2110203 COMP ENG MATH II (CEDT) -- Stats Homework I

Instructions:

- Only ASSIGNMENT 1 and 2 will be graded. The rest are for your practise.
- Submit as PDF file. You can create PDF file using File -> Print from the Google Colab menu.

```
import numpy as np
import matplotlib.pyplot as plt
import math, random
```

Sampling

Sampling is a process that is very important for writing simulations. In this section, you will try to sample from some common distributions. You may implement them yourself or use the provided distribution from `scipy.stats`.

```
from scipy.stats import norm, bernoulli, binom, uniform, geom, expon

# Sample from Uniform(a, b)
def sample_uniform(sample_size, a, b):
  # [YOUR CODE HERE]
  dist = uniform(a, b)
  return dist.rvs(sample_size)

def sample_normal(sample_size, mu, sigma):
  # [YOUR CODE HERE]
  dist = norm(mu, sigma)
  return dist.rvs(sample_size)

def sample_bernoulli(sample_size, p):
  # [YOUR CODE HERE]
  dist = bernoulli(p)
  return dist.rvs(sample_size)

def sample_binomial(sample_size, n, p):
  # [YOUR CODE HERE]
  dist = binom(n, p)
  return dist.rvs(sample_size)

def sample_geometric(sample_size, p):
  # [YOUR CODE HERE]
  dist = geom(p)
  return dist.rvs(sample_size)

def sample_exponential(sample_size, l):
  # [YOUR CODE HERE]
  dist = expon(l)
  return dist.rvs(sample_size)
```

ASSIGNMENT 1

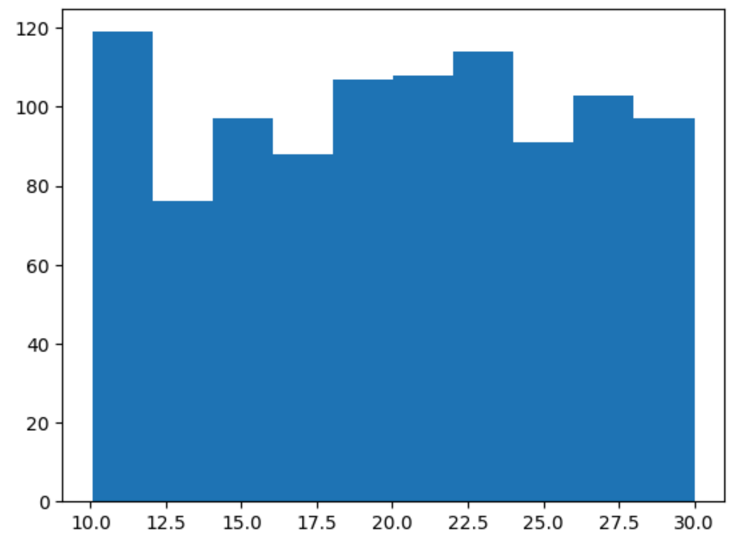
Hamtaro and his friends are collecting sunflower seeds. The bigger the sunflower, the more seeds they can find! The probability of finding a sunflower of a certain height x (in cm, from 0 to 10) increases with its height, following the probability density function $f(x) = \frac{x}{50}$. Write a function `sample_increasing(sample_size)` to simulate the heights of the sunflowers the Ham-Hams find.

```
# ASSIGNMENT 1
# sample from pdf f(x)=x/50, 0<=x<=10
def sample_increasing(sample_size):
  # [YOUR CODE HERE]
  u = np.random.rand(sample_size)
  dist = np.sqrt(100 * u)
  return dist
```

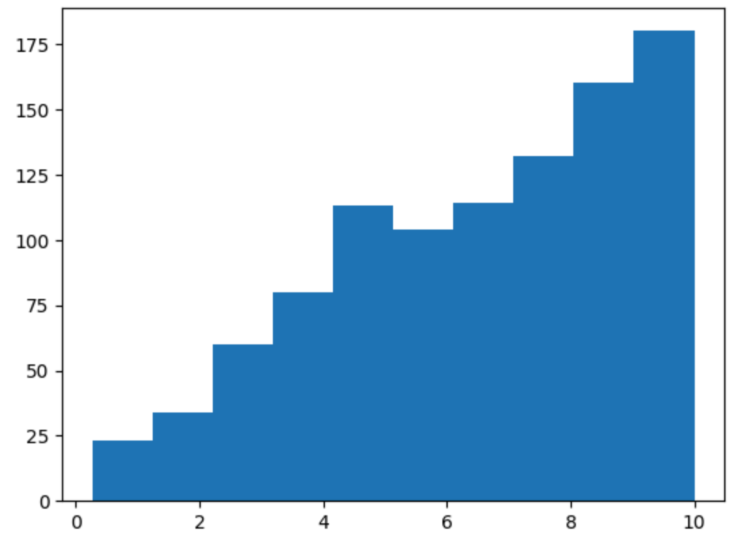
We can plot the histogram of our samples. If the sample functions are implemented correctly, the histogram should looks like our distribution.

```
def plot_histogram(data):
  plt.hist(data)
```

```
plot_histogram(sample_uniform(1000, 10, 20))
```



```
plot_histogram(sample_increasing(1000))
```



Problem 0

Try playing with the sample size and see how the histogram change with each run. Check if the result match what you think. Explain in detail.

