SDS | Page No. | Date

Bayesian Computing

Name : Breksha Ashok Patel
Sapid : 60004210126
Branch : Computer Engineering
Div : C2

Experiment no. 3

Aim :- To apply Markov chain Monte Carlo (MCMC) method to estimate the area of unit circle. [Hit - and - miss] [example : to calculate the area of a circle] [The area of a circle is computed as πR^2]

Theory :-

Markov chain Monte Carlo (MCMC) Method :-

Markov chain Monte Carlo (MCMC) is a computational method for approximating probability distributions by generating a sequence of random samples from the distribution. MCMC methods are widely used in Bayesian computing to sample from posterior distributions, which are typically high-dimensional and difficult to sample from directly. The basic idea behind MCMC is to construct a markov chain that has the desired probability distribution or as its stationary distribution. A markov chain is a sequence of random random states where the probability of transitioning to the next state depends only on the current state. The key is to design the transition probabilities so that the markov chain will eventually converge to the desired distribution. There are several different MCMC algorithms, but they all show the same general structure. Each algorithm starts with an initial state, which may or may not be from the desired distribution. The algorithm then generates a sequence of new states by randomly selecting a new state

from a candidate distribution. The candidate distribution is typically ~~dist~~ designed to be close ~~to~~ the desired distribution but it may not be exactly the same. The new state is accepted with a certain probability, which is determined by the ratio of the desired distribution of the new state to the desired distribution of the current state; otherwise, current state remains the same. MCMC is run for large no. of iterations. The no. of iterations depends on complexity of the distribution and efficiency of MCMC algorithm. MCMC methods are powerful tool for Bayesian computing and they have been used to solve problems in statistics, science and engineering.

Conclusion :-

In this experiment, we learned how to use the MCMC method to estimate ~~a~~ area of unit circle & thus approximate value of π. We used 2 different codes to approximate random points inside a square and count how many of them are also inside the circle. We then calculated ratio of circle area to square area and multiplied it by square area to get an approximation of π. We plotted the points and their colours to show ~~whether~~ whether they are inside or outside the circle. MCMC has many applications in different fields, such as physics, biology, statistics & machine learning.

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)
Department of Computer Engineering

SAP ID.: 60004210126 Name: Preksha Ashok Patel

Batch: C2-1

Subject: Bayesian Computing Laboratory Semester: VII

Experiment No. 3

Aim:

To apply Markov Chain Monte Carlo (MCMC) method to estimate the area of Unit Circle. [Hit-and-miss example: to calculate the area of a circle][The area of a circle is computed as πR^2]

Code:

Importing Libraries

```
import numpy as np import
matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('white')
sns.set_context('paper')
```

Part 1

```
radius = 1



N = 100000  #Use 10000 points

X = np.random.uniform(low=-radius, high=radius, size=N)  # Random numbers from -1 to
   1

Y = np.random.uniform(low=-radius, high=radius, size=N)



# calculate the distance from the center

R = np.sqrt(X**2+Y**2);
```

```
box_area =(2.0*radius)**2      # This is the big

box  is_point_inside = R<radius

N_inside=np.sum(is_point_inside) circle_area =

box_area*N_inside/N


plt.scatter(X,Y, c=is_point_inside, s=5.0, edgecolors='none', cmap=plt.cm.Paired)

plt.axis('equal')


print "Area of the circle = ", circle_area print
"pi = ", circle_area/radius**2
```

Part 2 (Alternative)

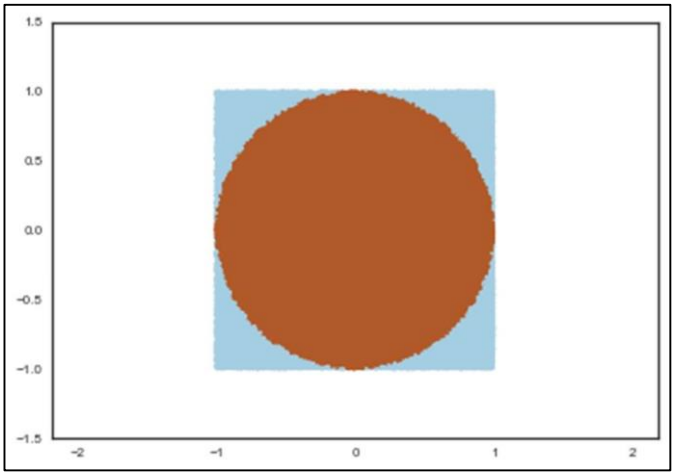
```
# Dan's think-through


N = 100000.0 # number of points

radius = 1.0 # radius of circle


# generate points in circumscribing square

X = np.random.uniform(low=-radius, high=radius, size=int(N))

Y = np.random.uniform(low=-radius, high=radius, size=int(N))


origin_dist_sqr = X*X + Y*Y is_in_circle
= origin_dist_sqr < radius ** 2 in_circle
= np.sum(is_in_circle)
```

```
square_area = (2 * radius) ** 2 circle_area = in_circle / N *

square_area pi_approx = circle_area/radius ** 2  #

manipulate A = pi*r**2


plt.scatter(X, Y, c=is_in_circle, edgecolors='none', cmap=plt.cm.Blues)

plt.axis('equal')


print 'Circle area:', circle_area print 'Pi approximation with %4s trials is
%.6s' % (str(int(N)), pi_approx)
```

Output:

Part 1

Area of the circle = 3.14436
pi = 3.14436



Part 2 (Alternative)

Circle area: 3.14884
Pi approximation with 100000 trials is 3.1488