Homework 2

Intro to Statistics

This notebook is arranged in cells. Texts are usually written in the markdown cells, and here you can use html tags (make it bold, italic, colored, etc). You can double click on this cell to see the formatting.

The ellipsis (...) are provided where you are expected to write your solution but feel free to change the template (not over much) in case this style is not to your taste.

Hit "Shift-Enter" on a code cell to evaluate it. Double click a Markdown cell to edit.

Problems are directly taken from MacKay Chapter 3 (http://www.inference.org.uk/itprnn/book.pdf). We recommend you to read Chapter 3 before starting HW2. Problem 2 could be challenging. If you have any questions/concerns, talk to us during discussion/office hours.

Link Okpy

In [ ]:

```
from client.api.notebook import Notebook
ok = Notebook('hw2.ok')
_ = ok.auth(inline = True)
```

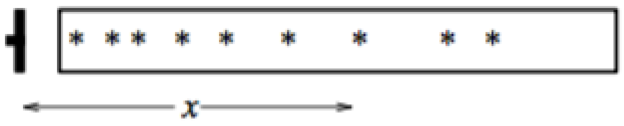
Imports

In [ ]:

```
import numpy as np
from scipy.integrate import quad
#For plotting
import matplotlib.pyplot as plt
%matplotlib inline
```

Problem 1 - Inferring a Decay Constant

Unstable particles are emitted from a source and decay at a distance x, a real number that has an exponential probability distribution with characteristic length λ. Decay events can be observed only if they occur in a window extending from x = 1cm to x = 20cm. N decays are observed at locations $\{x_1, \ldots, x_N\}$. What is λ?



Given λ, the probability of observing a particle at a distance x is:

$$P(x \mid \lambda) = \begin{cases} \frac{1}{\lambda} e^{-x/\lambda} \Big/ Z(\lambda) & a < x < b \\ 0 & \text{otherwise} \end{cases}$$

where

$$Z(\lambda) = \int_a^b dx \frac{1}{\lambda} e^{-x/\lambda} = \left(e^{-a/\lambda} - e^{-b/\lambda}\right).$$

Here, $a = 1,\ b = 20.$

1. Write a function for $Z(\lambda)$. Then, use it to write another function for $P(x|\lambda)$.

Henceforth, we refer to λ as L (for the sake of simplicity).

Check if your function can return a correct value if either x or L is a 2D array. Say x is a scalar, and L is a vector with N elements. If you calculate the product $x * L$, the dimension of x is stretched to $N \times 1$ in order to match that of L (broadcasting (https://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)). If x and L are both vectors, then they must have the same dimensions to perform arithmetic operations on them ($x * L$, x+L, x/L, etc).

In [ ]:

```
def Z(L, a, b):
    '''Normalizing constant function for a characteristic length L, assuming tha
t the data are
    truncated between x = a and x = b.
    '''
    return ...

# Z = quad(lambda y, L: 1./L*exp(-y/L), a, b, args = (L, ))[0] will return the s
ame value.

def pdf_decay(x, L, a, b):
    'Probability of one data point, given L'
    return ...
```

2. Plot $P(x|\lambda)$ as a function of x for $\lambda = 2, 5, 10$. Make sure to label each plot.

In [ ]:

```
# Create arrays for x and L.
x = np.linspace(1e-1, 20, 100)
L = ...

# Plot the probability desity as a function of x for each lambda.
# Hint: You can use a for-loop and make a plot for each element of an array L.
 (But you don't
#have to do it in this way.)
# Hint2: You should label each plot. To do this in a for-loop, you should rememb
er that you can
#insert values into a string with the placeholder % (https://docs.python.org/2.
4/lib/typesseq-strings.html).

...

plt.xlim(1, 20)
plt.xticks(np.append(np.array([1]), np.arange(2, 20+1, 2)))
plt.ylim(0, 0.3)
plt.xlabel(' ... ')
plt.ylabel(' ... ')
plt.legend()
plt.show()
```

*3. Plot $P(x|\lambda)$ as a function of λ for $x = 3, 5, 12$. (This function is known as the **likelihood** of λ) Make sure to label each plot. Note that a peak emerges in each plot.*

In [ ]:

```
# Create arrays for x and L.
x = ...
L = np.logspace(-1, 2, 100)

# Plot the probability desity as a function of L for each x. Label each plot.
...

plt.xlim(1.e-1, 1.e2)
plt.ylim(0, 0.2)
plt.xlabel(' ... ')
plt.ylabel(' ... ')
plt.legend()
plt.show()
```

4. Plot $P(x|\lambda)$ as a function of x and λ. Create a surface plot.

In [ ]:

```
# Import packages for making a 3D plot
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
```

