ADVANCED STATISTICAL METHODS

Project Report

Monte Carlo Methods
And
Markov Chain Monte Carlo Methods

Group-25

Team Members:-

Yagna Karthik Vaka	S20170010170	UG-3
Hemanth Phaneedra Varma	S20170010048	UG-3
E. Pruthvik Reddy	S20170020203	UG-3

Monte Carlo Methods

Monte Carlo Sampling Methods:

- **Monte Carlo sampling(*using integration*)**:
  - The basic idea of Monte Carlo integration is very simple and only requires elementary statistics. Suppose we want to find the value of

$$\int_a^b f(x)dx$$

in some regions with volume V. Monte Carlo integration estimates this integral by estimating the fraction of random points that fall below f(x)
f(x) multiplied by V.

In a statistical context, we use Monte Carlo integration to estimate the expectation

$$E[h(X)] = \int_X h(x)f(x)dx$$

With

$$\bar{h_n} = \frac{1}{n}\sum_{i=1}^{n} h(x_i)$$

where xi~f is a draw from the density f. We can estimate the **Monte Carlo variance** of the approximation as

$$v_n = \frac{1}{n^2}\sum_{o=1}^{n}(h(x_i) - \bar{h_n})^2)$$

Example_1:

We want to estiamte the following integral $\int_0^1 e^x dx$. The minimum value of the function is 1 at $x = 0$ and e at $x = 1$.

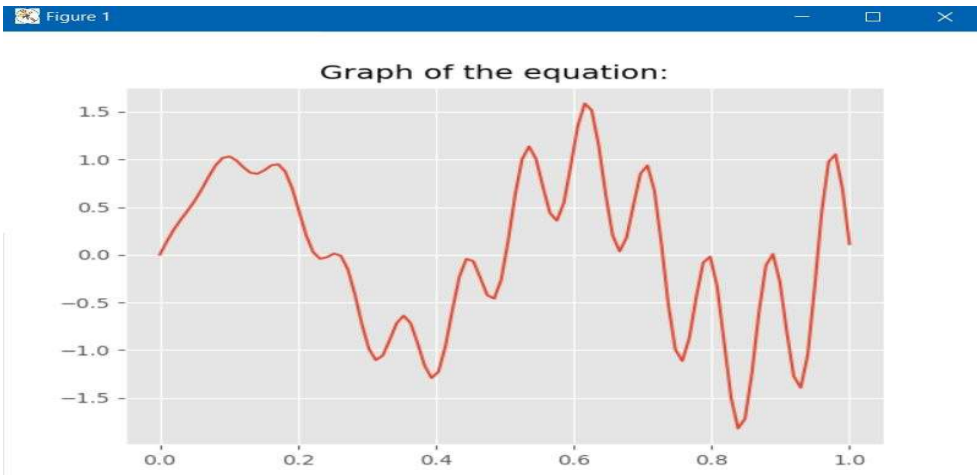
```
Integraation value is :
1.71828182845905

As the value n goes on then we converge the solution to integration value as above:
n value is :         10 and solution((volume * count) /n) is : 1.902797
n value is :        100 and solution((volume * count) /n) is : 1.549421
n value is :       1000 and solution((volume * count) /n) is : 1.701644
n value is :      10000 and solution((volume * count) /n) is : 1.726109
n value is :     100000 and solution((volume * count) /n) is : 1.717927
n value is :    1000000 and solution((volume * count) /n) is : 1.718732
n value is :   10000000 and solution((volume * count) /n) is : 1.718081
n value is :  100000000 and solution((volume * count) /n) is : 1.718367
```

Monitoring Variance:

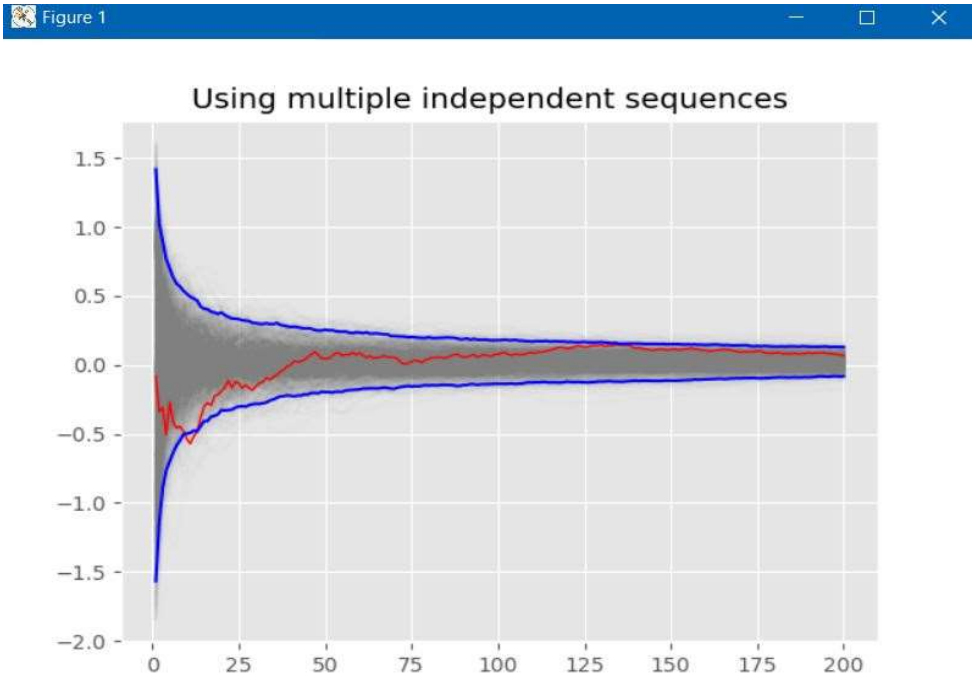
We are often interested in knowing how many iterations it takes for Monte Carlo integration to “converge”. To do this, we would like some estimate of the variance, and it is useful to inspect such plots. One simple way to get confidence intervals for the plot of Monte Carlo estimate against number of iterations is simply to do many such simulations.

For the **example_2**, we will try to estimate the function

$$f(x) = x\cos 7x + \sin 13x, \;\; 0 \leq x \leq 1$$



```
Integraation value is :
0.0202549
```



Application of Monte Carlo Method in Finance:

Monte Carlo simulation offers numerous applications in finance. The most common applications of the model in finance include Stock-market valuation, portfolio valuation, Interest rate derivatives, risk assessments and other financial modellings.

Consider a scenario where a sales company allocates a budget for next year's sales compensation. Depending on the percentage of actual sales compared to the sales target, compensation rates are calculated. Monte Carlo simulation is used to predict the range of potential values for a sales compensation budget.

