Week 2 - Classical information in a nutshell

(c) Ariel Guerreiro 2023

In [ ]:
```
import numpy as np
import scipy as sp
from scipy.stats import entropy

import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np #imports the numpy package which has various tools that help
import math
# Importing standard Qiskit libraries
from qiskit.visualization import array_to_latex

import random
%matplotlib inline
#%config InlineBackend.figure_format = 'svg'
```

Question 1: Given the conditional probability of two variables, is it possible to recover the joint probability? How?

Yes it is possible, from the definition conditional probability

Question 2: Given the marginal probability, is it possible to infer any information about the joint and conditional probability distributions of two random variables?

Its not possible because we don't know how the random variables are related to one another.

Ensembles for multivariable probability distributions

So far we have represented joint distribution of two variables with the help of a rank-2 array, a matrix if you like, say:

$$\begin{bmatrix} 0.1 & 0.3 & 0.2 \\ 0.2 & 0.2 & 0.3 \\ 0.3 & 0.1 & 0.2 \end{bmatrix}$$

This approach could be extended to joint distributions of more variable by increasing the rank of the array, but this is not always the most practical. Here is an alternative:

In [ ]:
```
X = ['A', 'B', 'C']    # possible values of the first random variable
Y = [1, 2]             # possible values of the second random variable
Z = ['Yes', 'No']      # possible values of the third random variable

# create a dictionary to represent the output space
output_space = {}
for x in X:
    for y in Y:
        for z in Z:
            # assign a probability of 1/12 to each possible combination of outcomes
            output_space[(x,y,z)] = 1/12

# print the output space
print(output_space)
```

```
{('A', 1, 'Yes'): 0.08333333333333333, ('A', 1, 'No'): 0.08333333333333333, ('A', 2, 'Yes'): 0.08333333333333333, ('A', 2, 'No'): 0.08333333333333333, ('B', 1, 'Yes'): 0.0833
3333333333333, ('B', 1, 'No'): 0.08333333333333333, ('B', 2, 'Yes'): 0.08333333333333333, ('B', 2, 'No'): 0.08333333333333333, ('C', 1, 'Yes'): 0.08333333333333333, ('C', 1,
'No'): 0.08333333333333333, ('C', 2, 'Yes'): 0.08333333333333333, ('C', 2, 'No'): 0.08333333333333333}
```

This code defines a three-partite random variable with three possible values for the first variable ('A', 'B', and 'C'), two possible values for the second variable (1 and 2), and two possible values for the third variable ('Yes' and 'No'). The output space is represented as a dictionary, where each key is a tuple representing a possible combination of outcomes, and each value is the corresponding probability. In this case, each possible combination of outcomes has a probability of 1/12, since there are 322=12 possible outcomes in total.

This type of structure

Quantifying information

In a probability distribution, there are several quantities that can be used to quantify and characterize the information. These quantities include entropy, mutual information, Kullback-Leibler divergence, and cross-entropy.

The entropy of a probability distribution is a measure of the amount of uncertainty associated with a random variable. It is defined as:

$$H(X) = -\sum_{x_i \in A_X} p_i \log p_i$$

where p_i is the probability of observing the i-th state x_i of the random variable X.

The mutual information is a measure of the amount of shared information between two random variables. It is defined as:

$$I(X;Y) = \sum_{x_i \in A_X} \sum_{y_j \in A_Y} p(x_i, y_j) \log \left(\frac{p(x_i, y_j)}{p_i p_j} \right)$$

where $p(x_i, y_j)$ is the joint probability of observing states x_i and y_j for random variables X and Y, respectively, and p_i and p_j are the marginal probabilities of observing states x_i and y_j, respectively.

The Kullback-Leibler divergence, also known as the relative entropy, is a measure of the difference between two probability distributions. It is defined as:

$$D_{KL}(P||Q) = \sum_{x_i \in A_X} p_i \log \left(\frac{p_i}{q_i} \right)$$

where p_i and q_i are the probabilities of observing state x_i for random variables P and Q, respectively.

The cross-entropy is a measure of the difference between a true probability distribution and an estimated probability distribution. It is defined as:

$$H(P, Q) = -\sum_{x_i \in A_X} p_i \log q_i$$

where p_i is the true probability of observing state x_i for random variable P, and q_i is the estimated probability of observing state x_i for random variable Q.

These quantities can be used to describe and analyze the information content of a probability distribution, and they play an important role in many applications of classical information theory.

Entropy

Entropy is a measure of uncertainty in a random variable. In other words, it quantifies the amount of information required to specify the state of the random variable.

The entropy of a discrete random variable with states $x_1, x_2, \ldots, x_n$ and probabilities $p_1, p_2, \ldots, p_n$ is given by:

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

Let's try to calculate the entropy of a simple example.

