, \sigma) + P(x \mid \lambda)$. What are the parameters μ, σ, λ that best fit the data? Please provide your explanations and any relevant code.

In [ ]:
```
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt


'''--------------------Write your code below this line----------------------




'''--------------------Write your code above this line----------------------
```

Moment Generating Function (MGF)

The MGF of a random variable x is denoted as $M(t)$, where t is a real number. It is defined as:

$$M(t \mid \lambda) \sim \int dx e^{tx} P(x \mid \lambda).$$

It generates moments by differentiation:

$$\frac{\partial}{\partial t} M(t \mid \lambda)\Big|_{t=0} \sim \int dx \Big(x P(x \mid \lambda)\Big)$$

The key idea is to take successive derivatives of the MGF with respect to the parameter t and evaluate them at $t = 0$. Each derivative, when evaluated at $t = 0$, corresponds to a specific moment of the random variable. The process is systematic, and it allows you to calculate moments of different orders (mean, variance, skewness, kurtosis, etc.) in a structured way.

MGF of the decay process

Let's recal the probability density function in **Problem 2**,

$$P(x \mid \lambda) = \frac{1}{Z(\lambda)} \frac{1}{\lambda} e^{-x/\lambda} \qquad 0 < x < \infty$$

where

$$Z(\lambda) = \int_0^\infty dx \frac{1}{\lambda} e^{-x/\lambda} = e^{-1/\lambda}.$$

A)

What is the MGF for the above probability density function? Calculate the above condition **analytically**,
''**write you answer below**''

Steps:

'' **write you answer above**''

B)

Using the above result, calculate the mean and variance **analytically**,
''**write you answer below**''

Steps:

'' **write you answer above**''

C)

Numerically evaluate (finite differences) the first-order derivative and the second-order derivative that yield the mean and variance at various λ values. Plot the analytical functions against the numerical results.

In [1]:
```
'''--------------------Write your code below this line--------------------




'''--------------------Write your code above this line---------------------
```

Out[1]:
```
'--------------------Write your code above this line----------------------
---------'
```

MGF of a biased coin

Let's recal the likelihood function (probability mass function) in **Problem 1** ,

$$P(M, N|p) = \binom{N}{M} p^M (1-p)^{N-M}.$$

D)

What is the MGF for the above probability mass function? Calculate the above condition **analytically**,
''**write you answer below**''

Steps:

'' **write you answer above**''

Hint, you should consider the sum,

$$M(t) = \sum_{x=0}^{n} e^{tx} \binom{n}{x} p^x (1-p)^{n-x}$$

and use the fact,

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k} b^k$$

E)

Using the above result, calculate the mean and variance **analytically**,
''**write you answer below**''

Steps:

'' **write you answer above**''

F)

Numerically evaluate (finite differences) the first-order derivative and the second-order derivative that yield the mean and variance at various λ values. Plot the analytical functions against the numerical results.

In [2]:
```
'''--------------------Write your code below this line----------------------




'''--------------------Write your code above this line----------------------
```

Out[2]:
```
'--------------------Write your code above this line----------------------
---------'
```