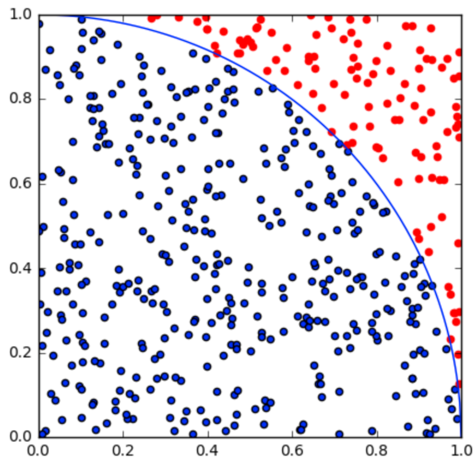
HOMEWORK 7

Due Thursday, March 9, at 11pm

Please enter your answers into a Jupyter notebook and submit by the deadline via canvas.

Monte-Carlo Estimation of π. Estimate π by using the following method. Generate $N = 5000$ (or more) pairs of numbers that are each uniformly random in the interval $(0, 1)$. Count the number of pairs that are within the circle defined by $x^2 + y^2 = 1$.



Estimate π this way and make the above plot using the code below:

```
plt.figure(figsize=(5,5))
plt.scatter(X_out, Y_out, color="red")
plt.scatter(X_in,Y_in)
plt.plot(X, Y)
plt.axis([0, 1, 0, 1])
```

Random walks in two and three dimensions. In the last homework, we had the drunk bear Randi do a random walk in one dimension. We will now to the same in 2 and 3 dimensions.

- Imagine Randi is on a grid, at position $(0, 0)$. At each time step, Randi goes, with equal probability, up, down, left or right (up means y increases by 1 and right means x increases by 1). Simulate 1000 random walks of length $m = 1000, 2000, \ldots, 10000$ in this 2-dimensional grid, and count the number of 2-dimensional random walks of

Randi that end up with him going back to the origin at some point; and plot this as a function of m. Don't fill a giant numpy array with every simulation like we did in the last lecture (uses too much memory; instead, generate random walks one by one and store only the information you need).

- Do the same in a three-dimensional grid (6 different directions for Randi to go at each step).

As we discussed in class, this problem is about Polya's theorem. Which says that if you take a random walk in a 1 or 2-dimensional grid, then you will *eventually* return to the starting position with probability 1 (i.e. the probability goes to 1 as m goes to infinity). On the other hand, this is not true in 3 dimensions and above. For the 2D-random-walk plot you made, notice (mentally) how slowly the value goes to 1, whereas for the 3d version, it's not even close.

Simulation of the Law of Large numbers and the Central Limit Theorem. We are drawing n uniform random numbers. The average of these numbers will also be random but will have a different distribution. We want to see how the average changes (as a distribution) as n increases.

More precisely, let $x_1, x_2, x_2, \ldots$ be drawn from the uniformly distribution between 0 and 1 (i.e. `random.random()`). Let

$$y_n = \frac{x_1 + \cdots + x_n}{n}$$

be the average.

- Write a function `get_x()` that will return a uniformly random number between 0 and 1.
- Write a function `get_y(n)` that will return the average of n random numbers picked by `get_x()`.
- (You can copy this from Lecture 20) Write a function `make_hist(YY)` that will take a numpy array `YY`, make a histogram (use `normed=True` in `plt.hist(...)`) and plot the best fitting Gaussian curve to the content of YY.
- For each of $n = 1, 10, 100, 1000, 10000, 100000$, sample 1000 values of y_n (using `get_y(n)`) and draw a histogram and the best fitting distribution. Print out the standard deviation of the data when $n = 1$ (i.e. average of one single x; it should be 0.28).
- For $n = 1, 2, \ldots, 100$, draw 1000 values from `get_y()` and compute the standard deviation of the data (don't draw the histogram). Then plot the standard deviation value versus n. Also plot, in the same graph, the function $0.28/\sqrt{n}$.
- Write briefly, why the results you got are consistent with the law of large numbers and the central limit theorem.