In [ ]:
```python
def entropy(p):
    """
    Calculates the entropy of a probability distribution p

    Parameters
    ----------
    p : numpy array
        Probability distribution

    Returns
    -------
    h : float
        Entropy of the distribution
    """
    h = - np.sum(p * np.log2(p))
    return h


#use example

p = np.array([0.25, 0.75])
ent = entropy(p)
print("The entropy of the random variable is {:.2f} bits".format(ent))
```

```
The entropy of the random variable is 0.81 bits
```

Problem 1: *Entropy of a classical random variable.*

Suppose we have a random variable X that takes on two possible values, 0 and 1, with equal probability. What is the entropy of X?

In [ ]:
```python
joint=np.array([0.5,0.5])
print(entropy(joint))
```

```
1.0
```

Solution:

The entropy of a random variable X is defined as:

$$H(X) = -\sum_{x \in \text{outcomes}} P(x) \log_2 P(x)$$

For this example, the two possible outcomes are 0 and 1, and each has a probability of 0.5. Hence,

$$H(X) = -\left(0.5 \log_2 0.5 + 0.5 \log_2 0.5\right) = -(0.5 \cdot -1 + 0.5 \cdot -1) = 1$$

So the entropy of the random variable X is 1 bit.

Mutual Information

Mutual information is a measure of the amount of shared information between two random variables. It quantifies the reduction in uncertainty of one random variable given the knowledge of another.

The mutual information between two discrete random variables X and Y with joint probability mass function $p_{X,Y}(x,y)$ is given by:

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p_{X,Y}(x,y) \log_2 \frac{p_{X,Y}(x,y)}{p_X(x) p_Y(y)}$$

Let's try to calculate the mutual information between two simple examples.

In [ ]:
```
def mutual_entropy(p_xy):
    """
    Calculates the mutual entropy between two random variables X and Y

    Parameters
    ----------
    p_xy : numpy array
        Joint probability distribution of X and Y

    Returns
    -------
    h_xy : float
        Mutual entropy between X and Y
    """

    print(p_xy)
    p_x = np.sum(p_xy, axis=1)
    p_y = np.sum(p_xy, axis=0)

    h_x = entropy(p_x)
    h_y = entropy(p_y)
    h_xy = entropy(p_xy.flatten())

    i_xy = h_x + h_y - h_xy

    return h_xy,i_xy
```

Problem 2: *Mutual information between two classical random variables.*

Suppose we have two random variables X and Y. X takes on two possible values, 0 and 1, with equal probability, and Y takes on two possible values, 0 and 1, with probabilities 0.7 and 0.3 respectively. What is the mutual entropy between X and Y?

$$P(X) = \begin{pmatrix} 0.5 & 0.5 \end{pmatrix} P(Y) = \begin{pmatrix} 0.7 & 0.3 \end{pmatrix}$$

In [ ]:
```
p_x=np.array([0.5,0.5])
p_y=np.array([0.7,0.3])

h,i=mutual_entropy(np.outer(p_x,p_y))
print("The mutual entropy is {}".format(h))
print("The mutual information is {}".format(i))
```

```
[[0.35 0.15]
 [0.35 0.15]]
The mutual entropy is 1.8812908992306925
The mutual information is 2.220446049250313e-16
```

Solution: The mutual entropy between two random variables X and Y is defined as:

$$H(X,Y) = -\sum_{x \in \text{outcomes of X}} \sum_{y \in \text{outcomes of Y}} P(x,y) \log_2 P(x,y)$$

We can calculate the joint probability of each combination of outcomes (0,0), (0,1), (1,0), and (1,1) and then plug it into the above formula. For example,

$$P(0,0) = P(X=0) \cdot P(Y=0) = 0.5 \cdot 0.7 = 0.35$$

Similarly, the other joint probabilities can be calculated as:

$$P(0,1) = 0.5 \cdot 0.3 = 0.15$$

$$P(1,0) = 0.5 \cdot 0.7 = 0.35$$

$$P(1,1) = 0.5 \cdot 0.3 = 0.15$$

Plugging these values into the formula for mutual entropy, we get:

$$H(X,Y) = -\left(0.35 \log_2 0.35 + 0.15 \log_2 0.15 + 0.35 \log_2 0.35 + 0.15 \log_2 0.15\right)$$

$$H(X,Y) = -(0.35 \cdot -1.67 + 0.15 \cdot -2.97 + 0.35 \cdot -1.67 + 0.15 \cdot -2.97)$$

$$H(X,Y) = 1.51$$

So the mutual entropy between X and Y is 1.51 bits.

Conditional Information

Conditional information is a measure of the amount of information that can be obtained about a random variable given the knowledge of another random variable. It is a fundamental concept in classical information theory that is closely related to entropy and mutual entropy.

