RL Lab 1

Harsh Raj (180010017)

Akhilesh Bharadwaj (180010009)

Note: Code for all questions (including simulations etc) is attached in the submitted zip. Artifacts including the images generated are also attached in the zip. Please make sure to install dependencies mentioned in the requirements.txt before running any submitted code.

```
# Install dependencies
pip3 install -r requirements.txt

# run codes corresponding to each question
python3 runner.py

# change hyperparameters:
# either change manually from conf/configs.yaml for each ques
# or change dynamically from command line
python3 runner.py seed=5
# config management is done via hydra. Read more
https://github.com/facebookresearch/hydra
```

Ques 1.

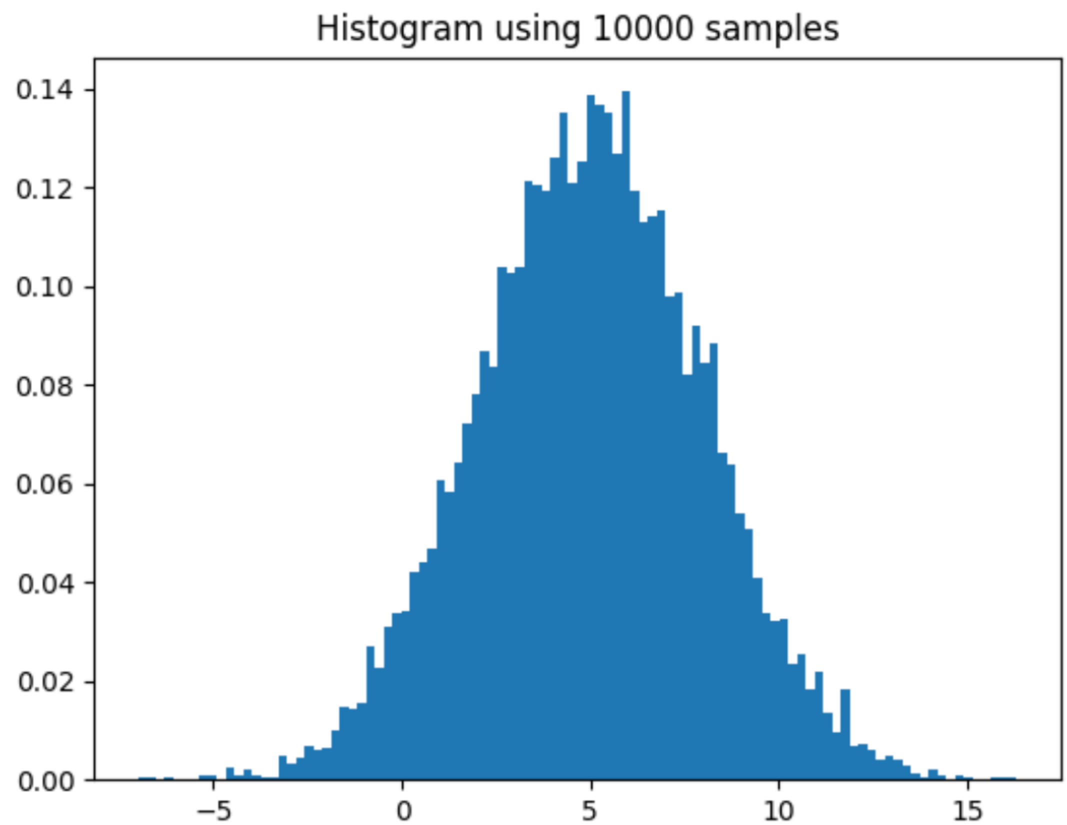
Plots attached in the zip.

Ques 2.

Acknowledgement: [Stackoverflow](), [statisticalengineering.com]()

Central Limit Theorem states that "the distribution of sample means approximates a normal distribution as the sample size gets larger". In order to generate a Normal distribution using a uniform random variable, we collect fixed sized samples from the population (let us say size =

10). Each sample mean would then represent a sample from the Normal distribution with mean 0.5 and some standard deviation. We convert it to standard Normal by subtracting the mean and dividing by std of the data points. In order to convert this Uniform Normal Distribution into the expected normal distribution, we add μ to each sample mean and multiply with σ. The final numbers belong to the required normal distribution. We verify the same by plotting the curve.



Here we considered μ=5, and σ=3

Update: The method of generating Normal Samples is improved using [Box Muller Transforms]
It uses only two uniform samples to generate a couple of samples for Normal distribution.