In [ ]:

```
# Create arrays for x and L. These define your "x" and "y" coordinates.
x = np.linspace(1, 3, 30)
L = np.logspace(-0.5, 1, 30)
# Create coordinate matrices from coordinate vectors.
x, L = np.meshgrid(x, L)

# Evaluate probability densities at all (x,y) coordinates. This is your "z" coor
dinate.
z = ...

# Make plot
fig = plt.figure(figsize = (12,8))
ax = fig.gca(projection='3d')
surf = ax.plot_surface(x, L, z, vmax=3, rstride=1, cstride=1, cmap=cm.coolwarm,
linewidth=0, antialiased=True)

# Add contour plots
cset = ax.contour(x, L, z, 10, zdir='x', offset=0.9, cmap=cm.Set1)
cset = ax.contour(x, L, z, 20, zdir='y', offset=10.5, cmap=cm.Set1)

ax.set_xlim(0.9, 3)
ax.set_ylim(1.e-1, 1.e1)

ax.set_xlabel(' ... ')
ax.set_ylabel(' ... ')
ax.set_zlabel(' ... ')

fig.colorbar(surf, shrink=0.5, aspect=5)
plt.show()
```

In the above figure, two contour plots (constant x and y slices) are also included. Compare them to the figures you created in part 2 and 3. They are the same; they correspond to vertical sections through surface.

Now write Bayes' theorem:

$$P(\lambda \mid \{x_1, \ldots, x_N\}) = \frac{P(\{x\}|\lambda)P(\lambda)}{P(\{x\})}$$
$$\propto \frac{1}{(\lambda Z(\lambda))^N} \exp\Big(- \sum_1^N x_n/\lambda \Big) P(\lambda)$$

5. Define the likelihood function $P(\{x\}|\lambda)$ and plot $P(\{x\} = \{1.5, 2, 3, 4, 5, 12\}|\lambda)$ as a function of λ. Estimate the peak posterior value of λ and the error on λ by fitting to a gaussian at the peak.

In [ ]:

```
def likelihoodP(x, L):
    'The likelihood function given a dataset (x array) and a characteristic leng
th L'
    ...
    return ...
```

In [ ]:

```
# Create an array for L. Assume that it is evenly spaced numbers over the interv
al (1e-1, 1e2).
L = np.logspace(-1, 2, 1000)
# Create an array for x.
x = ...

# Evaluate the likelihood function and plot it as a function of L
P = ...

# Make plot
plt.semilogx( ... )

plt.xlim(1.e-1,1.e+2)
plt.ylim(0, 1.4e-6)
plt.xlabel(' ... ')
plt.ylabel(' ... ')
plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
plt.show()
```

In [ ]:

```
# Estimate the peak posterior value of L (Hint - https://docs.scipy.org/doc/nump
y/reference/generated/numpy.argmax.html)
max_L = ...

print("The peak posterior value of the characteristic length is = ", max_L)
```

In [ ]:

```
# Estimate the error on L  by fitting to a gaussian at the peak
# Import packages for curve fitting
from scipy.optimize import curve_fit

# Create an array of L near L_max
L = np.linspace(max_ind-0.2, max_ind+0.2, 100)
x = ...

# Define Gaussian function with arbitrary amplitude (See https://en.wikipedia.or
g/wiki/Normal_distribution)
def gaussian(x, Amp, mu, sig):
    return ...

# Fit a Gaussian function to a data
#(https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.
html)
# You can use different packages if you wish. This is only a suggestion.
popt, pcov = curve_fit( ... )

# Plot both data and fit
plt.plot( ... , 'b-', label = 'likelihood')
plt.plot( ... , 'r--', label='fit')
plt.legend()
plt.show()

Error = ...
print("The error on L is estimated to be = ", Error)
```

Problem 2 - Biased Coin

When spun on edge 256 times, a Belgian one-euro coin came up heads 142 times and tails 114. Do these data give evidence that the coin is biased rather than fair?

We compare the models $\mathcal{H}_0$ - the coin is fair - and $\mathcal{H}_1$ - the coin is biased.

First, suppose that the model $\mathcal{H}_1$ assumes a uniform prior distribution for p (the probability of getting heads in a single toss): $P(p|\mathcal{H}_1) = 1$.

Let the data D be a sequence which contains counts of the two possible outcomes (H - head / T - tail): e.g. HHTHT, HHHTTHTT, etc.

