Assignment-3

January 5, 2023

1 Question 1

f is a p.f. for a discrete distribution s.t.

$$f(x) = 0 \qquad x \notin [0, 1]$$

Let f be non-zero on exactly N points $x_1, x_2 ... x_N$. And we have:

$$0 \leq x_i \leq 1$$

$$f(x_i) > 0$$

$$\Sigma_i f(x_i) = 1$$

Hence we have the mean as:

$$\mu = \sum_{i=1}^{N} x_i * f(x_i)$$

It is easy to see that $x_i \in [0, 1] \rightarrow$ \$ [0, 1]\$

Now, let's calculated the variance:

$$\sigma^2 = E(X^2) - E(X)^2 = \sum_{i=1}^{N} x_i^2 * f(x_i) - \mu^2$$

Since all $x_i \in [0, 1] \rightarrow x_i^2 \leq x_i$

$$\sigma^2 \leq \sum_{i=1}^{N} x_i * f(x_i) - \mu^2 \leq \mu - \mu^2$$

Since $\mu \in [0, 1]$ and $\mu - \mu^2$ takes it maximum value at $1 - 2 * \mu' = 0$, we have $\mu' = 1/2$

$$\sigma^2 \leq \mu' - \mu'^2$$

$$\sigma^2 \leq 1/4$$

2 Question 2

For each of the three girls, the average number of hits can be represented as a normal distribution with mean p and var $p(1-p)$, where p is the probability that target is hit.

For n attemts, the distribution of total hits is $\sim N(np, np(1-p))$

Adding these normal distributions for all 3 girls, we have:

$$mean = 10 * 0.3 + 15 * 0.2 + 20 * 0.1 = 8$$

$$variance = 10 * 0.3 * 0.7 + 15 * 0.2 * 0.8 + 20 * 0.1 * 0.9 = 6.3$$

So, the total number of hits is a normal distribution $X \sim N(8, 6.3)$

$$Pr(X \geq 12) = 1 - Pr(X \leq 12) = 0.055508$$

[1]:
```
from scipy.stats import norm
import math
1- norm.cdf(12, loc=8, scale=math.sqrt(6.3))
```

[1]: 0.05550855275609312

3 Question 3

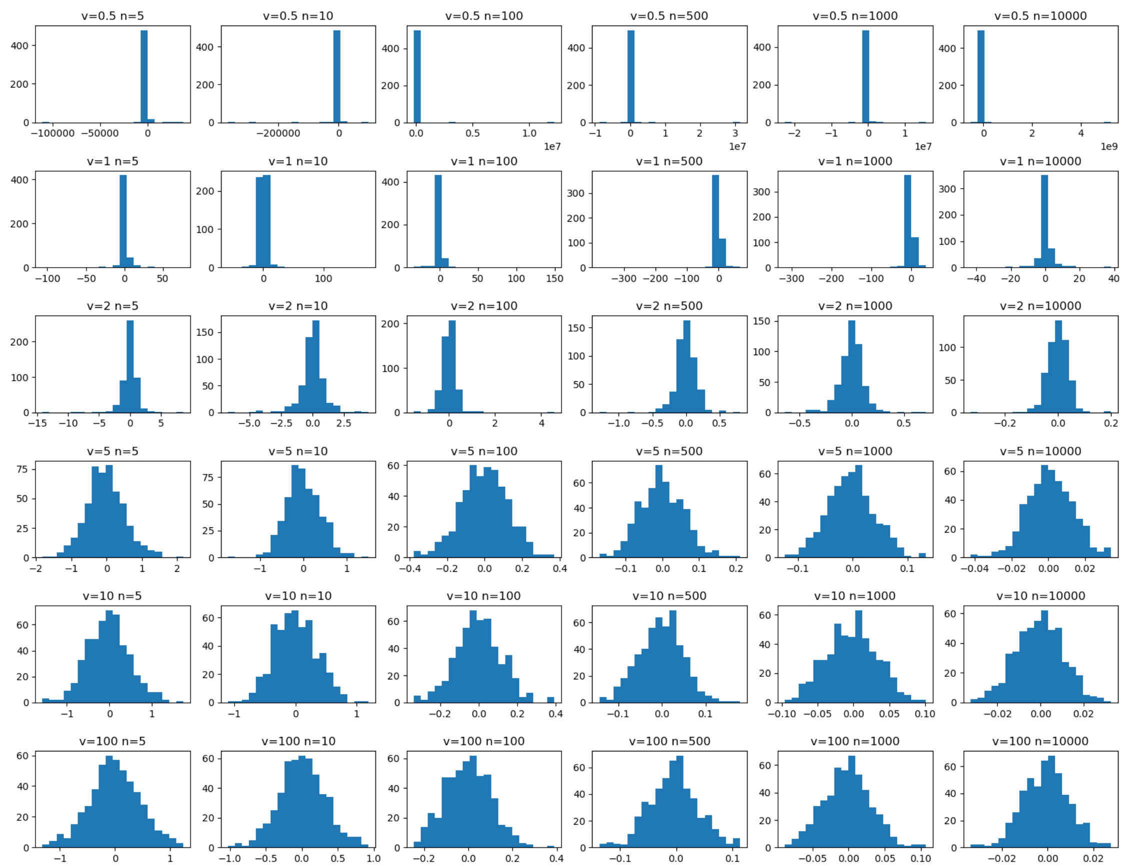
[2]:
```
import numpy as np
from tabulate import tabulate
import time
M = 500
N = 10_000

v_vals = [0.5, 1, 2, 5, 10, 100]
n_vals = [5, 10, 100, 500, 1_000, 10_000]
sample_means = {}
stats = [['v', 'n', 'mean', 'variance', 'std']]

np.random.seed(int(time.time()))
for v in v_vals:
    samples_mn = np.random.standard_t(v, (M, N))
    for n in n_vals:
        # Part (a)
        selected_samples_Mn = samples_mn[:, :n]
        # Part (b)
        sample_means_M = np.mean(selected_samples_Mn, axis=1)
        sample_means[(v,n)] = sample_means_M
        stats.append([v, n, sample_means_M.mean(), sample_means_M.var(),
 ↪sample_means_M.std()])
```

```python
# Part (c, f)
print(tabulate(stats))
```