Problem 0 Explaination: The result is an increasing function

Maximum Likelihood Estimation

Problem 1

Machines in Hamtaro's factory have their lifetime modelled by exponential distribution with an unknown parameter. Hamtaro found out that his machines failed after $x_1, x_2, \ldots, x_n$ years. Estimate the unknown parameter.

```
machine_failed_time = [2, 3, 1, 3, 4]   # In class example
# machine_failed_time = sample_exponential(5, 0.3)  # Sampled from exponential distribution

def prob1_mle(X):
  return len(X)/np.sum(X)

print('The estimated parameter is: {}'.format(prob1_mle(machine_failed_time)))
```

```
The estimated parameter is: 0.38461538461538464
```

Problem 2

Cappy is learning to perfectly replicate a new hat design. The number of attempts he needs follows a Geometric distribution with unknown parameter. For n different designs, he failed $x_1, x_2, \ldots, x_n$ times before succeed. Find the MLE of the parameter.

```
X = [0, 0, 2] # In Class Example
# X = sample_geometric(10, 0.9)   # Sampled from actual geometric distribution
```

```
def prob2_mle(X):
  # [YOUR CODE HERE]
  return 1 / np.mean(X)

print('The MLE is {}'.format(prob2_mle(X)))
```

```
The MLE is 1.5
```

Problem 3

Suppose our data $x_1, x_2, \ldots, x_n$ is randomly drawn from uniform distribution $U(a, b)$. Find MLE of a and b.

```
X = sample_uniform(100, 60, 78)

def prob3_mle(X):
  # [YOUR CODE HERE]
  a = np.min(X)
  b = np.max(X) - a
  return a, b

a, b = prob3_mle(X)
print('The MLE is ({}, {})'.format(a, b))
```

```
The MLE is (60.5026700941325, 76.73107356695073)
```

Problem 4 (Assignment 2)

Dexter tracks the growth of his prized sunflower over three days (day 0, 1, and 2). He believes the sunflower's height at the end of each day y_{t+1} is its height from the previous day y_t multiplied by a secret growth factor, α, plus some random daily noise. For a two-day period, we can observe the following Markov process: $P(y_2, y_1, y_0|\alpha) = P(y_2|y_1)P(y_1|y_0)P(y_0|\alpha)$ where $y_2 \sim \mathcal{N}(\alpha y_1, \sigma^2), y_1 \sim \mathcal{N}(\alpha y_0, \sigma^2), y_0 \sim \mathcal{N}(0, \lambda)$

Find the Maximum Likelihood Estimate (MLE) for the secret growth factor, α, given the observed heights at the end of each day y_2, y_1, y_0. In other words, compute for the value of α that maximizes $P(y_2, y_1, y_0|\alpha)$.

Solution

$$P(y_2, y_1, y_0|\alpha) = P(y_2|y_1)P(y_1|y_0)P(y_0|\alpha) = \frac{1}{2\pi\sigma}e^{-\frac{(y_2-\alpha y_1)^2}{2\sigma^2}} \times \frac{1}{2\pi\sigma}e^{-\frac{(y_1-\alpha y_0)^2}{2\sigma^2}} \times \frac{1}{2\pi\sqrt{\lambda}}e^{-\frac{y_0^2}{2\lambda}}$$

Using **log likelihood**:

$$\ln(P(y_2, y_1, y_0|\alpha)) = \left[\ln(\tfrac{1}{2\pi\sigma}) + \left(-\tfrac{(y_2-\alpha y_1)^2}{2\sigma^2}\right)\right] + \left[\ln(\tfrac{1}{2\pi\sigma}) + \left(-\tfrac{(y_1-\alpha y_0)^2}{2\sigma^2}\right)\right] + \left[\ln(\tfrac{1}{2\pi\sqrt{\lambda}}) + \left(-\tfrac{y_0^2}{2\lambda}\right)\right]$$

$$\ln(P(y_2, y_1, y_0|\alpha)) = \left[\ln(\tfrac{1}{2\pi\sigma}) + \ln(\tfrac{1}{2\pi\sigma}) + \ln(\tfrac{1}{2\pi\sqrt{\lambda}})\right] - \tfrac{(y_2-\alpha y_1)^2+(y_1-\alpha y_0)^2}{2\sigma^2} - \tfrac{y_0^2}{2\lambda}$$

.