Given a particular p, the probability that F tosses results in a sequence D of F_H heads and F_T tails is:

$$P(D|p, \mathcal{H}_1) = p^{F_H}(1-p)^{F_T}.$$

Then,

$$P(D|\mathcal{H}_1) = \int_0^1 dp\, p^{F_H}(1-p)^{F_T} = \frac{\Gamma(F_H+1)\Gamma(F_T+1)}{\Gamma(F_H+F_T+2)}.$$

Note that the above integral is a "Beta function" $B(F_H+1, F_T+1)$ and can be written in terms of the gamma function. (See http://www.math.uah.edu/stat/special/Beta.html (http://www.math.uah.edu/stat/special/Beta.html))

The gamma function is an extension of the factorial function $\Gamma(n+1) = n!$

$$\frac{\Gamma(F_H+1)\Gamma(F_T+1)}{\Gamma(F_H+F_T+2)} = \frac{F_H!F_T!}{(F_H+F_T+1)!}$$

Similarly,

$$P(D|\mathcal{H}_0) = \left(\frac{1}{2}\right)^F.$$

1. Find the likelihood ratio $\frac{P(D|\mathcal{H}_1)}{P(D|\mathcal{H}_0)}$, assuming the uniform prior of $\mathcal{H}_1$. Which model does the data favor?

(Hint: If the argument of the gamma function is large, math.gamma() overflows. You can prevent this by using the fact:

$$log(xy/z) = log(x) + log(y) - log(z)$$

Then, you can evaluate $P = \Gamma(x) * \Gamma(y)/\Gamma(z)$ in the following way:

$$Q = log(P) = log(\Gamma(x)) + log(\Gamma(y)) - log(\Gamma(z))$$
$$P = e^Q$$

You can easily evaluate logarithm of the gamma function using "lgamma" (from math import lgamma) see https://docs.python.org/2/library/math.html (https://docs.python.org/2/library/math.html))

(Hint2: For reference, you can read: https://en.wikipedia.org/wiki/Bayes_factor (https://en.wikipedia.org/wiki/Bayes_factor))

In [ ]:

```
F = ...; F_H = ...; F_T = F - F_H

from math import factorial, log, exp, lgamma

Likelihood_H1 = ...
Likelihood_H0 = ...

ratio = ...

print("The likelihood ratio is = ", ratio)
print("The data give evidence in favor of ...")
```

Instead of assuming a uniform prior, suppose that we add a small bias, and consequently the prior were presciently set:

$$P(p|\mathcal{H}_1, \alpha) = \frac{1}{Z(\alpha)} p^{\alpha-1}(1-p)^{\alpha-1}, \text{ where } Z(\alpha) = \Gamma(\alpha)^2/\Gamma(2\alpha)$$

2. Find the likelihood ratio $\frac{P(D|\mathcal{H}_1)}{P(D|\mathcal{H}_0)}$, assuming the above prior of $\mathcal{H}_1$. Let α = \{ .37, 1.0, 2.7, 7.4, 20, 55, 148, 403, 1096 \}.

Answer:

In [ ]:

```
alpha = ...

def likelihood_ratio(F_H, F_T, alpha):
    ...
    return ...

ratio = np.zeros_like(alpha)
for i in range(len(alpha)):
    ratio[i] = likelihood_ratio(F_H, F_T, alpha[i])

print("For alpha = ", alpha, ", the likelihood ratios are = ", np.around(ratio,
decimals=2), "respectively.")
```

3. Does the likelihood ratio for $\mathcal{H}_1$ over $\mathcal{H}_0$ increases as α increases?

Answer:

4. Now, let $\mathcal{H}_1$ be the model in which the probability of getting heads is descrete at 142/256. What is the likelihood in this case?

In [ ]:

```
Likelihood_H1 = ...
Likelihood_H0 = ...

ratio = ...

print("The likelihood ratio is = ", ratio)
```

5. Explain the above result.

Answer:

6. Now let us test the null hypothesis. Assuming the central limit theorem, we model the binomial as a gaussian centered at $\mu = F/2$ and with the width given by $\sigma^2 = F * (p_{heads}) * (p_{tails})$. (in this case, $p_{heads} = p_{heads} = 1/2$)

In [ ]:

```
# Define mu (mean) and sigma (square root of the variance)
mu = ...
sigma = np.sqrt( ... )

# Calculate Z.
Z = abs(F_H-mu)/sigma

print("F_H is %.2f sigma away from the mean." %Z)

# Integrate a normal distribution from x=F_H to x=np.inf (See https://en.wikiped
ia.org/wiki/Normal_distribution)
def gaussian_normalized(x, mu, sigma):
    return ...

pvalue = quad( ... )[0]*100

print("The p-value is %.2f percent." %pvalue)
```

Problem 3 - Monty Hall

On a game show, a contestant is told the rules as follows:

There are three doors, labelled 1, 2, 3. A single prize has been hidden behind one of them. You get to select one door. Initially your chosen door will not be opened. Instead, the gameshow host will open one of the other two doors, and he will do so in such a way as not to reveal the prize. For example, if you first choose door 1, he will then open one of doors 2 and 3, and it is guaranteed that he will choose which one to open so that the prize will not be revealed.