v	n	mean	variance	std
0.5	5	-69.41945426521231	29966158.933756363	5474.1354508046625
0.5	10	-1524.5823095137766	497890885.6524275	22313.46870507648
0.5	100	31014.34011145233	318830864309.6458	564651.0996267039
0.5	500	64394.87569788464	2215577434051.354	1488481.5867357426
0.5	1000	-8966.755751254506	1607103572446.6006	1267715.8879049362
0.5	10000	7631805.294666929	5.620696388516489e+16	237080079.0559276
1	5	0.47959051407552356	78.79703686831809	8.876769506319182
1	10	0.39954466787949	97.72747487660428	9.885720756556108
1	100	0.33455077798918165	80.6799139971204	8.982199841749258
1	500	-0.5576324597902084	346.8787962590982	18.624682447201568
1	1000	-1.0428283091553334	301.4573488425346	17.362527144472217
1	10000	0.1498496328486327	26.11003764226102	5.1097981997590685
2	5	0.010626329431091224	1.9723260189901441	1.4043952502732784
2	10	0.0031221292471105074	0.9141927416766158	0.9561342696905157
2	100	0.017770173776017827	0.13338797400040717	0.36522318382108104
2	500	-0.0022463012708793446	0.025397872278276067	0.15936709910855523
2	1000	-0.0021887564439385357	0.01424556231179281	0.1193547749853051
2	10000	-0.0009654711467566979	0.0016852473593495956	0.04105176438777748
5	5	-0.026869831969260394	0.3027880702220871	0.5502618197023005
5	10	-0.004527042848113494	0.15952271422523562	0.3994029471914743
5	100	-0.0012023441751491846	0.015528995315023714	0.12461538955933057
5	500	-0.0007653484097352778	0.0035098854600005094	0.05924428630678666
5	1000	-0.0025517502213305353	0.0017742791616889141	0.04212219322030744
5	10000	0.0010203406615304856	0.00015678824641922562	0.01252151134724661
10	5	-0.026075455279535704	0.25148099229219656	0.5014788054267065
10	10	0.0015593283811619126	0.11954927703681664	0.3457589869212609
10	100	-0.0004877745573768353	0.013926289091543588	0.11800969914182304
10	500	-0.0021684976401448244	0.002570458901931934	0.050699693312010616
10	1000	-0.002252507192627102	0.0012620438874498183	0.03552525703566152
10	10000	-0.0005619197898050718	0.00012184228710861561	0.011038219381250566
100	5	-0.0049343657453706415	0.21306558948387477	0.46159028313416084
100	10	0.007434822067794055	0.10029409864158721	0.316692435403164
100	100	-0.00056862623944648	0.010603579057121646	0.10297368138083461
100	500	-0.004155049101416592	0.0019221387385407272	0.043842202710866696
100	1000	-0.0015018078982640074	0.0010509703676396165	0.03241867313200243
100	10000	-0.0007588925337191001	9.706390166563541e-05	0.00985210138323979