Thus, $\frac{\mathrm{d}}{\mathrm{d}\alpha}\ln(P(y_2, y_1, y_0|\alpha)) = -\frac{-2y_1(y_2-\alpha y_1)-2y_0(y_1-\alpha y_0)}{2\sigma^2} = 0$

$$y_1(y_2 - y_1\hat{\alpha}_{MLE}) = -y_0(y_1 - y_0\hat{\alpha}_{MLE}) \implies y_1y_2 - y_1^2\hat{\alpha}_{MLE} = -y_0y_1 + y_0^2\hat{\alpha}_{MLE} \implies \hat{\alpha}_{MLE}(y_1^2 + y_0^2) = y_0y_1 + y_1y_2$$

Therefore: $\boxed{\implies \hat{\alpha}_{MLE} = \dfrac{y_0y_1 + y_1y_2}{y_1^2 + y_0^2}}$

Maximum A Posteriori Estimation

Problem 5

Hamtaro is trying to find acorns hidden by Boss. Boss has three favourite types of hiding sports:

- Type A – P[Acorn] = c_a
- Type B – P[Acorn] = c_b
- Type C – P[Acorn] = c_c

Boss has a habit to use Type A p_a of the time, Type B p_b of the time, and Type C p_c of the time. Find the MAP estimate for finding acorns at a new spot.

```
num_spot = 3
spot_acorn_prob = [0.8, 0.5, 0.4]   # From slide
spot_select_prob = [0.4, 0.4, 0.2]  # From slide
n = 5
h = 2
```

```
def spot_posterior(n, h, head_prob, select_prob):
  # [YOUR CODE HERE]
```

```
    return math.comb(n, h) * ((head_prob) ** h) * ((1 - head_prob) ** (n - h)) * select_prob

p_map = 0
p_map_val = 0
for i in range(num_spot):
  posterior = spot_posterior(n, h, spot_acorn_prob[i], spot_select_prob[i])
  print('Spot {} has posterior of {}'.format(i, posterior))
  if posterior > p_map_val:
    p_map_val = posterior
    p_map = spot_acorn_prob[i]

print()
print('The estimated parameter is {}'.format(p_map))
```

```
Spot 0 has posterior of 0.020479999999999999
Spot 1 has posterior of 0.125
Spot 2 has posterior of 0.06912

The estimated parameter is 0.5
```

Problem 6

From https://xkcd.com/1132/. Assume that chance of the sun actually explode is 10^{-6}. What are the chance that the machine said the sun exploded when it actually isn't?

```
def calculate_posterior_prob_of_lie(sun_prior, lie_prob):
  """
  Calculates the posterior probability that the machine is lying (the event
  did not happen) given that the machine reported 'YES'.
  """
  truth_prob = 1 - lie_prob

  # P(A) - Prior probability of the event happening
  p_A = sun_prior
  # P(¬A) - Prior probability of the event NOT happening
  p_not_A = 1 - p_A

  # P(B|A) - Likelihood of machine saying 'YES' if event happened (it tells the truth)
  p_B_given_A = 1 - lie_prob
  # P(B|¬A) - Likelihood of machine saying 'YES' if event did not happen (it lies)
  p_B_given_not_A = lie_prob

  # P(B) - Total probability of the machine saying 'YES' (the evidence)
  # This is the sum of true positives and false positives.
  p_B = p_B_given_A * p_A + p_B_given_not_A * p_not_A

  # P(¬A|B) - The posterior probability we want to find.
  # This is the probability of a false positive, given a positive result.
  p_not_A_given_B = p_B_given_not_A * p_not_A / p_B
  return p_not_A_given_B

# Parameters from the problem
sun_prior = 1e-6
lie_prob = 1/36

# Calculate and print the result
chance_of_lie = calculate_posterior_prob_of_lie(sun_prior, lie_prob)
print(f"Given the machine said 'YES', the probability it's a false alarm is: {chance_of_lie:.8f}")
```

```
Given the machine said 'YES', the probability it's a false alarm is: 0.99996500
```

Problem 7

Go back to problem 2-6, and try to play with input size and parameter. Observe the change in result. Explain in detail.

Problem 2

```
X = sample_geometric(100000, 0.9)   # Sampled from actual geometric distribution
print('The MLE is {}'.format(prob2_mle(X)))
```

```
The MLE is 0.901315018612155
```

Problem 3

```
X = sample_uniform(100000, 60, 78)
a, b = prob3_mle(X)
print('The MLE is ({}, {})'.format(a, b))
```

```
The MLE is (60.001910943078066, 77.99684410852788)
```