At this point, you will be given a fresh choice of door: you can either stick with your first choice, or you can switch to the other closed door. All the doors will then be opened and you will receive whatever is behind your final choice of door.

Imagine that the contestant chooses door 1 first; then the gameshow host opens door 2, revealing nothing behind the door, as promised. Should the contestant (a) stick with door 1, or (b) switch to door 3, or (c) does it make no difference?

Let $\mathcal{H}_i$ denote the hypothesis that the prize is behind door i. We make the following assumptions: the three hypotheses $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3$ are equiprobable a *priori*, i.e.,

$$P(\mathcal{H}_1) = P(\mathcal{H}_2) = P(\mathcal{H}_3) = \frac{1}{3}$$

The datum we receive, after choosing door 1, is one of D = 3 and D = 2 (meaning door 3 or 2 is opened, respectively).
1. Find $P(D = 2|\mathcal{H}_1), P(D = 3|\mathcal{H}_1), P(D = 2|\mathcal{H}_2), P(D = 3|\mathcal{H}_2), P(D = 2|\mathcal{H}_3), P(D = 3|\mathcal{H}_3)$.

Answer:

Now, using Bayes' theorem, we evaluate the posterior probabilities of the hypotheses:

$$P(\mathcal{H}_i|D = 2) = \frac{P(D = 2|\mathcal{H}_i)P(\mathcal{H}_i)}{P(D = 2)}$$

2. First, we need to calculate the normalizing constant (denominator). Find $P(D = 2), P(D = 3)$

Answer:

3. Evaluate the posterior probability and argue if the contestant should switch to door 3.

Alternatively, you can perform a thought experiment in which the game is played with 100 doors. The rules are now that the contestant chooses one door, then the game show host opens 98 doors in such a way as not to reveal the prize, leaving the contestant's selected door and one other door closed. The contestant may now stick or switch. Where do you think the prize is?

Answer:

Imagine that the game happens again and just as the gameshow host is about to open one of the doors a violent earthquake rattles the building and one of the three doors flies open. It happens to be door 3, and it happens not to have the prize behind it. The contestant had initially chosen door 1.

Repositioning his toupee, the host suggests, 'OK, since you chose door 1 initially, door 3 is a valid door for me to open, according to the rules of the game; I'll let door 3 stay open. Let's carry on as if nothing happened.' Should the contestant stick with door 1, or switch to door 2, or does it make no difference? Assume that the prize was placed randomly, that the gameshow host does not know where it is, and that the door flew open because its latch was broken by the earthquake.

[A similar alternative scenario is a gameshow whose confused host forgets the rules, and where the prize is, and opens one of the unchosen doors at random. He opens door 3, and the prize is not revealed. Should the contestant choose what's behind door 1 or door 2? Does the optimal decision for the contestant depend on the contestant's beliefs about whether the gameshow host is confused or not?]

If door 3 is opened by an earthquake, the inference comes out differently – even though visually the scene looks the same. The nature of the data, and the probability of the data, are both now different. The possible data outcomes are, firstly, that any number of the doors might have opened. We could label the eight possible outcomes **d** = (0,0,0),(0,0,1),(0,1,0),(1,0,0),(0,1,1),...,(1,1,1).

Secondly, it might be that the prize is visible after the earthquake has opened one or more doors. So the data D consists of the value of **d**, and a statement of whether the prize was revealed. It is hard to say what the probabilities of these outcomes are, since they depend on our beliefs about the reliability of the door latches and the properties of earthquakes, but it is possible to extract the desired posterior probability without naming the values of $P(\mathbf{d}|\mathcal{H}_i)$ for each **d.**

All that matters are the relative values of the quantities $P(D|\mathcal{H}_1)$, $P(D|\mathcal{H}_2)$, $P(D|\mathcal{H}_3)$, for the value of D that actually occurred. The value of D that actually occurred is '**d** = (0, 0, 1), and no prize visible'.

4. How does $P(D|\mathcal{H}_1)$ compare with $P(D|\mathcal{H}_2)$? What is $P(D|\mathcal{H}_3)$? Find $P(D|\mathcal{H}_1)/P(D)$ and $P(D|\mathcal{H}_2)/P(D)$.

Answer:

5. Evaluate the posterior probability and argue if the contestant should switch.

Answer:

To Submit

Execute the following cell to submit. If you make changes, execute the cell again to resubmit the final copy of the notebook, they do not get updated automatically.
We recommend that all the above cells should be executed (their output visible) in the notebook at the time of submission.
Only the final submission before the deadline will be graded.

```
In [ ]:
```

```
_ = ok.submit()
```