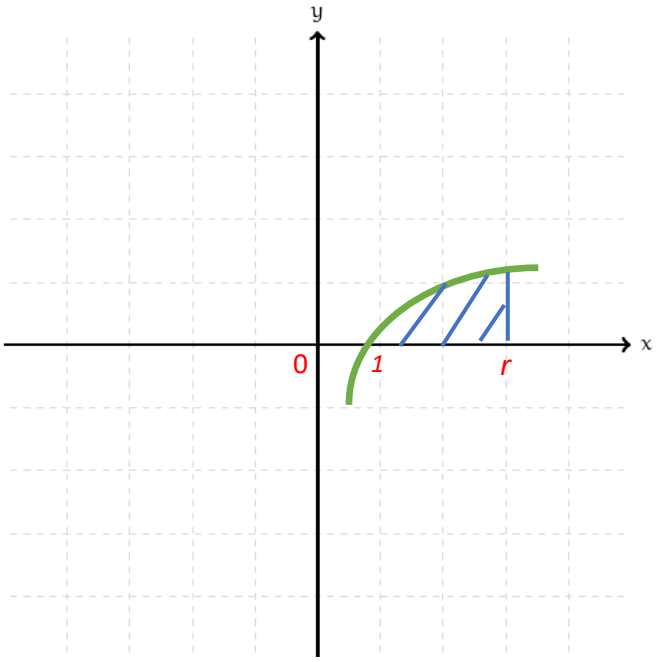
Assignment 1-CIS694/EEC693/CIS593 Deep Learning-2024 Spring

Due: 11:30pm of Feb 19th

Two questions: 100 points in total. Please zip all your Python files into one single zipped file, and then submit the single zipped file on Blackboard. Note: The Python version is Python 3.

Q1 [50 points]. Let us design a **Monte Carlo** method to estimate the area of the shadow region under the green function $y = log_2(x)$ when *x* is from 0 to *r*. As shown in the following figure, *r* is a variable (*r* > 1) for range.



Please write a Python program named "**MonteCarlo.py**" to implement your designed Monte Carlo method. In the first line of your code, please write "*n*=10000" to define *n* as the experiment number of random sampling. Finally, please print out the estimated area for 3 different range variables with the following format:

When r=5, the estimated shadow area is xxx

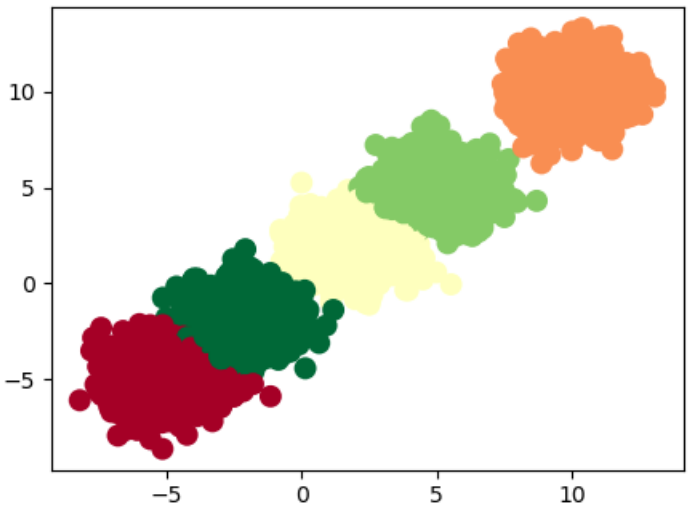
When r=10, the estimated shadow area is xxx

When r=15, the estimated shadow area is xxx

[Notes] TA might change your *r* to test the computing result. Please use **Monte Carlo** based methods *similar to that during our class teaching* for this question, and other methods (like Integral, Geometry, etc) will receive 0 points for this question.

Q2 [50 points]. In this question, you will try to implement the Kmeans clustering algorithm based on the algorithm (pseudo code) described in the class.

First, please read the provided "**myKmeans.py**" file and try to understand the code. If you run it, it will call the Sklearn built-in function "KMeans" and draw the following clustering figure:



Then, you need to finish two tasks for Q2:

Task 1: Implement your own kmeans clustering function myKmeans() (40 points)

```
# Write your code here for your own kmeans clustering
function
def myKmeans(x, K, max_iteration=1000):
    # write your code here based on the algorithm
described in class, and you cannot call other kmeans
clustering packages here
    return center, mean_intra_cluster_distance,
mean_inter_cluster_distance
```

Please do not modify the input arguments and returned output. "x" is already defined as the fake data created before, "K" is the cluster number, "center" is the center vector of each cluster.

Task 2: Find optimal K for your own kmeans clustering (10 points)

```
# Write your code for a loop to call your own function
myKmeans() by setting cluster_number=K from 2 to 10
# print the ratio of mean_intra_cluster_distance over
mean_inter_cluster_distance for each K.
# print the optimal K with minimum ratio
```

For example, running your code might generate the following prints in the command window:

When K=2, mean intra-cluster distance: 1, mean inter-cluster distance: 2, ratio: 0.5

When K=3, mean intra-cluster distance: 2, mean inter-cluster distance: 2, ratio: 1.0

…

When K=10, mean intra-cluster distance: 2, mean inter-cluster distance: 5, ratio: 0.4

The optimal K is xxx, because K=xxx obtains the minimum ratio xxx.