In [ ]:
```python
def conditional_entropy(p_xy):
    """
    Calculates the conditional entropy of X given Y for a joint probability distribution of X and Y
    Parameters
    ----------
    p_xy : numpy array
    Joint probability distribution of X and Y

    Returns
    -------
    h_x_given_y : float
    Conditional entropy of X given Y
    """
    p_y = np.sum(p_xy, axis=1)
    p_x_given_y = np.divide(p_xy, p_y, out=np.zeros_like(p_xy), where=p_y!=0)
    h_x_given_y = np.sum(np.sum(np.multiply(p_xy, np.log2(p_x_given_y)), axis=0),axis=0)

    return -h_x_given_y
```

Problem 3: Consider two random variables X and Y, with joint probability distribution $P_{X,Y}$. Calculate the conditional entropy of Y given X for the following joint probability distribution:

$$\begin{bmatrix} 0.1 & 0.3 & 0.2 \\ 0.2 & 0.2 & 0.3 \\ 0.3 & 0.1 & 0.2 \end{bmatrix}$$

In [ ]:
```
joint=np.array([[0.1,0.3,0.2],[0.2,0.2,0.3],[0.3,0.1,0.2]])
print(conditional_entropy(joint))
```

```
2.818397953523025
```

Solution:

First, let's calculate the conditional probability $P_{Y|X}(y|x)$:

$$P_{Y|X}(y|x_1) = \frac{P_{X,Y}(x_1, y)}{P_X(x_1)} = \frac{[0.1, 0.3, 0.2][y]}{0.1 + 0.3 + 0.2} = [0.125, 0.375, 0.5]$$

$$P_{Y|X}(y|x_2) = \frac{P_{X,Y}(x_2, y)}{P_X(x_2)} = \frac{[0.2, 0.2, 0.3][y]}{0.2 + 0.2 + 0.3} = [0.25, 0.25, 0.5]$$

$$P_{Y|X}(y|x_3) = \frac{P_{X,Y}(x_3, y)}{P_X(x_3)} = \frac{[0.3, 0.1, 0.2][y]}{0.3 + 0.1 + 0.2} = [0.6, 0.2, 0.2]$$

Next, we can use these conditional probabilities to calculate the conditional entropy:

$$H(Y|X) = -\sum_{x \in X} \sum_{y \in Y} P_{X,Y}(x, y) \log P_{Y|X}(y|x)$$

$$H(Y|X) = -[0.1 \times \log(0.125) + 0.3 \times \log(0.375) + 0.2 \times \log(0.5)] - [0.2 \times \log(0.25) + 0.2 \times \log(0.25) + 0.3 \times \log(0.5)] - [0.3 \times \log(0.6) + 0.1 \times \log(0.2) + 0.2 \times \log(0.2)]$$

$$H(Y|X) = 0.276$$

So, the conditional entropy of Y given X is approximately 0.276.

Conclusions

Congratulations! You have just taken your first steps towards understanding some of the foundational concepts of modern information theory. The study of probability and classical information theory provides us with the tools to quantify and understand uncertainty, randomness, and information in the world around us. By learning about ensembles, joint and marginal probabilities, and conditional probability, you have gained a deeper understanding of how information is stored and transmitted.

As you continue your journey into the world of quantum information theory, you may have many questions in your mind. What is the relationship between classical information theory and quantum information theory? How do quantum mechanics influence the way we think about information? What are some of the unique challenges and opportunities that quantum information theory presents?

As you delve deeper into these topics, keep in mind the importance of the concepts you have learned. Probability theory provides the foundation for much of modern information theory, and it is crucial to have a strong understanding of these concepts in order to make progress in the field. Take these building blocks, and use them as a foundation for your future studies. Good luck!

Question 3: So, when you are manipulating information, are you working with distribution functions? I mean, when you send a Morse encoded message through a channel you are not sending the probability that the next letter is a "X" or a "Z". You are actually sending a specific letter and the receiver may receive, not receive, or receive mistakenly. How do you translate these situations into probability distributions? Provide your views on this question.

Yes, because the information i receive could have noisy information within it. So usually there is an associated distribution function, because in the example given, i could get the correct signal, no signal, and the wrong signal. In most lab works in my bachelors i had to remove the noise which came into the data, thankfully there are tools which can remove this interferences.

Question 4: In your own words, provide definitions and contextualization of the following concepts: ensemble, probability vector, probability, joint probability, conditional probability, entropy, mutual information, relative entropy and inference.

Ensemble is a set of properties about a system, in information theory, (x, A_X, P_X), where x is the outcome of a random variable X, which can have $A_X = \{x_1, x_2, \ldots, x_n\}$, and the probability associated with each outcome $P_X = \{P_1, P_2, \ldots, P_n\}$.

Probability vector is a set of terms, for which each index refers to a different outcome and the value of the index is the corresponding probability.

Probability is a measure by which an event is likely to happen. Its a number between 0 and 1. Its used in a variety of areas, physics, maths, finance.

Joint probability is the probability of 2 or more events happening at the same time, it is usually represented as a matrix of probability vectors for 2 random variables with a number of outcomes.