[3]:
```python
import matplotlib.pyplot as plt

# Part (d, f)
```

```python
fig, ax = plt.subplots(len(v_vals), len(n_vals), figsize=(20,12))
plt.subplots_adjust(top = 0.99, bottom=0.01, hspace=0.5, wspace=0.2)
for i, v in enumerate(v_vals):
    for j, n in enumerate(n_vals):
        ax[i, j].hist(sample_means[(v, n)], bins=20)
        ax[i, j].set_title(f"v={v} n={n}")
fig.show()
```



Part (e)

LLN Law of large numbers state, that the sample mean converges to the true mean for a large enough value of sample size n.

This is visible for $v \in 2, 5, 10, 100$, since the mean becomes closer and closer to 0, as n incearses.

For $v \in 0.5, 1$, for mean doesn't converge to a particular value, but keeps moving. This is in live with the mean not existing for t_v distribution for $v \leq 1$

CLT Central limit throrem states, that the sample means converges to a normal distribution. The histograms visibally form a normal distribution for $v \in 2, 5, 10, 100$ and $M = 500$ around the sample mean. The variance also reduced as n increases, as expected.

For $v \in 0.5, 1$, for mean doesn't exist, so the sample means don't converge to a normal distribution.

4 Question 4

$$Q_1 + Q_2 ... + Q_k = 1$$
$$n_1 + n_2 ... + n_k = n$$

Let X be the sample population, which satisfies the given constraints.

The likelihood function can be written as:

$$f_n(X|Q_1, Q_2, ...Q_k) = f(x_1|Q_1, Q_2, ...Q_k) * f(x_2|Q_1, Q_2, ...Q_k)..f(x_n|Q_1, Q_2, ...Q_k)$$
$$f_n(X|Q_1, Q_2, ...Q_k) = Q_1^{n_1} * Q_k^{n_k} ... Q_k^{n_k}$$

Log likelihood is:

$$L(Q_1, Q_2, ...Q_k) = \sum_{i=0}^{k} n_i * log(Q_i)$$

$$L(Q_1, Q_2, ...Q_k) = \sum_{i=0}^{k-1} n_i * log(Q_i) - n_k * log(1 - Q_1, Q_2, ...Q_{k-1})$$

Calculating the partial derivative:

$$\frac{\partial L(Q_1, Q_2, ...Q_k)}{\partial Q_i} = n_i/Q_i - n_k/Q_k$$

Setting all partial derivates to zero, we have:

$$n_i/Q_i = a \qquad \forall i \in 1...k$$

$$n_i = a * Q_i$$
$$\Sigma n_i = \Sigma a * Q_i = n$$
$$a * \Sigma Q_i = n$$
$$a = n$$

Solving this equation for a, and using the initial conditions gives $a = n$

$$Q_i = n_i/n$$

5 Question 5

For A

$$f(x|\alpha,\beta) = \frac{x^{\alpha-1}e^{-\beta x}\beta^{\alpha}}{\Gamma(\alpha)}$$
$$f(x=2|\alpha=3,\beta=\theta) = \frac{2^2 e^{-2\theta}\theta^3}{\Gamma(2)}$$
$$L(\theta) = 4e^{-2\theta}\theta^3$$

For B

$$f(y|\lambda) = \frac{\lambda^y e^{-\lambda}}{y!}$$
$$f(y=3|\lambda=2\theta) = \frac{(2\theta)^3 e^{-2\theta}}{3!}$$
$$L(\theta) = (4/3)e^{-2\theta}\theta^3$$

As we can see, both the likelihood functions are the same, except for the constant multiplier.

Solving one of these for θ:

$$L(\theta) = e^{-2\theta}\theta^3$$
$$\frac{dL(\theta)}{d\theta} = -2e^{-2\theta}\theta^3 + 3e^{-2\theta}\theta^2$$
$$\frac{dL(\theta)}{d\theta} = e^{-2\theta}\theta^2(-2\theta+3)$$

This gives $\theta = 0$ or $\theta = 3/2$ as the possible values, and $\theta = 3/2$ gives the maxiumum likelihood.