Example of the Values for a five salesman

Sales Person	Sales Target (in Rupees)	Actual Sales (in Rupees)	Actual Sales/Sales Target	Commission Rate(Percentage)	Commission Amount (in Rupees)

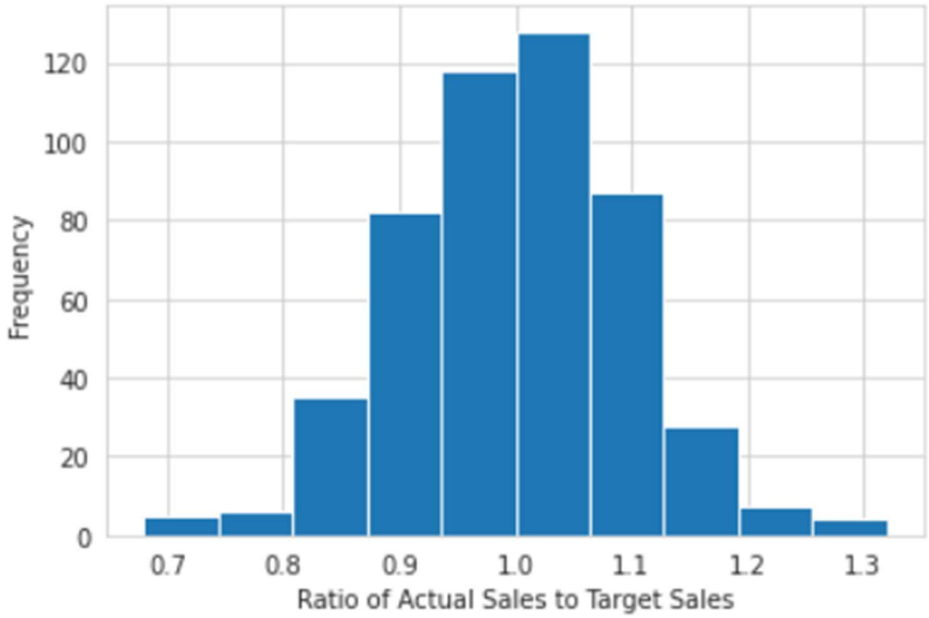
1	100,000	88,000	0.88	2	1760
2	200,000	202,000	1.01	4	8080
3	75,000	90,000	1.2	4	3600
4	400,000	360,000	0.9	2	7200
5	500,000	350,000	0.7	2	7000

Commission Amount=Actual Sales * Commission Percentage

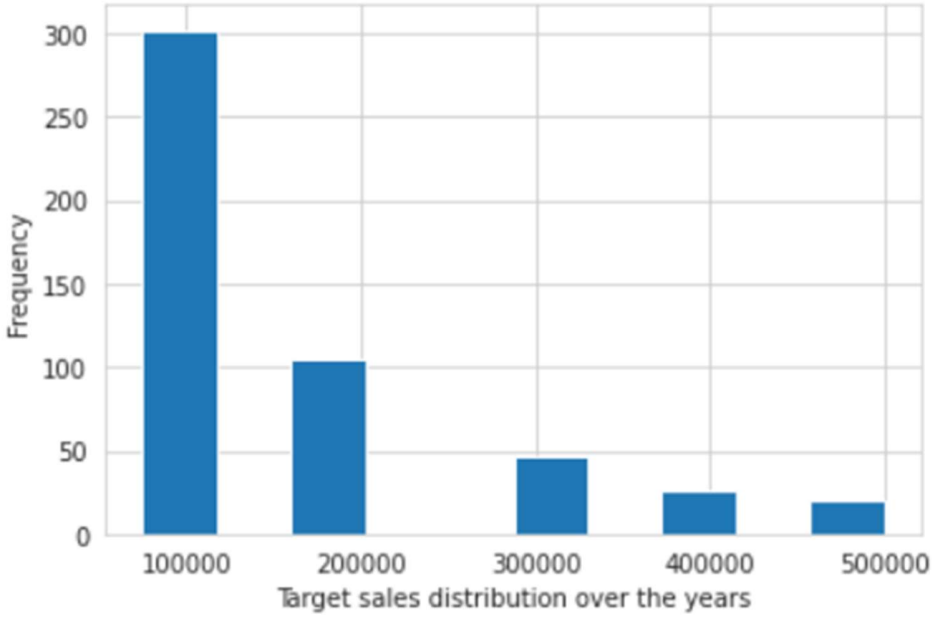
Commission Percentage depends on the ratio of actual sales to target sales.
If ratio >=1,commission rate is 4%
If ratio >0.9 and <1.0,commission rate is 3%
If ratio <=0.9,commission rate is 2%

These values are based on a lot of uncertainties and their estimation for the future can be analyzed using monte carlo simulation. It involves running many scenarios with random inputs and summarizing the distribution of the results.

Based on the data from the previous years, the ratio of sales to Target is obtained as:



From the data, it can be estimated that it follows a Normal Distribution and the Mean is 1.0 with a standard deviation of 0.1



From this distribution, it can be observed that most have Rs100,000 and Rs200,000 followed by Rs300,000 ,Rs 400,000 , Rs500,000

The above distributions of past data give us an inference on how to generate the data that will replicate the past year performances.

So, for the ratio of actual sales to target sales, we generate data that is normally distributed with mean 1 and standard deviation 0.1. For the Target sales, the frequency decreases as the sales amount increases and the sales target falls under any one of the six possible values. So, considering past data we use the uniform distribution but with probabilities decreasing as the amount increases.

Actual Sales Amount	Probability
75,000	0.3
100,000	0.3
200,000	0.2

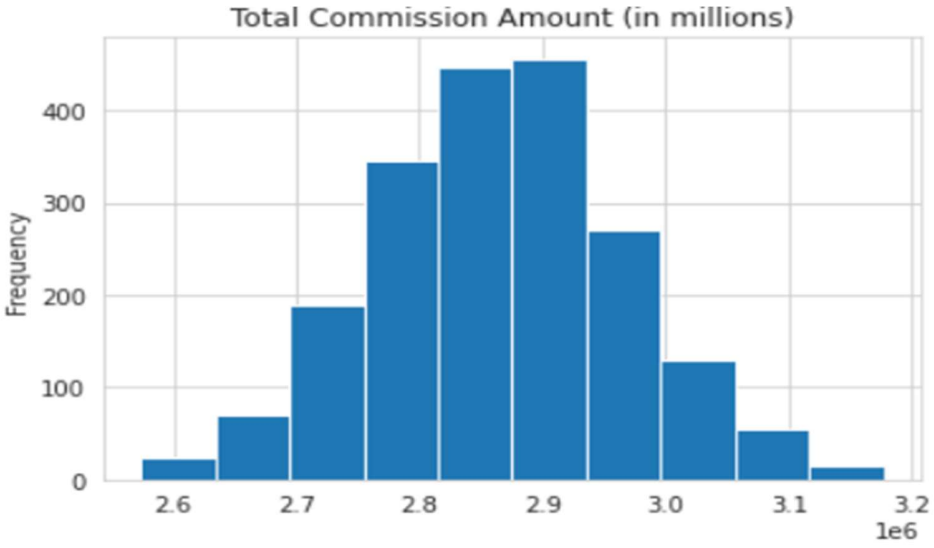
300,000	0.1
400,000	0.05
500,000	0.05

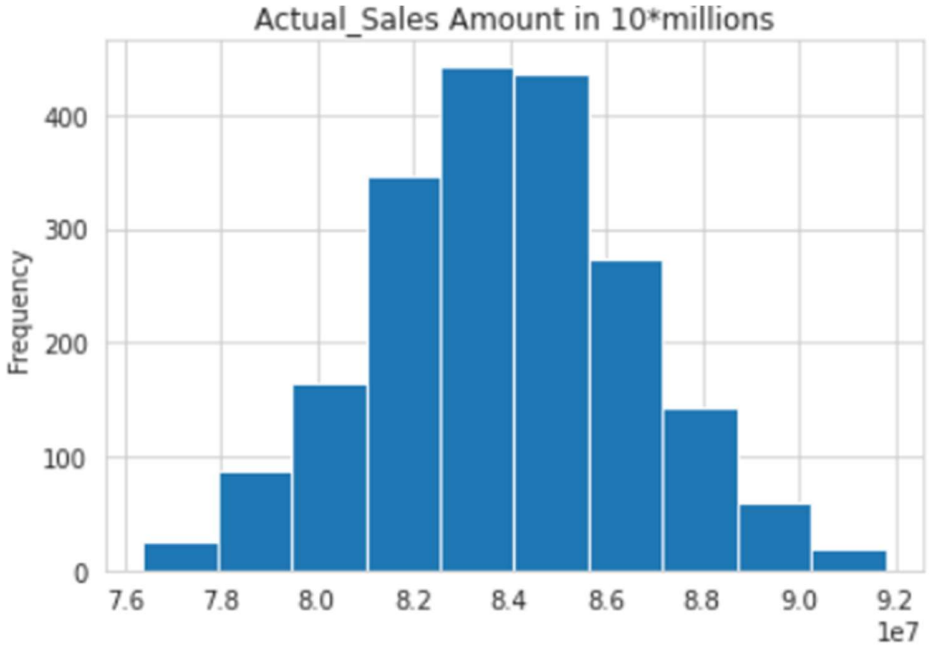
We generate the data using the above inferences obtained. The data looks like the following:

	ratio_actual_to_target	Sales_Target	Actual_Sales	Commission_Rate	Commission_Amount
0	0.94	100000	94000.0	0.03	2820.0
1	0.79	400000	316000.0	0.02	6320.0
2	0.97	200000	194000.0	0.03	5820.0
3	0.98	200000	196000.0	0.03	5880.0
4	0.90	100000	90000.0	0.02	1800.0

After running the simulations for 2000 iterations, we obtain the following results

	Actual_Sales	Commission_Amount	Sales_Target
count	2,000.0	2,000.0	2,000.0
mean	83,885,941.5	2,864,262.5565	83,888,000.0
std	2,679,801.974309539	101,909.40312982885	2,643,945.4369817977
min	76,384,500.0	2,573,575.0	76,475,000.0
25%	82,144,875.0	2,796,133.0	82,175,000.0
50%	83,862,000.0	2,864,536.5	83,825,000.0
75%	85,626,562.5	2,930,978.0	85,650,000.0
max	91,804,250.0	3,177,088.0	91,600,000.0





The results obtained from the simulation can be used to predict values for the future and it also allows us to change a few parameters like the distribution of target sales, decreasing the number of salespeople and Increasing the commission rates. Overall, by using Monte carlo simulation, we are able to predict future results by calculating a formula multiple times with different random inputs.

- **Importance Sampling:**
  - Importance sampling is a variance reduction technique that can be used in the Monte Carlo method. If these "important" values are emphasized by sampling more frequently, then the estimator variance can be reduced. Hence, the basic methodology in importance sampling is to choose a distribution which "encourages" the important values.
  - Basic Monte Carlo sampling evaluates

$$E[h(X)] = \int_X h(x)f(x)dx$$

Using another distribution g(x) - the so-called “importance function”, we can rewrite the above expression

$$E_f[h(x)] = \int_X h(x)\frac{f(x)}{g(x)}g(x)dx = E_g\left[\frac{h(X)f(X)}{g(X)}\right]$$

Giving us the new estimator

$$\bar{h_n} = \frac{1}{n}\sum_{i=1}^{n}\frac{f(x_i)}{g(x_i)}h(x_i)$$

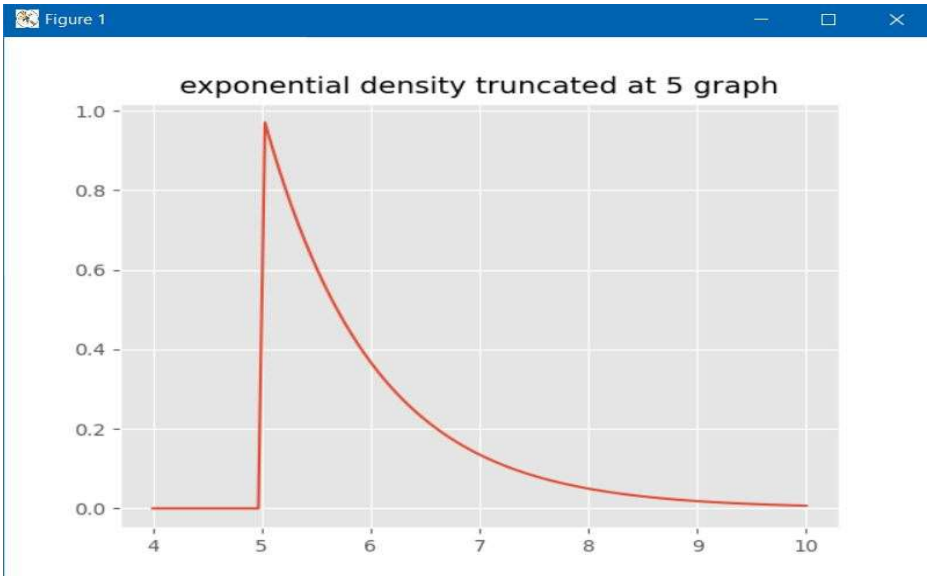
where xi~g is a draw from the density g. Conceptually, what the likelihood ratio f(xi)/g(xi) provides an indicator of how “important” the sample

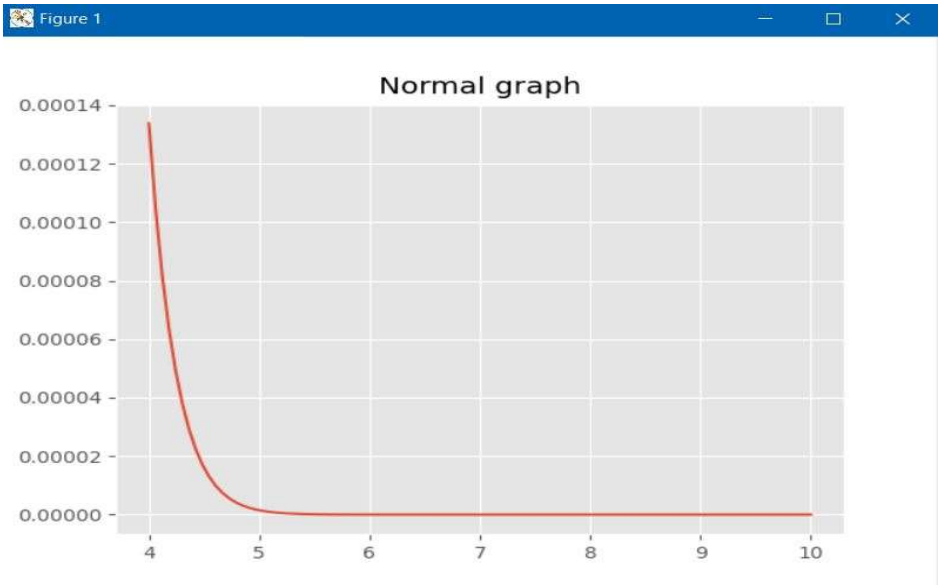
h(xi) is for estimating h_n^- . This is very dependent on a good choice for the importance function g.

Example_1:

Suppose we want to estimate the tail probability of N(0,1) for P(X>5).

Regular MC integration using samples from N(0,1) is hopeless since nearly all samples will be rejected. However, we can use the exponential density truncated at 5 as the importance function and use importance sampling.





Output:-

```
Expected answer

2.866515719235352e-07

------------------------------------------------------------
Using direct Monte Carlo integration

estimate and relative error
0.0 1.0

Using importance sampling

estimate and relative error
5.0239240721518184e-08 0.824737606061621

n value is :  2000
------------------------------------------------------------
```

And So on ….

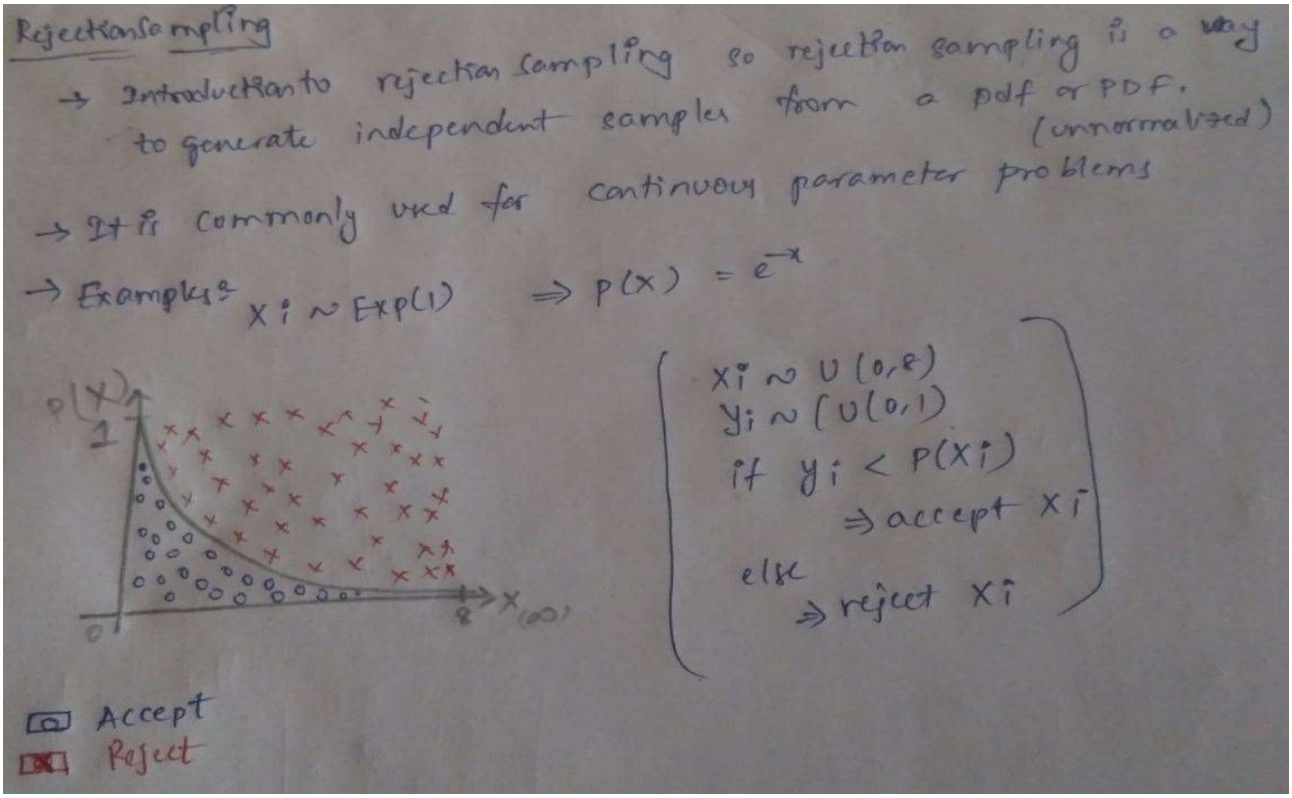
```
------------------------------------------------------------
Using direct Monte Carlo integration

estimate and relative error
0.0 1.0

Using importance sampling

estimate and relative error
2.8706618493730222e-08 0.8998553599371583

n value is :  8000
------------------------------------------------------------
```

- **Rejection Sampling:**
  - Rejection Sampling is a way to generate independent samples from a probability distribution function or a probability density function that is potentially unnormalized.
  - It is also commonly called as “acceptance-rejection method” or “accept-reject algorithm” and is a type of exact simulation method

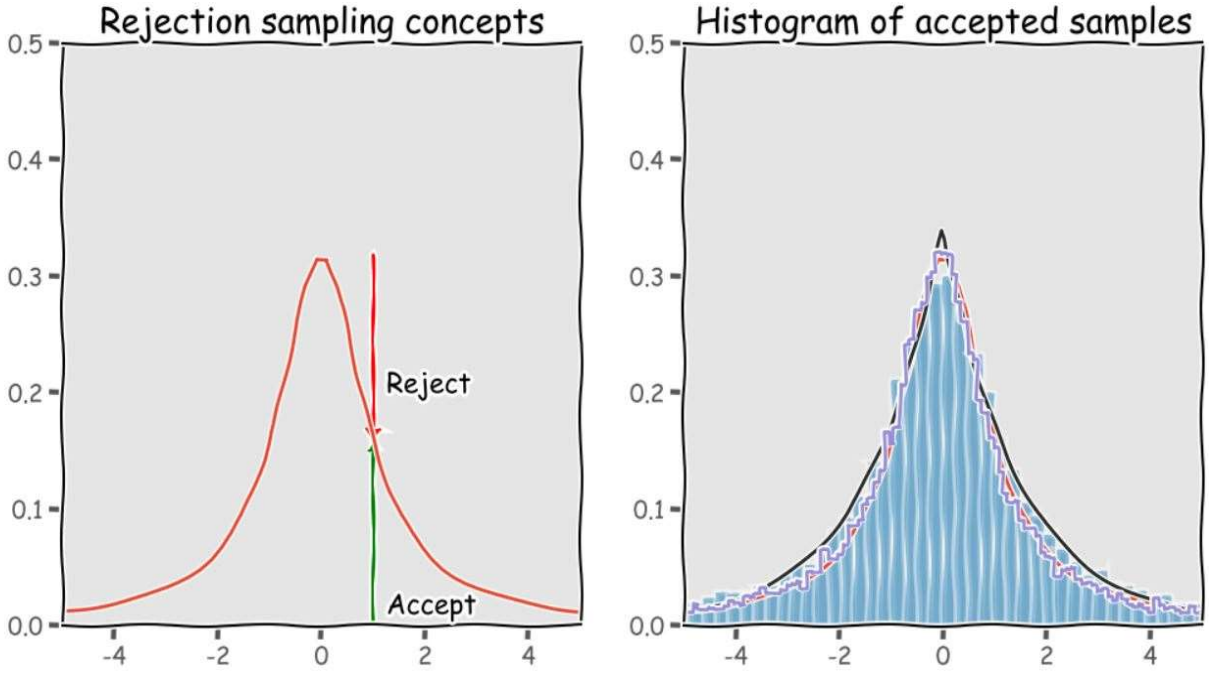
Drawbacks:

- This can lead to a lot of unwanted samples being taken if the function being sampled is highly concentrated in a certain region(ex:-A function that has a spike at some location).

Theory and Example_1:-.



Example_2: Using Cauchy continuous random variable



Markov Chain Monte Carlo Methods

The problem with Monte Carlo sampling is that it does not work well in high-dimensions. The volume of the sample space increases exponentially with the number of dimensions and hence leading to the curse of dimensionality. Also, Monte Carlo sampling assumes each random sample drawn from the target distribution is independent which may not be the case for inference with Bayesian Structured probabilistic models.

So, for sampling probability distributions in high dimensions, we use Markov Chain Monte Carlo (MCMC). This allows a combination of Markov Chain and Monte Carlo which allows random sampling of high dimensionality probability functions that honors the probabilistic dependence between samples by constructing a Markov Chain that comprise the Monte Carlo sample.

Markov chain Monte Carlo (MCMC) methods comprise a class of algorithms for sampling from a probability distribution. By constructing a Markov chain that has the desired distribution as its equilibrium distribution, one can obtain a sample of the desired distribution by recording states from the chain. The more steps that are included, the more closely the distribution of the sample matches the actual desired distribution. Various algorithms exist for constructing chains, including the Metropolis-Hastings algorithm.

Applications of Markov Chain Monte Carlo methods:

- Data Mining and Machine Learning:

To carry out the **Metropolis-Hastings** **random-walk algorithm**, we need to draw random samples from the following distributions

- The standard **uniform distribution**
- A **proposal distribution** p(x) that we choose to be N(0,σ).
- The target distribution g(x) which is proportional to the **posterior probability**

Given an initial guess for θ with positive probability of being drawn, the Metropolis-Hastings algorithm proceeds as follows:-

- Choose a new proposed value (θp) such that $\theta p=\theta+\Delta\theta$ where $\Delta\theta \sim N(0,\sigma)$
- Calculate the ratio

$$\rho = \frac{g(\theta_p \mid X)}{g(\theta \mid X)}$$

Where g is the posterior probability.

- If the proposal distribution is not symmetrical, we need to weigh the acceptance probability to maintain detailed balance(reversibility) of the stationary distribution and instead calculate.

$$\rho = \frac{g(\theta_p \mid X)p(\theta \mid \theta_p)}{g(\theta \mid X)p(\theta_p \mid \theta)}$$

- Since we are taking ratios, the denominator cancels any distribution proportional to g will also work - so we can use

$$\rho = \frac{p(X|\theta_p)p(\theta_p)}{p(X|\theta)p(\theta)}$$

- If ρ≥1, then set θ=θp
- If ρ<1, then set θ=θp with probability ρ, otherwise set θ=θ (this is where we use the standard uniform distribution)
- Repeat the earlier steps

An acceptance probability that meets this condition is(applied in code also)

$$\min\left(1, \frac{g(\theta_p \mid X)p(\theta \mid \theta_p)}{g(\theta \mid X)p(\theta_p \mid \theta)}\right)$$

Rigorous demonstration of convergence is an unsolved problem, but simple ideas such as running **multiple chains** and checking that they are **converging to similar distributions** are often employed...

Output:--

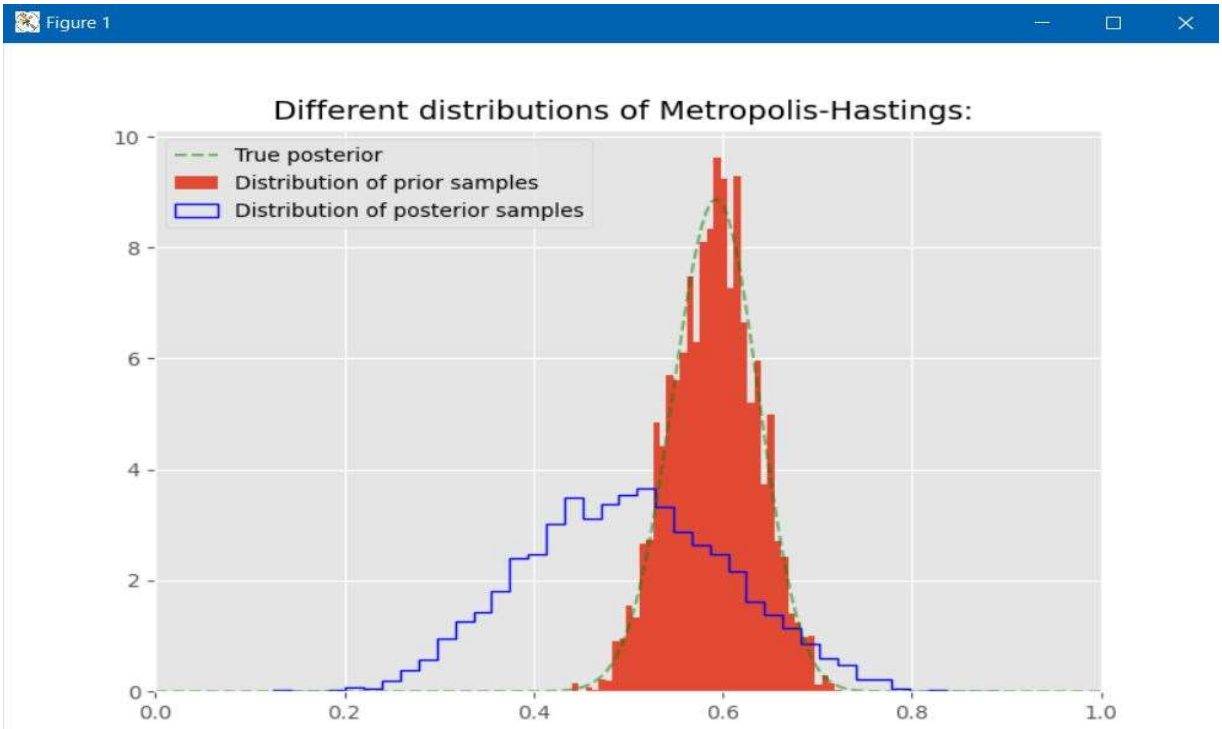


Figure 1
Different distributions of Metropolis-Hastings:
True posterior
Distribution of prior samples
Distribution of posterior samples
10
8
6
4
2
0
0.0
0.2
0.4
0.6
0.8
1.0

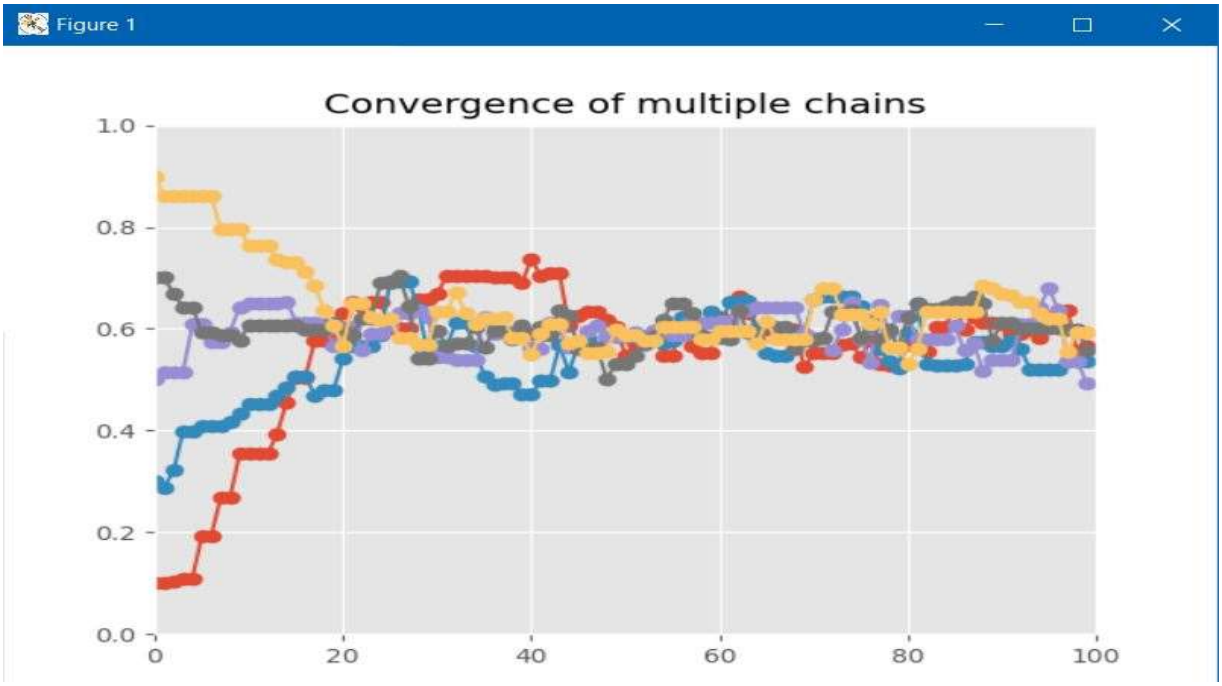


Figure 1
Convergence of multiple chains
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

Gibbs Sampler:-

Suppose we have a vector of parameters θ=(θ1,θ2,…,θk), and we want to estimate the joint posterior distribution p(θ|X). Suppose we can find and draw random samples from all the conditional distributions

$$p(\theta_1|\theta_2,\ldots\theta_k,X)$$
$$p(\theta_2|\theta_1,\ldots\theta_k,X)$$
$$\ldots$$
$$p(\theta_k|\theta_1,\theta_2,\ldots,X)$$

With Gibbs sampling, the Markov chain is constructed by sampling from the conditional distribution for each parameter θi in turn, treating all other parameters as observed. When we have finished iterating over all parameters, we are said to have completed one cycle of the Gibbs sampler. Where it is difficult to sample from a conditional distribution, we can sample using a Metropolis-Hastings algorithm instead - this is known as Metropolis within Gibbs.

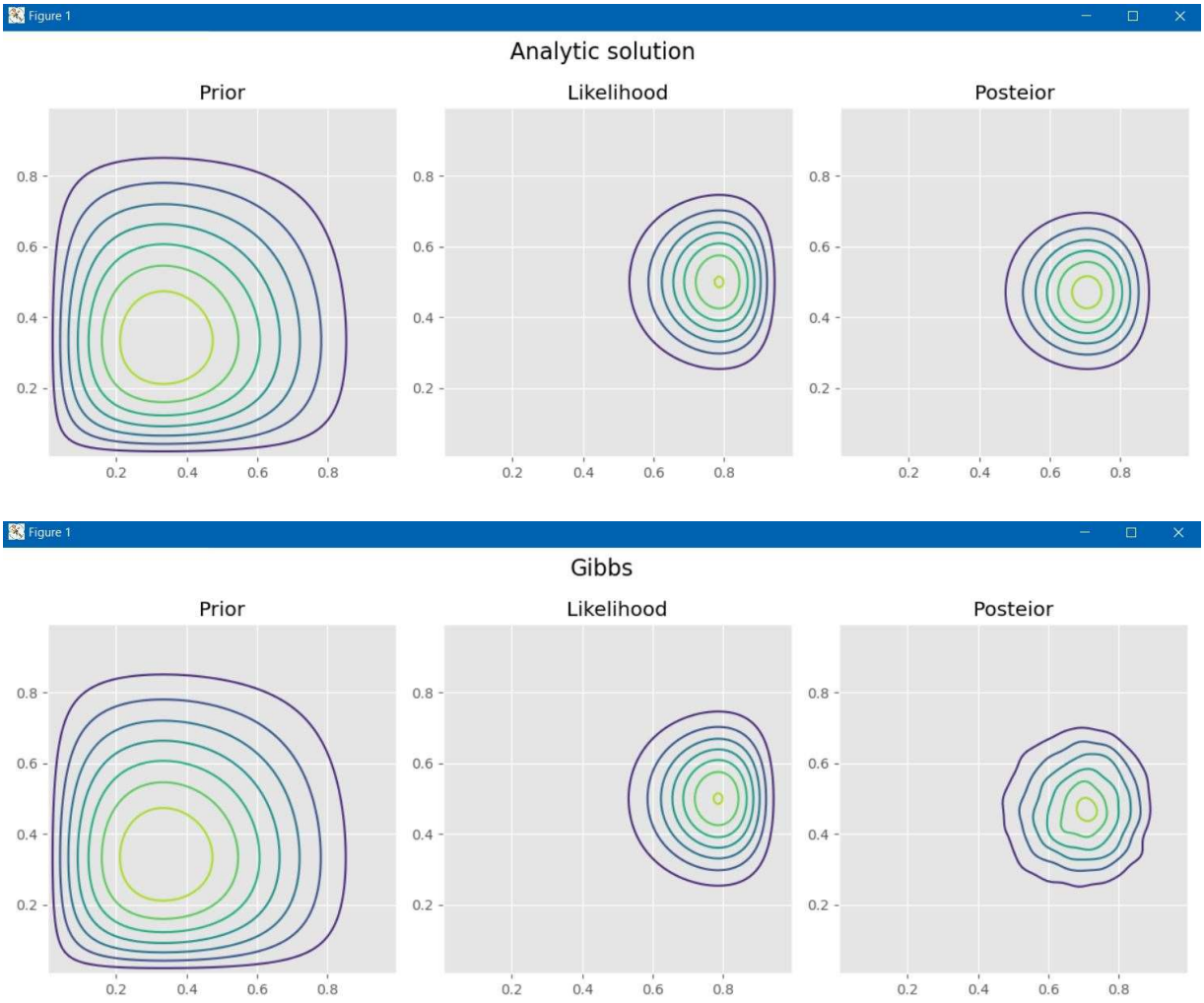
Drawbacks of using Gibbs Samplings:-

1) Long Convergence time especially with the dimensionality of the data growing. Convergence time also depends on the shape of the distribution.
2) Difficulty in finding the posterior for each variable.

Example_1:

We will use the toy example of estimating the bias of two coins given sample pairs (z1,n1) and (z2,n2) where zi is the number of heads in ni tosses for coin i.

Output:--

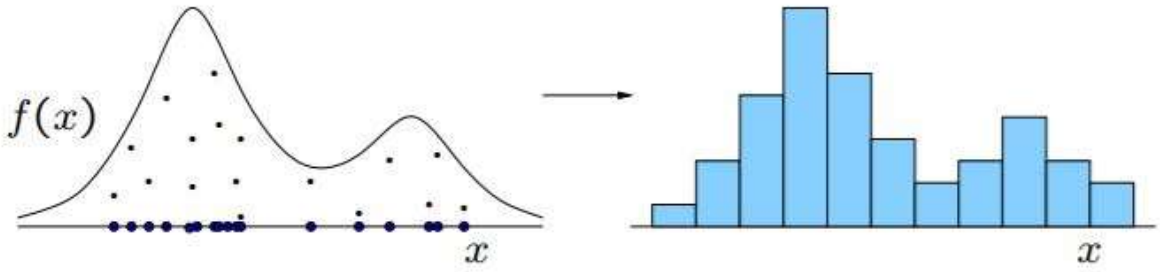


Slice sampler:-

Yet another MCMC algorithm is slice sampling. In slice sampling, the Markov chain is constructed by using an auxiliary variable representing slices through the (unnormalized) posterior distribution that is constructed using only the current parameter value. Like Gibbs sampling, there is no tuning process and all proposals are accepted. For slice sampling, you either need the inverse distribution function or some way to estimate it.

A toy example:-- illustrates the process - Suppose we want to draw random samples from the posterior distribution N(0,1) using slice sampling Start with some value x - sample y from U(0,f(x)) - this is the horizontal “slice” that gives the method its name - sample the next x from f−1(y) - this is typically done numerically - repeat

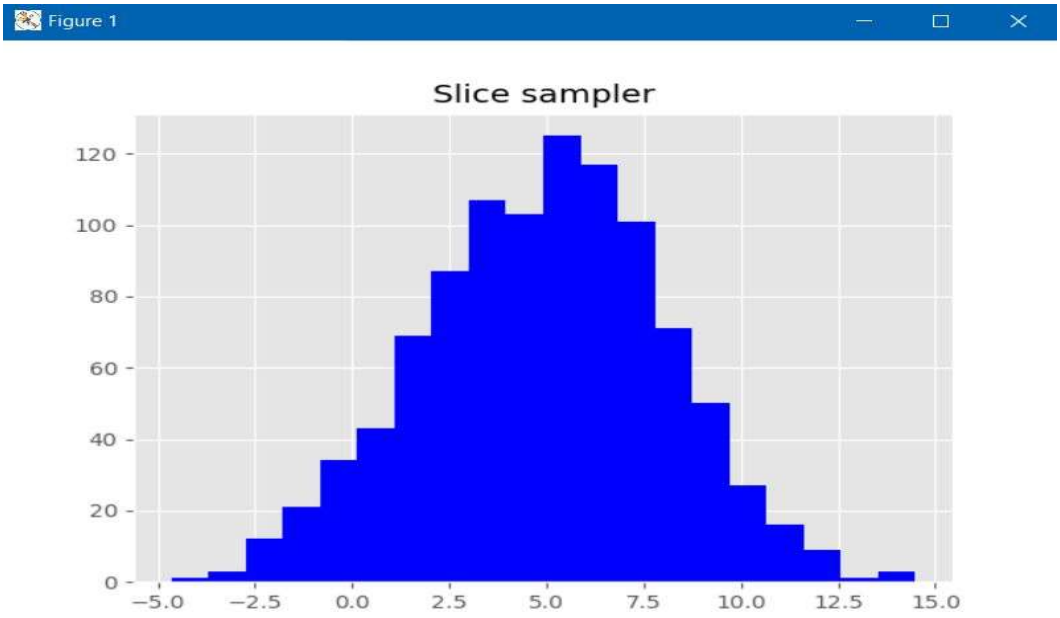
To sample from a distribution, simply sample uniformly from the region under the density function and consider only the horizontal coordinates.



One way to do this is to

- introduce latent (auxiliary) variables,
- then use Gibbs sampling on the area beneath the density

Output:--



Applications of Markov Chain Monte Carlo methods:

- Data Mining and Machine Learning: They have various unusual and weird distributions that cannot be solved deterministically. Computer vision makes heavy use of MCMC
- Bayesian Methods require multiplication of prior distribution with some likelihood distribution which result in messy distributions. MCMC is used in these cases.
- Biological and Genetic Research: The human genome is huge(big data) but usually people work with small data. So, smaller samples are drawn and combined at the end. Also, this field heavily uses Bayesian Methods and hence MCMC is very popular in Biological and Genetic Research.

Decrypting Substitution Ciphers using MCMC:

A substitution cipher is the one in which every symbol is mapped bijectivity over a set of arbitrary symbols. If "n" is the cardinality of the domain set, the key for decryption can be any of the n! Combinations.

The transition Matrix of all pairs in the alphabet space is obtained by using the first order transitions of "War and Peace" pdf.

$$Pl(f) = \prod_{i \in \mathcal{S}} M(f(s_i), f(s_i + 1))$$

Where functions f which have high values of Pl(f) are good candidates for decryption.

Algorithm:

- We start with a preliminary guess f
- Compute Pl(f)
- Choose random f* and compute Pl(f*)
- If Pl(f*) > Pl(f), accept f*
- Else, toss a coin and if heads, accept f*
- Keeping iterating through the above steps until desired results are obtained.

Simulated Annealing:

Simulated Annealing is a probabilistic technique for approximating the global optimum of a given function. Specifically, it is a metaheuristic to approximate global optimization in a large search space for an optimization problem.

It is analogous to Annealing in Metallurgy where metal is heated and cooled to increase the size of the crystals and reduce their defects. A similar analogy is followed in Simulated Annealing where slow cooling is interpreted as slow decrease in the probability of choosing the worst decisions. Following the analogy, the temperature is decreased from initial positive value to zero. At each time step, the algorithm randomly selects a solution close to the current one, measures its quality, and moves to it according to the temperature-dependent probabilities of selecting better or worse solutions, which during the search respectively remain at 1 (or positive) and decrease towards zero.

Applications of Simulated Annealing:

1)Travelling Salesman Problem: A list of cities and distances between each pair of cities is given. We need to calculate the shortest possible route that visits each city and returns to the origin city.

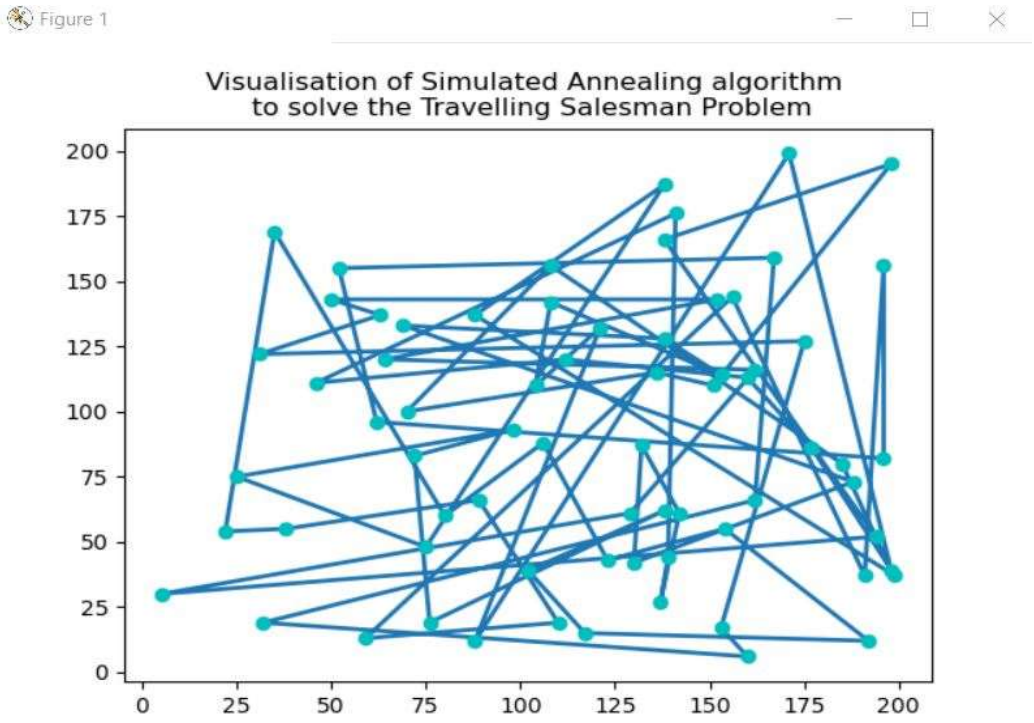
The Algorithm:

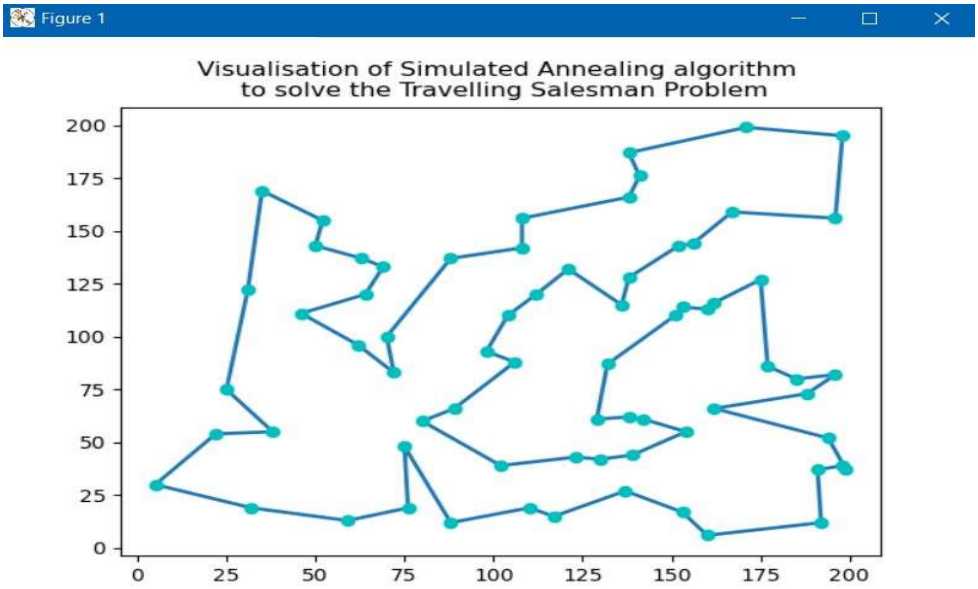
1) Start with a random tour through the selected cities.
2) Pick a new candidate tour at random from all neighbors of the existing tour.
3) If the candidate tour is better than the existing tour, then accept it as the new one
4) If the candidate tour is worse than the existing tour, we still need to accept it and assign a probability to it. The probability of accepting an inferior tour is a function of how much longer the candidate is compared to the current tour. A higher temperature makes you more likely to accept an inferior tour.

5) We need to iterate through the second step so many times till we are satisfied with our answer. After each iteration, the temperature is cooled. We need to change the cooling schedule if we are not satisfied with the result.

   Though Simulated Annealing doesn't always guarantee of a global optimum every time, it is better than the other Naive methods as the mean distance obtained by performing Simulated Annealing is less.

   Sample Simulations :-





2) Simulated Annealing can also be applied to scheduling.

Consider University Timetabling where exams are conducted over a period and multiple students have different subjects. It is not easy to always devise the timetable manually. Simulated Annealing can be applied to such case.

The Algorithm follows the following steps:

- We need to generate a random partial solution. Each partial solution can be considered as a state in MCMC.
- The Randomly generated solution has to be meaningful to an extent to avoid extensive calculations.
- A new state has to be randomly generated and the cost of it has to be compared with the existing state.
- If the new state is better, we have to choose the new state as it decreases the cost.

- If the existing state is better, cost will increase and we still have to assign a probability of considering to move to the new state. In this case, we choose a random number and we compare it with the quantity obtained using probability function and move depending on the comparison.
- We have to repeat the above steps till we achieve the desired result.

Reference links:-

https://bookdown.org/rdpeng/advstatcomp/rejection-sampling.html
http://www.mit.edu/~ilkery/papers/GibbsSampling.pdf
https://www.youtube.com/watch?v=7THzVD7jxiU&list=PLwJRxp3blEvZ8AKMXOy0fc0cqT61GsKCG&index=52
https://www.youtube.com/watch?v=V8f8ueBc9sY&list=PLwJRxp3blEvZ8AKMXOy0fc0cqT61GsKCG&index=56
https://www.youtube.com/watch?v=U561HGMWjcw
https://wiseodd.github.io/techblog/2015/10/17/metropolis-hastings/
https://www.youtube.com/watch?v=ER3DDBFzH2g&list=PLwJRxp3blEvZ8AKMXOy0fc0cqT61GsKCG&index=67

Thank you...