Conditional probability is the probability of an event occuring given another event occurred before it. Its usually denoted as P(A|B), where A is the event and B is the condition

Entropy is the measure of disorder, randomness and uncertainty in a system. In information theory, entropy is usually used to measure the amount of information contained in a message or signal.

Mutual information is the measure of information shared between 2 random variables.

Relative entropy is the measure of the difference between 2 systems.

Inference is the act of reaching an hypotheses based on the information given. Its often used to estimated parameters on observed data

Question 5: Underlying the concepts presented in this notebook is the notion of probability distribution of a random variable, whether that variable is a physical state or the result of a measurement, say $p(x)$. In future notebooks we shall use the notion of wave vector. Discuss what you expect a wave vector to be, what properties should it have and how do they differ from a probability vector?

I expect that a wave vector will not only contain information about the probability of a given outcome but also other underlying physical properties of the system, like the position, momentum, magnetic momentum, i also expect the vector to be normalized, and it differs from the probability vector in that way as well, the probability vector is not normalized, because each entry already represents the probablity.

Question 6: Given that wave vectors and probability vectors are distinct, how would you combine the the two objects into a single one combining the informational content of probability vector and the capacity to predict the probability of measurements of a wave vector?

Using the density operator, because it combines the content of a probability vector and it can predict measurements by enconding probability and the wave function.

Question 7: If you have a physics background then, you must be recognizing many concepts from statistical mechanics in classical information theory. Can you extrapolate on which other concepts can be adopted in information theory from statistical physics?

Percolation theory which is used in statistical physics more recently in the Ising model, for one of the most recent Fields Medal. Because it studies connected clusters in random networks, and can be used in communication networks.

Fluctuations, which was a topic in the notebook of the first week, it can be used to study the possible random fluctuations that can occur in the signal of a wireless network.

Phase space, which is used as mathematical space to describe the states of a system in terms of the position and momentum. And it can be used to describe all the possible signals that can be transmitted.

Question 8: Let now think the other way around. What concepts of information theory do you think can be borrowed by physics, specially quantum physics of complex systems? This question in not about providing a correct answer but rather to stimulate your brain muscles beyond the technical aspects. Let your mind fly.

I can only see a relation between mutual information and quantum entangled states

EXERCISES:

Today's exercises are probably about probabilities in classical information theory but trying to bridge to the quantum counterpart.

In [ ]:
```
class Main:
    def __init__(self,value,name):
        self.name=name
        self.prob=value
    def prob_vec(self,ax):
        return np.sum(self.prob,axis=ax)
    def entropy(self):
        return -np.sum(self.prob*np.log2(self.prob))
    def cond_entropy(self,other):
        return -np.sum(other.prob*np.log2(self.prob))
    def str(self,x):
        return "The entropy of the {} probability is {}".format(self.name,x)
    def _latex_(self):
        text_dis= "The distribution of {} is".format(self.name)
        print(text_dis)
        return array_to_latex(self.prob, prefix="")
    def cond_prob(self,ax):
        if ax==1:
            return self.prob/self.prob_vec(ax)[:,None]
        elif ax==0:
            return self.prob/self.prob_vec(ax)[None,:]
```

Exercise 1: *Calculation of entropy, mutual entropy and conditional information*

A system consists of two random variables, X and Y, with joint probability distribution given by the following table:

X / Y	0	1	2
0	0.1	0.2	0.1
1	0.1	0.3	0.2
2	0.2	0.1	0.3

Calculate the entropy of X, entropy of Y, and the mutual entropy of X and Y.

In [ ]:
```
joint=Main(np.array([[0.1,0.2,0.1],[0.1,0.3,0.2],[0.2,0.1,0.3]]),"P(X,Y)")
px=Main(joint.prob_vec(1),"P_X") #0 for y #1 for x
py=Main(joint.prob_vec(0),"P_Y")

print(px.str(px.entropy()))
print(py.str(py.entropy()))
print(joint.str(joint.entropy()))
```

```
The entropy of the P_X probability is 1.4131299509543922
The entropy of the P_Y probability is 1.4131299509543922
The entropy of the P(X,Y) probability is 3.764107451387086
```

Exercise 2: *Advanced entropy and mutual entropy calculations*

Consider a discrete random variable X with the following probability distribution:

$$P(X=0) = \frac{1}{3}$$

$$P(X=1) = \frac{2}{3}$$

and another discrete random variable Y with the following probability distribution:

$$P(Y=0) = \frac{1}{2}$$

$$P(Y=1) = \frac{1}{2}$$

Calculate the entropy of X, entropy of Y, the joint entropy of X and Y, and the conditional entropy of X given Y.