6 Question 6

Part (a)

Mean of the defined uniform distribution is $\theta/2$. Using MM, we have:

$$\bar{X} = \hat{\theta}_{MM}/2$$
$$\hat{\theta}_{MM} = 2\bar{X}$$

Part (b)

(1)

$$E(\hat{\theta}_{MM}) = 2E(\bar{X}) = \theta_o$$

(2)

$$Var(\hat{\theta}_{MM}) = 4Var(\bar{X})$$

We know that the $Var(X) = \theta^2/12 \rightarrow Var(\bar{X}) = \theta^2/12n$

$$Var(\hat{\theta}_{MM}) = \theta^2/3n$$

For MLE, we will first derive the CDF and PDF:

$$\hat{\theta}_{MLE} = max(x_1, x_2...x_n)$$

For a general $x \in [0, \theta_o]$

$$Pr(\hat{\theta}_{MLE} \leq x) = Pr(x_1 \leq x, x_2 \leq x...x_n \leq x)$$

Since all x_i are iid:

$$Pr(\hat{\theta}_{MLE} \leq x) = Pr(x_i \leq x)^n$$

We have:

$$Pr(x_i \leq x) = \int_0^x 1/\theta \ dx_i = x/\theta$$

So:

$$Pr(\hat{\theta}_{MLE} \leq x) = Pr(x_i \leq x)^n$$
$$Pr(\hat{\theta}_{MLE} \leq x) = \frac{x^n}{\theta^n}$$

Using this CDF, we can easily derive the PDF:

$$Pr_{\hat{\theta}_{MLE}}(x) = \frac{nx^{n-1}}{\theta^n} \qquad x \in [0, Q_o]$$

Using this PDF, we can now calculate the mean and variance of $\hat{\theta}_{MLE}$

Mean:

$$E(\hat{\theta}_{MLE}) = \int_0^{Q_o} x * \frac{nx^{n-1}}{\theta^n} dx$$
$$E(\hat{\theta}_{MLE}) = \frac{nQ_o}{n+1}$$

Variance:

$$E(\hat{\theta}^2_{MLE}) = \int_0^{\theta_o} x^2 * \frac{nx^{n-1}}{\theta^n_o} dx$$
$$E(\hat{\theta}^2_{MLE}) = \frac{n\theta^2_o}{n+2}$$

$$Var(\hat{\theta}^2_{MLE}) = \frac{nQ^2_o}{n+2} - (\frac{nQ_o}{n+1})^2$$
$$Var(\hat{\theta}^2_{MLE}) = \frac{nQ^2_o}{(n+2)(n+1)^2}$$

6.0.1 Bias and Consistent

MM is unbias and consistent, this is because $E(\hat{\theta}_{MM}) = \theta_o$ and as $n \to \text{inf}$, this gives θ_o.

MLE is bias and consistent, this is because $E(\hat{\theta}_{MM}) \neq \theta_o$ but as $n \to \text{inf}$, this gives θ_o.

6.0.2 Asymptotic distribution

MM is normally distributed as $n \to \text{inf}$, this is easy to say because it is a mean estimator and they are normally distributed.

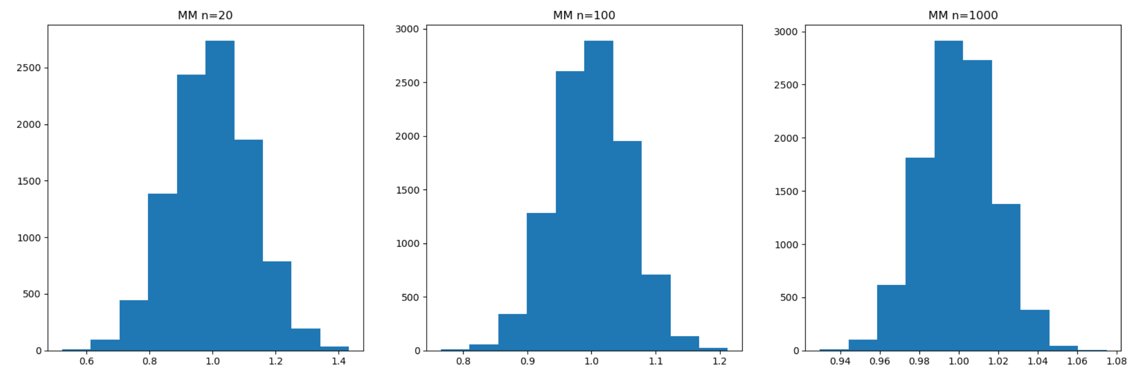
MLE is not normally distributed $n \to \text{inf}$, it is rather a point mass at θ.