Ques 3.

~~We know that the area under the curve is represented by its integral with the variable under the given domain. Also, this integral can be approximated with the sum of rectangles. Let us say, we need to find the area under the curve given by function f with domain as (0, pi)~~

~~We divide the domain into n equal intervals (each being equal to pi / n). The area would then be approximated by~~

$$\cancel{\Sigma_{i=1 \text{ to } n} f(i/n)\ pi / n}$$

~~Consider a random variable X which can take values from {f(1/n)*pi, f(2/n)*pi, …., f(1)*pi} each with equal probability (ie. the distribution of the random variable is uniform). The above sum is the same as the expected value of the given random variable.~~
~~We find the expected value of the random variable by simulation, by sampling the random variable from the given uniform distribution. We keep the value of n to be quite large, and number of samples to be large in order to make approximation closer to the real value.~~

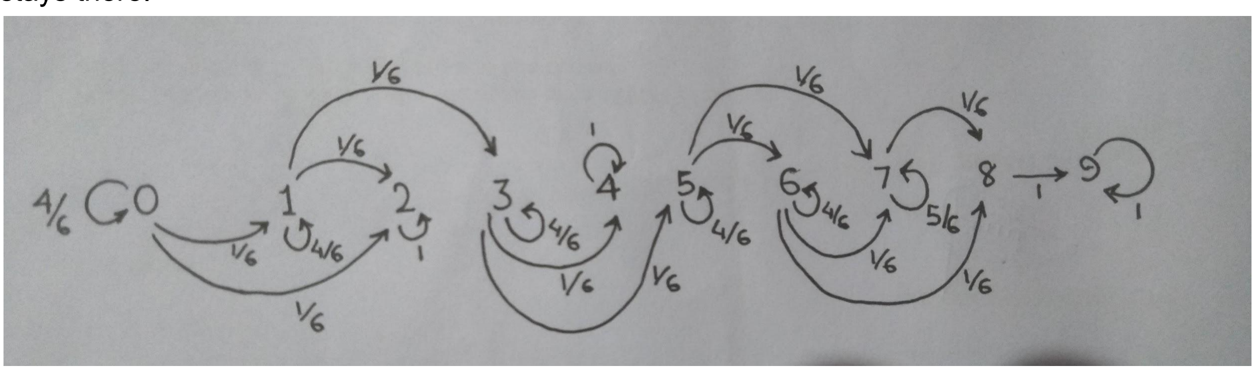
For calculating the area via sampling, we take many samples in the range [0, pi], and take the average of the values of the function at those samples. For question 1, we use uniform random while for question 2, we modify the normal distribution with mean = 0, and sigma = 1/ sqrt(2). We finally take the absolute value of the generated random variable, and clip it between 0 and pi. This is then scaled appropriately before finding the value of the function at the given point. Finally, the calculated area is:

The required area in [0, pi] for:

a) $\sqrt{\sin(x)}$ is 2.39627
b) $\sqrt{\sin(x)} \exp(-x^2)$ is 0.57484

Ques 4.

Assumptions: From state 7, we can move to state 8 only when the dice roll out to 1. Else it stays there.



Depicted above is the required Markov Chain. Here the random variable can take values in {0, 1,..., 9} at each timestep with certain probability. The state transition probabilities are marked over the edges. States 2 and 4 are dead states. We introduce a dummy node 9, with transition probabilities defined as shown in the figure.

$$\text{Req. Prob} = \Sigma_{i=0 \text{ to inf}} \text{ Prob of ending at state 8 in i'th timestamp}$$
$$= \Sigma_{i=0 \text{ to inf}} (\mu_o P^i)[8]$$

The initial probability μ_o equals a vector of size 10, with 0th element as 1 and rest all 0.

We run this simulation for inf=10**7, and find that the Required probability equals 0.125

We cross verify our result, by simulating an RL game, where an agent chooses action (from 1 to 6) by rolling a dice. The environment starts with the agent being at state 0 initially, and finishes if the agent reaches state 8 or if the agent makes more than a predefined MAX_TIMESTEPS. The environment returns a reward of 1 if the game ends with the agent being in state 8 else 0. We calculate the average rewards across 10**7 games with MAX_TIMESTEPS as 100, and find that the average reward (which is expected to represent the probability of agent ending at state 8 using the frequency count approach), equals 0.125