In [ ]:
```
p_x=np.array([1/3,2/3])
p_y=np.array([1/2,1/2])
px=Main(p_x,"P_X")
py=Main(p_y,"P_Y")
print(px.str(px.entropy()))
print(py.str(py.entropy()))
p_xy=np.outer(p_x,p_y)
pxy=Main(p_xy,"P(X,Y)")
print(pxy.str(pxy.entropy()))
p_x_given_y=pxy.cond_prob(0)
px_given_y=Main(p_x_given_y,"P(X|Y)")
print(px_given_y.str(px_given_y.cond_entropy(pxy)))
```

```
The entropy of the P_X probability is 0.9182958340544896
The entropy of the P_Y probability is 1.0
The entropy of the P(X,Y) probability is 1.9182958340544896
The entropy of the P(X|Y) probability is 0.9182958340544896
```

In [ ]:
```
px_given_y._latex_()
```

```
The distribution of P(X|Y) is
```

Out[ ]:

$$\begin{bmatrix} \frac{1}{3} & \frac{1}{3} \\ \frac{2}{3} & \frac{2}{3} \end{bmatrix}$$

Exercise 4: *Calculation of Marginal and Conditional Probabilities*

Suppose we have a joint probability distribution $P(X,Y)$ with the following values:

$$P(X=0,Y=0)=0.2$$

$$P(X=0,Y=1)=0.1$$

$$P(X=1,Y=0)=0.3$$

$$P(X=1,Y=1)=0.15$$

$$P(X=2,Y=0)=0.15$$

$$P(X=2,Y=1)=0.05$$

Task: Calculate the marginal probability distributions $P(X)$ and $P(Y)$ and the conditional probability distributions $P(X|Y)$ and $P(Y|X)$.

In [ ]:
```
joint_=np.array([[0.2,0.1],[0.3,0.15],[0.15,0.05]])
joint=Main(joint_,"P(X,Y)")
p_x=joint.prob_vec(1)
p_y=joint.prob_vec(0)
px=Main(p_x,"P(X)")
py=Main(p_y,"P(Y)")
p_x_given_y=joint.cond_prob(0)
p_y_given_x=joint.cond_prob(1)
pxgiveny=Main(p_x_given_y,"P(X|Y)")
pygivenx=Main(p_y_given_x,"P(Y|X)")
```

In [ ]:
```
px._latex_()
```

The distribution of P(X) is

Out[ ]:

$$\left[\frac{3}{10} \quad \frac{9}{20} \quad \frac{1}{5}\right]$$

In [ ]:
```
py._latex_()
```

The distribution of P(Y) is

Out[ ]:

$$\left[\frac{13}{20} \quad \frac{3}{10}\right]$$

In [ ]:
```
pxgiveny._latex_()
```

The distribution of P(X|Y) is

Out[ ]:

$$\begin{bmatrix} \frac{4}{13} & \frac{1}{3} \\ \frac{6}{13} & \frac{1}{2} \\ \frac{3}{13} & \frac{1}{6} \end{bmatrix}$$

In [ ]:
```
pygivenx._latex_()
```

The distribution of P(Y|X) is

Out[ ]:

$$\begin{bmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{2}{3} & \frac{1}{3} \\ \frac{3}{4} & \frac{1}{4} \end{bmatrix}$$

Exercise 5: *Calculating Joint Probabilities*

Suppose you have two random variables X and Y, where X can take on one of two values, 0 or 1, and Y can take on one of three values, 0, 1, or 2. The joint probability mass function of X and Y is given as follows:

$$P(X=0,Y=0)=0.1$$

$$P(X=0,Y=1)=0.2$$

$$P(X=0,Y=2)=0.05$$

$$P(X=1,Y=0)=0.05$$

$$P(X=1,Y=1)=0.1$$

$$P(X=1,Y=2)=0.15$$

Task: Write a Python code to calculate the joint probabilities.

In [ ]:
```
px0y=np.array([0.1,0.2,0.05])
px1y=np.array([0.05,0.1,0.15])
p_xy=Main([px0y,px1y],"P(X,Y)")
p_xy._latex_()
```

```
The distribution of P(X,Y) is
```

Out[ ]:

$$\begin{bmatrix} \frac{1}{10} & \frac{1}{5} & \frac{1}{20} \\ \frac{1}{20} & \frac{1}{10} & \frac{3}{20} \end{bmatrix}$$

Exercise 6: *Calculation of Hypothesis Probability*

Suppose we have a set of random variables X, Y, and Z, where X can take on values from $1, 2, 3$, Y can take on values from $1, 2, 3, 4$, and Z can take on values from $0, 1$. We also have a joint probability mass function (PMF) $P(X, Y, Z)$ defined as follows:

$$
\begin{aligned}
P(X = 1, Y = 1, Z = 0) &= 0.05 \\
P(X = 1, Y = 2, Z = 0) &= 0.1 \\
P(X = 1, Y = 2, Z = 1) &= 0.1 \\
P(X = 1, Y = 3, Z = 0) &= 0.05 \\
P(X = 2, Y = 1, Z = 1) &= 0.1 \\
P(X = 2, Y = 2, Z = 0) &= 0.05 \\
P(X = 2, Y = 3, Z = 0) &= 0.1 \\
P(X = 2, Y = 4, Z = 1) &= 0.05 \\
P(X = 3, Y = 1, Z = 0) &= 0.1 \\
P(X = 3, Y = 3, Z = 1) &= 0.05 \\
P(X = 3, Y = 4, Z = 0) &= 0.05
\end{aligned}
$$

The exercise is to calculate the probability of the hypothesis $H = (X, Y, Z) : X = 1, Y = 2, Z = 1$.

In [ ]:
```
prob=dict()
prob["110"]=0.05
prob["120"]=0.1
prob["121"]=0.1
prob["130"]=0.05
prob["211"]=0.1
prob["220"]=0.05
prob["230"]=0.1
prob["241"]=0.05
prob["310"]=0.1
prob["311"]=0.05
prob["340"]=0.05

H=prob["121"]
print(H)
```

```
0.1
```

Exercise 7: *Hypothesis testing using the Bayes' theorem*

A disease is affecting a certain population. The probability of getting the disease is 1%. A diagnostic test for the disease is performed and is known to be 90% accurate. That means, if a person has the disease, the test will return positive result in 90% of the cases, and if a person does not have the disease, the test will return negative result in 90% of the cases.

A person goes through the test and the result is positive. What is the probability that the person has the disease?

We can express the conditional probability result of the test given the pacient is sick or healthy,

$$\mathrm{P}(\mathrm{Result\ of\ test}|\mathrm{State\ of\ the\ pacient}) = \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix}$$

And the marginal probability of the state of the pacient is

$$\mathrm{P}(\mathrm{State\ of\ the\ pacient}) = \begin{pmatrix} \mathrm{sick} & \mathrm{healthy} \end{pmatrix} = \begin{pmatrix} 0.01 & 0.99 \end{pmatrix}$$

Calculating the conditional probability of $P(\mathrm{Sick}|\mathrm{Positive})$ is not easy, because the random variables depend on each other, so we are going to use the Bayes's Theorem, which states that,

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{P(B|A)P(A)}{\sum_i P(B|A_i)P(A_i)}$$

In [ ]:
```
p_rs=np.array([[0.9,0.1],[0.1,0.9]]) #r stands for result and s for state
p_s=np.array([0.01,0.99])

p_s_given_r=p_rs[0]*p_s/(np.dot(p_rs,p_s))
print(np.dot(p_rs,p_s))
print(p_s_given_r)
```

```
[0.108 0.892]
[0.08333333 0.11098655]
```

Exercise 8: *Using the Bayes' theorem to update the probability of a hypothesis in the context of classical communications*

Consider a binary communication channel with a Bernoulli input distribution, where a binary data symbol "0" or "1" is transmitted over the channel with probability p and $q = 1 - p$, respectively. Suppose the channel is modeled as a binary symmetric channel (BSC), meaning that the received symbol is different from the transmitted symbol with probability e.

Suppose we transmit the symbol "0" and receive the symbol "1". We want to calculate the probability that the transmitted symbol was "0" given that the received symbol is "1". This is equivalent to finding the probability $P(X = 0|Y = 1)$, where X is the transmitted symbol and Y is the received symbol.

$$\mathrm{P}(\mathrm{Transmitted}) = [p, 1-p]$$

$$\mathrm{P}(\mathrm{Received}|\mathrm{Transmitted}) = \begin{pmatrix} 1-e & e \\ e & 1-e \end{pmatrix}$$

Bayes's Theorem

$$\begin{aligned}
&\mathrm{P}(\mathrm{Transmitted{=}0}|\mathrm{Received{=}1}) && (1)\\
&= \frac{\mathrm{P}(\mathrm{Received{=}1}|\mathrm{Transmitted{=}0})\mathrm{P}(\mathrm{Transmitted{=}0})}{\mathrm{P}(\mathrm{Received{=}1})} && (2)\\
&= \frac{\mathrm{P}(\mathrm{Received{=}1}|\mathrm{Transmitted{=}0})\mathrm{P}(\mathrm{Transmitted{=}0})}{(\mathrm{P}(\mathrm{Received{=}1}|\mathrm{Transmitted{=}0})\mathrm{P}(\mathrm{Transmitted{=}0}){+}\mathrm{P}(\mathrm{Received{=}1}|\mathrm{Transmitted{=}1})\mathrm{P}(\mathrm{Transmitted{=}1}))} && (3)\\
&= \frac{(e)(p)}{[(e)(p) + (1-e)(1-p)]} && (4)
\end{aligned}$$

Exercise 9: *Maximum Likelihood Inference*

Consider a communication channel where the transmission of bits is modeled as a Bernoulli distribution. In this channel, if a 1 bit is transmitted, the receiver gets a 1 with probability p, and if a 0 bit is transmitted, the receiver gets a 0 with probability q. Let X1, X2, ..., Xn be the transmitted bits and Y1, Y2, ..., Yn be the received bits.

Suppose we want to estimate the probability of receiving a 1 given that a 1 was transmitted, denoted as p. We can do this by using the maximum likelihood method.

Exercise 10: *Maximum Likelihood Inference*

Given the observed received bits Y = [1, 1, 0, 1, 0, 0, 1], find the maximum likelihood estimate for p.

$Received = Y = [1, 1, 0, 1, 0, 0, 1]$

$$\mathrm{P}(\mathrm{Received}|\mathrm{Transmitted}) = \begin{bmatrix} q & 1-q \\ 1-p & p \end{bmatrix}$$

The likelihood function is given by:

$$L(p) = p^k (1-p)^{n-k}$$

where k is the number of ones observed in the received bits and n is the total number of bits transmitted.

And so we have,

$$L(p) = p^4 (1-p)^{(7-4)} = p^4 (1-p)^{(3)}$$

The goal is to find the value of p that maximizes the likelihood function:

$$\frac{dL(p)}{dp} = 4p^3(1-p)^{(3)} - 3(1-p)^2 p^4 = 0 \Leftrightarrow 4(1-p) = 3p \Leftrightarrow 4 = p(3+4) \Leftrightarrow p = 4/7 = k/n$$

Exercise 11: *Bayesian Inference*

A box contains 4 balls, of which 3 are red and 1 is blue. A ball is picked from the box randomly and is not replaced. The person then picks a second ball. What is the probability that both balls are blue?

As the first ball isn't replaced and there only exist 1 blue ball, the probability is 0 :)

Exercise 12: *Inferring the parameters of a Poisson distribution**

In this exercise, you will use maximum likelihood estimation and Bayesian inference to infer the parameter of a Poisson distribution.

Suppose that the number of events in a given time period follows a Poisson distribution with an unknown parameter λ. The likelihood function for a sample of n events is given by:

$$L(\lambda) = \prod_{i=1}^{n} \frac{e^{-\lambda} \lambda^{x_i}}{x_i!}$$

where x_i is the number of events observed in the ith time period.

Find the maximum likelihood estimate of λ by taking the derivative of the log-likelihood function with respect to λ, setting it to zero, and solving for λ.

Write a python function to perform Bayesian inference for λ using a uniform prior. Draw samples from the posterior distribution using Markov Chain Monte Carlo (MCMC) and compare the results to the maximum likelihood estimate.

$$\frac{d\log(L(\lambda))}{d\lambda} = \frac{d\sum_{i=1}^{n}\log(\frac{e^{-\lambda}\lambda^{x_i}}{x_i!})}{d\lambda} = \frac{d\sum_{i=1}^{n}(-\lambda + x_i\log(\lambda) - \log(x_i!))}{d\lambda} = \sum_{i=1}^{n}\frac{d(-\lambda + x_i\log(\lambda) - \log(x_i))}{d\lambda} = \sum_{i=1}^{n}(-1 + x_i/\lambda) = -n + (\sum_{i=1}^{n}x_i)/\lambda = 0 \Leftrightarrow \lambda = (\sum_{i=1}^{n}x_i)/n$$

Exercise 13: *Discrete Variables Inference*

Consider a discrete random variable X that can take values {0,1,2,3} with probabilities {0.1, 0.3, 0.3, 0.3}. Suppose that we observe X=2 and we want to update the probability of X given this observation.

Calculate the maximum likelihood estimate (MLE) of X given the observation X=2. Calculate the Bayesian estimate of X given the observation X=2 and a prior probability {0.05, 0.2, 0.4, 0.35}.

$$P(X|2) = \frac{P(2|X)P(X)}{P(2)}$$

Exercise 14: *Quantum State Discrimination*

Consider an unknown quantum gate that transforms an input state X into an output state Y. The input state X is a quantum state in the set of states for a two-level system, and the probability of sending X through the gate is known, given by p(X). The output state Y is not known to the observer. The observer measures the observable operator Mz = |1><1|-|0><0|, yielding a random variable Z. The experiment is performed several times to obtain the frequency probability distribution p(z) at the arrival.

Task:

Define the joint probability distribution of X, Y, and Z. Calculate the marginal probability distributions of X and Z. Calculate the conditional probability distribution of X given Z. Compare the results of the marginal and conditional probability distributions with the frequency probability distribution obtained from the experiment.

Note: This exercise can be extended to consider more complex quantum systems, such as multi-level systems, and to explore the concepts of entropy, mutual entropy, and information gain in quantum systems.

Input X we have output Y, $AX = Y$

As we know that

$$P(Z|X,Y) = \frac{P(X,Y,Z)}{P(X,Y)} = \frac{P(X,Y,Z)}{P(Y|X)P(X)} \Leftrightarrow P(X,Y,Z) = P(Z|X,Y)P(Y|X)P(X)$$

by the Born rule we have that

$$p(X) = tr\left(\rho U_X\right)$$

we can relate this to

$$P(Z|X) = tr\left(M_Z X A\right) = tr\left(M_Z Y\right)$$

The probability of obtaining a measurement result Z depends only on the input state X and the output state Y, and is independent of how the output state was obtained. Therefore $P(Z|X,Y) = P(Z|X)$

$$P(X,Y,Z) = p(X)P(Y|X)tr\left(M_Z Y\right)$$

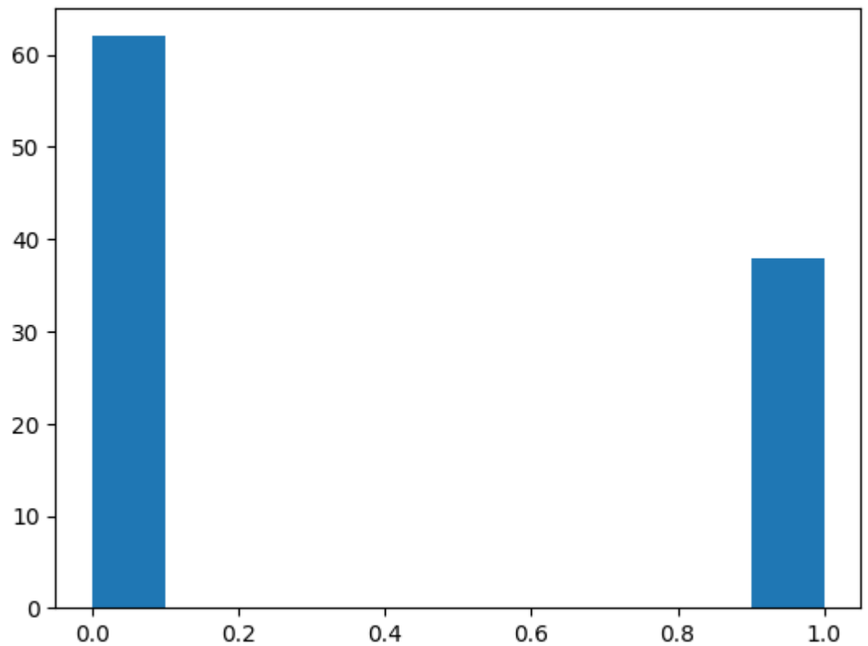
Exercise 15: *The black box problem*

This is an interesting exercise! Here is a code for the "black-box" function in Python that generates the random quantum states and applies either the "not gate" or the "identity gate" to it, and measures the resulting state. This function can be used to test different methods of inference to determine the type of gate that was used.

In [ ]:
```python
def quantum_black_box(n, distribution):
    """
    This function generates n instantiations of a random variable X, corresponding to a quantum state of a two level system.
    The superposition coefficients are generated following the given distribution. The function then applies either the
    "not gate" or the "identity gate" to the state X and measures the resulting state.
    :param n: number of instantiations
    :param distribution: a list of the form [p0, p1] containing the probabilities for the superposition coefficients
    :return: an array of n measurements
    """
    results = np.zeros(n)
    for i in range(n):
        # Generate the superposition coefficients based on the provided distribution
        p0, p1 = distribution
        prob = random.uniform(0, 1)
        if prob <= p0:
            alpha = np.sqrt(p0)
            beta = np.sqrt(1 - p0)
        else:
            alpha = np.sqrt(p1)
            beta = np.sqrt(1 - p1)

        # Choose either the "not gate" or the "identity gate" to apply to the state X + cz + HADAMART
        gate = random.choice([np.array([[0, 1],[1, 0]]), np.identity(2),np.array([[1, 0],[0, -1]]),(1/np.sqrt(2))*np.array([[1, 1],[1, -1]])])

        # Calculate the resulting state Y
        state = np.array([alpha, beta])
        state = np.matmul(gate, state)

        # Measure the resulting state using the observable Mz
        prob_0 = np.abs(state[0])**2
        prob_1 = np.abs(state[1])**2
        if random.uniform(0, 1) <= prob_0:
            results[i] = 0
        else:
            results[i] = 1

    return results
```

In [ ]:
```python
plt.hist(quantum_black_box(100,[0.7,0.3]))
```

Out[ ]:
```
(array([62.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0., 38.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <BarContainer object of 10 artists>)
```



The exercise can be extended by considering more complex quantum states, different quantum channels, or exploring the relationship between the entropy difference and the noise in the channel.

Today's afterthoughts:

Reflect on the meaning of classical information as a probability distribution or as an entropy. Can you think of limitations to these approaches? Why do you think that quantum information is not just applying classical information to random variables identified with quantum states? What should be missing, if anything, in a classical information theory about quantum states to become a full quantum theory?