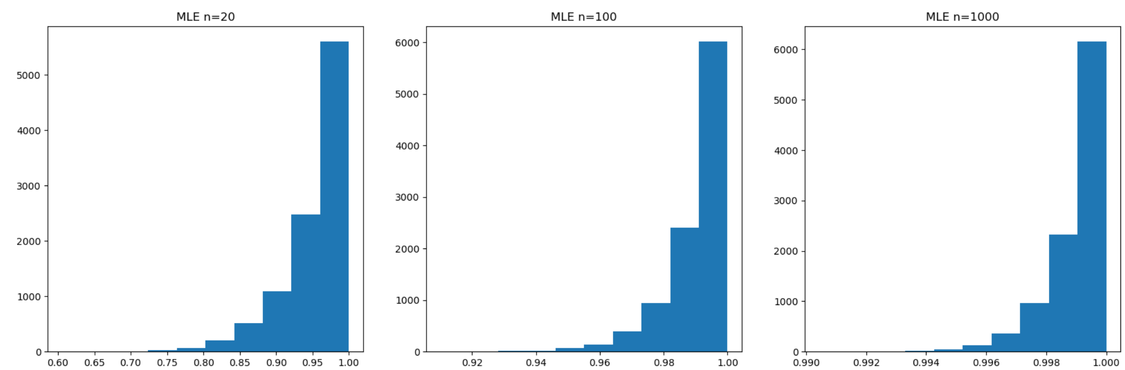
This is visible in the cdf $\frac{x^n}{\theta^n}$, which is 1 at $x = \theta$, but goes to 0 for all $x \in [0, \theta)$.

[9]:
```
# Part (c)
n_vals = [20, 100, 1000]
m = 10_000
Q = 1

fig, ax = plt.subplots(2, len(n_vals), figsize=(20,12))
fig.subplots_adjust(top = 0.99, bottom=0.01, hspace=0.5, wspace=0.2)
stats = [['estimator', 'n', 'mean', 'bias', 'std']]

for i, n in enumerate(n_vals):
    samples_mn = np.random.uniform(low=0, high=Q, size=(m, n))
    mm_estimation_m = samples_mn.mean(axis=1) * 2
    mle_estimation_m = samples_mn.max(axis=1)
    ax[0, i].hist(mm_estimation_m)
    ax[0, i].set_title(f"MM n={n}")
    ax[1, i].hist(mle_estimation_m)
    ax[1, i].set_title(f"MLE n={n}")

    stats.append(['MM', n, mm_estimation_m.mean(), abs(Q - mm_estimation_m.
 ↪mean()), mm_estimation_m.std()])
    stats.append(['MLE', n, mle_estimation_m.mean(), abs(Q - mle_estimation_m.
 ↪mean()), mle_estimation_m.std()])

fig.show()
print(tabulate(stats, floatfmt=".6f"))
```

```
---------  ----  ------------------  ---------------------
---------------------
estimator  n     mean                bias                   std
MM         20    0.9980677676252028  0.001932232374797227   0.126840844556426
MLE        20    0.9526211650604582  0.04737883493954176    0.04514806570024145
MM         100   0.9994555519688681  0.00054444803113185    0.05795010647510843
MLE        100   0.9902689100009692  0.009731089999030829   0.009622016854230258
```

```
MM         1000  1.00003507445371    3.507445371009332e-05  0.01833104720016529
MLE        1000  0.998998054697398   0.001001945302602003
0.0009941839560054456
---------  ----  -----------------  ---------------------
---------------------
```





Part (d)

For MLE, the mean becomes closer and closer to one, as n increases and the variance does down. This is in line with theoretical results. For MM, the mean becomes closer and closer to one, as n increases and the variance does down. This is in line with theoretical results.

MLE has greater bias than MM, which is not in line with the theoretical expectations.

Part (e)

The distribution of $\widehat{\theta}_{MLE}$ is not asympotically normal, which doesn't align with the general result for MLE estimators.
MLE has greater buas than MM, which doesn't align with the general result which says MLE estimators are the most optimal and efficient estimators.

[ ]: