Advanced Statistical Methods (ASM)

Monte Carlo Methods and Markov Chain Monte Carlo Methods

Group- 15

Sowmya Vasuki J - S20160010035
Anjali Poornima K - S20160020132
Sai Amulya M - S20170010099

Introduction

Markov Chain

A Markov chain is a mathematical system that experiences transitions from one state to another according to certain probabilistic rules. A Markov chain essentially consists of a set of transitions, which are determined by some probability distribution, that satisfy the Markov property. Markov property is the unique characteristic of Markov processes that renders them memoryless.

The probability distribution of state transitions is typically represented as the Markov chain's transition matrix. If the Markov chain has N possible states, the matrix will be an N x N matrix, such that entry (I, J) is the probability of transitioning from state I to state J. Additionally, the transition matrix must be a stochastic matrix, a matrix whose entries in each row must add up to exactly 1. This makes complete sense, since each row represents its own probability distribution.

Monte Carlo

Monte Carlo methods are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. The underlying concept is to use randomness to solve problems that might be deterministic in principle.

In principle, Monte Carlo methods can be used to solve any problem having a probabilistic interpretation. By the law of large numbers, integrals described by the expected value of some random variable can be approximated by taking the empirical mean (a.k.a. the sample mean) of independent samples of the variable.

The main idea behind this method is that the results are computed based on repeated random sampling and statistical analysis.

Importance Sampling

Importance sampling is a general technique for estimating properties of a particular distribution, while only having samples generated from a different distribution than the distribution of interest. The idea behind importance sampling is that certain values of the input random variables in a simulation have more impact on the parameter being estimated than others. If these "important" values are emphasized by sampling more frequently, then the estimator variance can be reduced

Our goal is to compute $I(f) = \int f(x)p(x)dx$. If we have a density $q(x)$ which is easy to sample from, we can sample $x(i)$ iid ~ $q(x)$.

Rejection Sampling

This sampling is used in Monte Carlo methods when direct sampling from density function is not possible. Rejection sampling is based on the observation that to sample a random variable in one dimension, one can perform a uniformly random sampling of the two-dimensional Cartesian graph, and keep the samples in the region under the graph of its density function.

To visualize the motivation behind rejection sampling, imagine graphing the density function of a random variable onto a large rectangular board and throwing darts at it. Assume that the darts are uniformly distributed around the board. Now remove all of the darts that are outside the area under the curve. The remaining darts will be distributed uniformly within the area under the curve, and the x-positions of these darts will be distributed according to the random variable's density.

Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) methods comprise a class of algorithms for sampling from a probability distribution. MCMC methods create samples from a continuous random variable, with probability density proportional to a known function. These samples can be used to evaluate an integral over that variable, as its expected value or variance.

Practically, an ensemble of chains is generally developed, starting from a set of points arbitrarily chosen and sufficiently distant from each other. These chains are stochastic processes of "walkers" which move around randomly according to an algorithm that looks for places with a reasonably high contribution to the integral to move into next, assigning them higher probabilities.

Gibbs Sampling

Gibbs sampling or a Gibbs sampler is a Markov chain Monte Carlo (MCMC) algorithm for obtaining a sequence of observations which are approximately from a specified multivariate probability distribution, when direct sampling is difficult. Gibbs sampling is applicable when the joint distribution is not known explicitly or is difficult to sample from directly, but the conditional distribution of each variable is known and is easy (or at least, easier) to sample from.

Simulated Annealing

Simulated annealing (SA) is a probabilistic technique for approximating the global optimum of a given function. Specifically, it is useful to approximate global optimization in a large search space for an optimization problem. It is often used when the search space is discrete (e.g., the traveling salesman problem).

At each virtual annealing temperature, the simulated annealing algorithm generates a new potential solution (or neighbour of the current state) to the problem considered by altering the current state, according to a predefined criterion. The acceptance of the new state is then based on the satisfaction of the Metropolis criterion, and this procedure is iterated until convergence.

Implementation

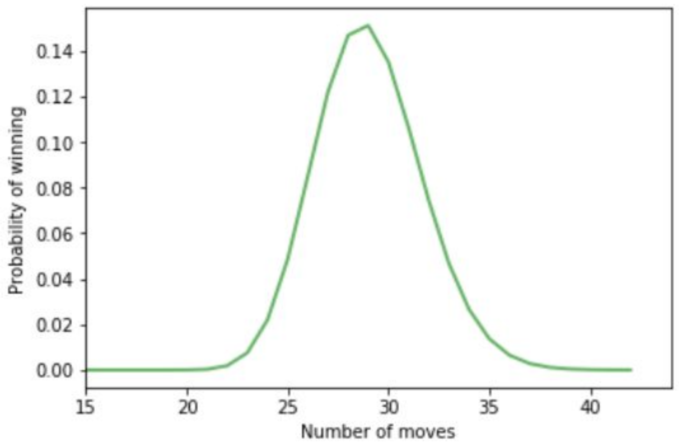
Markov Chain

Problem Solved: Snakes and Ladders Game

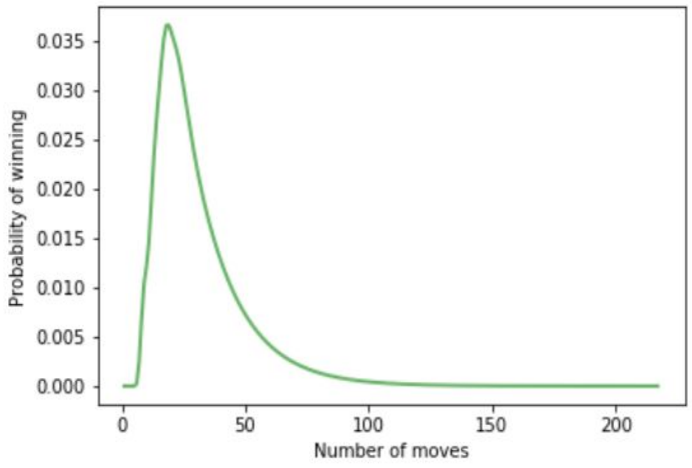
The game of Snakes and Ladders is a good candidate for analysis with a Markov Chain because of its memorylessness: at a given point in the game, the player's progression from the current square is independent of how they arrived at that square. Two problems are analysed in this regard; Game with and without Snakes and ladders.

In Markov Chain theory, the probability of a move from square i to square j is given by a transition matrix,T. The first row of the transition matrix therefore represents the probabilities of moving to each square from square 0, the start; the second row represents the probabilities of moving to each square from square 1, and so on. The moves are decided by the roll of a fair die with six faces.

The game can be analysed with a row vector, v with 101 components, representing the probabilities that the player is on each of the squares. Initially, v(0) is (1,0,0,⋯,0) : the player is certainly on square 0 before the game has begun. Subsequently, v evolves by the relation v(k+1) = v(k)T. That is, the probabilities for the next move, k+1, are given by the dot product of the current state vector, v(k) and the transition matrix, T.



Plot showing the probability of winning against number of moves for the **game without snakes and ladders**



Plot showing the probability of winning against number of moves for the **game with snakes and ladders**

Monte Carlo

Problem Solved: Average waiting time of customers at a restaurant

Assumptions:

- Average arrival of customers is 3 per minute i.e on average there will be one customer for every 20 sec.
- Customers can order 1 to 10 items and the cashier takes one second to return each item to the customer.
- Time frame of 600 sec is considered.
- Items served by cashier ranges from 10 to 40

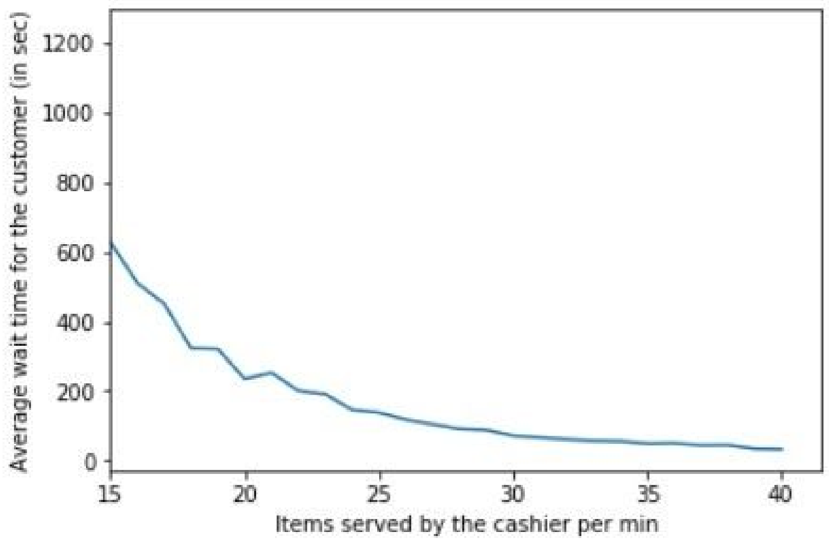
For each second in the time frame, if a random number generated between 1 to 20 is 20 a new customer is created and he/she is added to the queue. If the cashier is not busy and the queue is not empty, a customer in the queue will be served.

Waiting time of customer = service time - arrival time

Cashier will be set idle if all the items ordered by the current customer are served and the counter is available for the next customer.

The average waiting time of customers is simulated for each case of items served per minute by the cashier (i.e., 10,11,..40) over 100 times.

From the simulation results, if the number of items served per min are increased (i.e., the numbers of cashiers at the counter) it can be observed that more number of customers are being served and the average waiting time for the customers can be reduced.



Importance Sampling

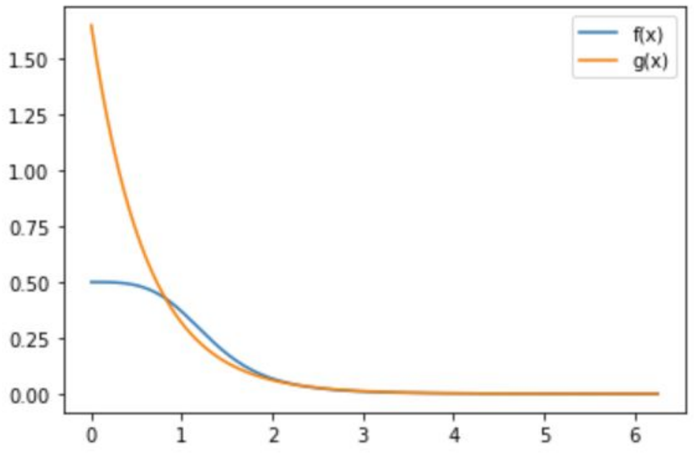
Using an example function f(x)

The Integral being sampled is below, where f(x) is the function

$$I = \int_0^{inf} \frac{e^{-x}}{1 + (x-1)^2} dx$$

Now a g(x) needs to be considered, which helps in estimating f(x) in its active zone accurately

$$g(x) = Ae^{-\lambda x}$$



First, the accurate A, λ. For that, choose the λ, with lowest variance. Here, the variance is (Sum of squares - Squared average) of f(x)/g(x) for every λ.

Now that the optimal λ is obtained, run the simulation and compute the below value

$$I \approx \frac{1}{N} \sum_i^N \frac{f(G^{-1}(r_i))}{g(G^{-1}(r_i))}$$

For every iteration (num of samples) and the approx value is the total sum/num_samples.

This is computed with the optimal λ. And hence we get the required integral value.

Importance Sampling Approximation: 0.6945567812486738

Variance: 0.04544894505293784
Error: 0.002131875818450452

Rejection Sampling

Using an example function f(x)

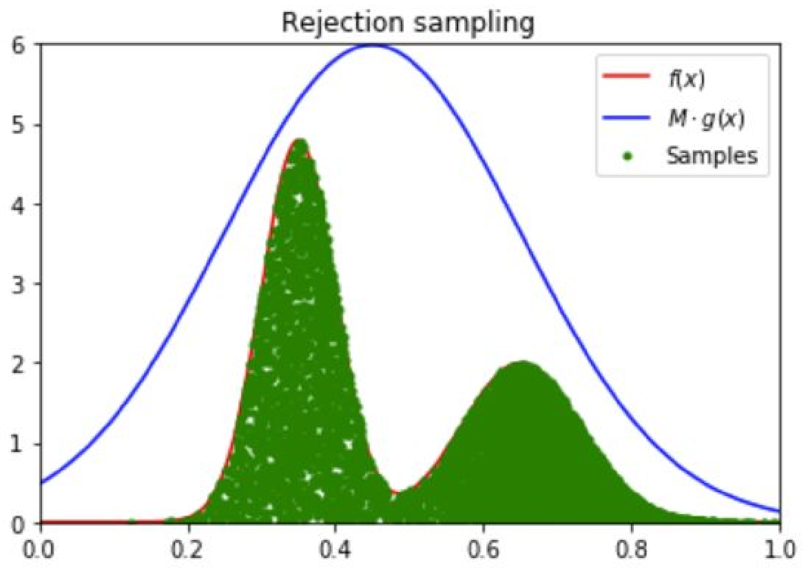
The target distribution f(x) = 0.6 * norm.pdf(x, 0.35, 0.05) + 0.4 * norm.pdf(x, 0.65, 0.08)

The Approximated PDF g(x)= lambda x: norm.pdf(x, 0.45, 0.2)

Now we take the Number of samples to be M * np.random.normal(0.45, 0.2, (N,)). Amongst all the samples, choose samples such that for each sample si, accept if

$u(i) < f(si)/M * g(si)$

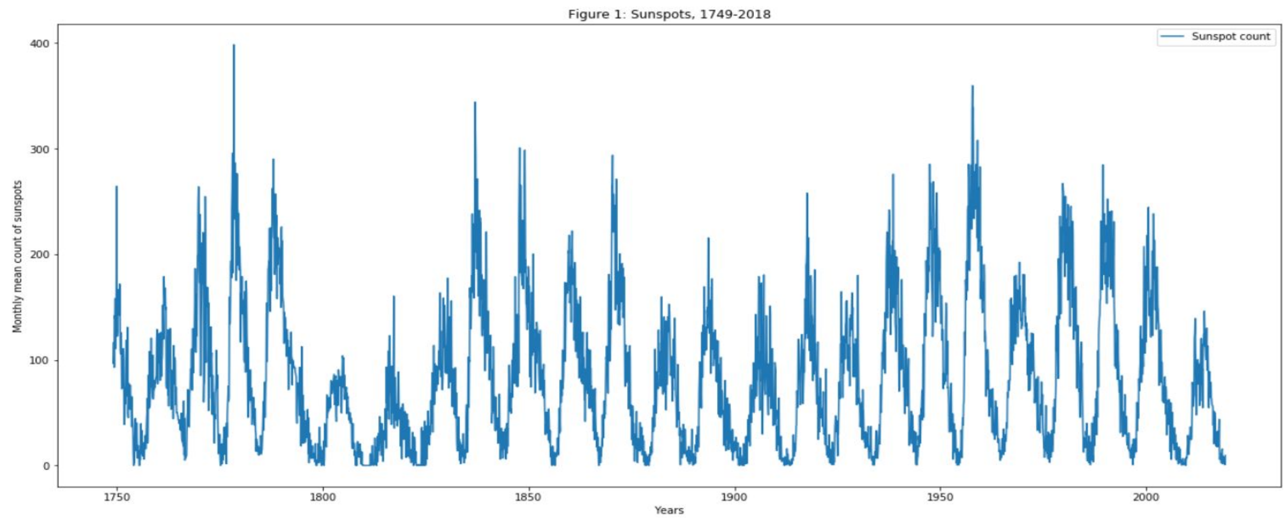
Now, on plotting all the accepted samples, the target distribution and the approximated PDF



Markov Chain Monte Carlo

Problem Solved : Parameter estimation for Sunspots distribution

MCMC is a class of techniques for sampling from a probability distribution and can be used to estimate the distribution of parameters given a set of observations. A sunspot is a region on the Sun's surface (photosphere) that is marked by a lower temperature than its environment. Their number varies according to the approximately 11-year solar cycle. The data we worked on is the "Monthly mean total sunspot number", for each month from January 1749 to November 2018.



This phenomenon has been modelled with a gamma distribution, with a new cycle resetting every 12 years. A random variable X that is gamma-distributed is noted X~Γ(a, b), and in this case X is the count of sunspots. The two parameters a and b are the unknowns that are to be calculated.

The gamma distribution is for PDF, $f \,/\, f(x;\, a,\, b) = \dfrac{b^a x^{a-1} e^{-bx}}{\Gamma(a)}$ where $\boldsymbol{\Gamma}$ is the gamma function.

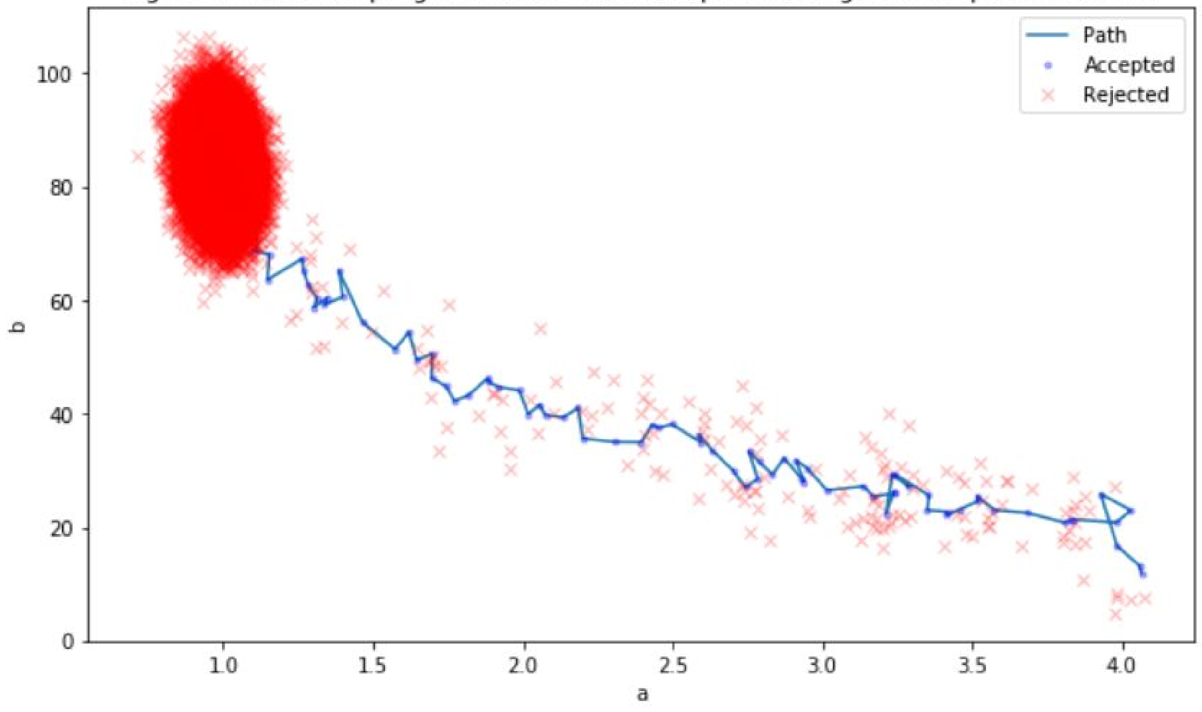
Definition of acceptance in MCMC:

$$\frac{Likelihood(D / \mu_{obs}, \sigma_{new}) * prior(\mu_{obs}, \sigma_{new})}{Likelihood(D / \mu_{obs}, \sigma_{current}) * prior(\mu_{obs}, \sigma_{current})} > 1$$

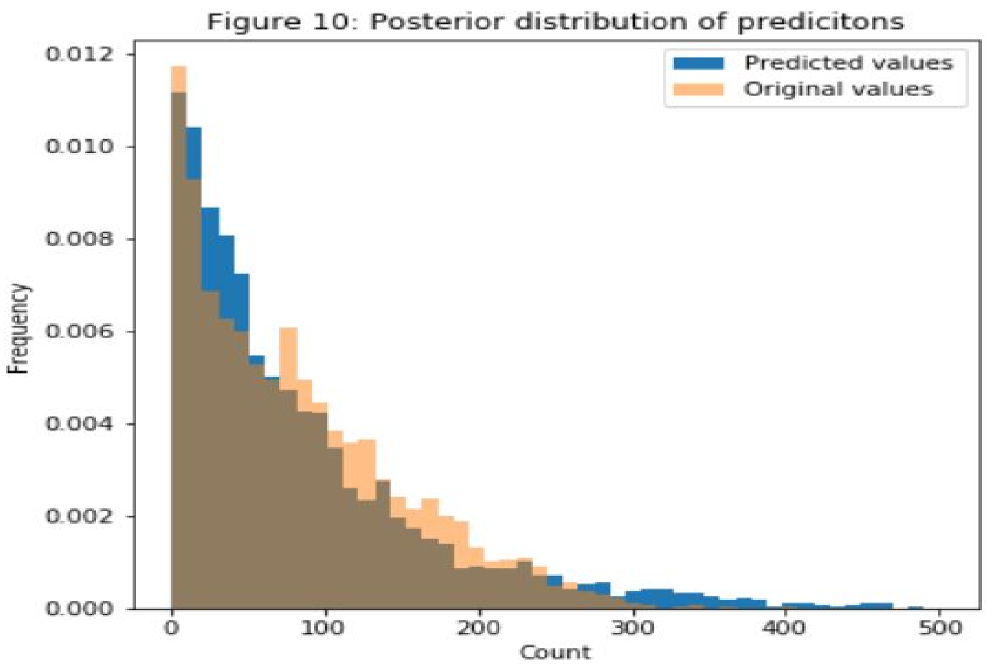
If this ratio is smaller or equal to 1, then compare it to a uniformly generated random number in the closed set [0,1]. If the ratio is larger than the random number, accept new value, otherwise we reject it.

Finally the sampling of Metropolis-Hastings (MCMC) for both the parameters is shown below:

Figure 4: MCMC sampling for *a* and *b* with Metropolis-Hastings. All samples are shown.



Finally the Posterior distribution of the predictions are plotted.



Gibbs Sampling

Problem Solved : Image Denoising

There are two images X, the noisy image and Y , the denoised version. First start with calculating the posteriors p(Y=1|Y_neighbor).

After loading the noisy image X and randomly initializing the restored image Y, start a loop to sample Y and calculate posterior probabilities P(Y|Y_neighbor).

At each step, iterate over i, j to sample each pixel yij in Y by calling function sample_y(), and update Y with the sampled value yij. In the function sample_y(), the four neighbours are used to sample and then the probability is calculated.

$$P(y_{ij}=1|y_{\mathrm{N}(ij)}) = \frac{P(y_{ij}=1, y_{\mathrm{N}(ij)})}{\sum_{y_{ij}} P(y_{ij}, y_{\mathrm{N}(ij)})} = \frac{1}{1+exp\,(-2w_{ij})}$$

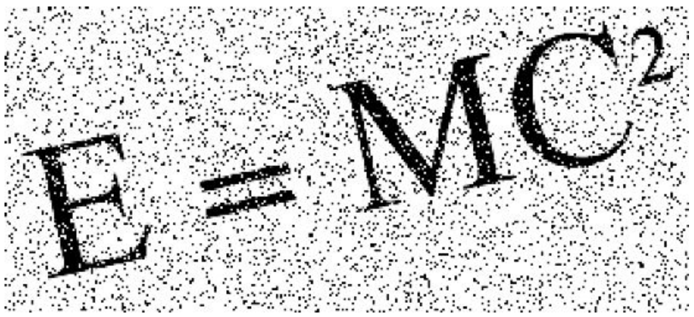
$$\text{where } w_{ij} = \eta x_{ij} + \beta \sum_{\mathrm{N}(ij)} y_{\mathrm{N}(ij)}$$

When the burn-in period ends, sum up the total occurrences of the event that yij = 1, for yij in Y. The energy is also calculated for checking the convergence.

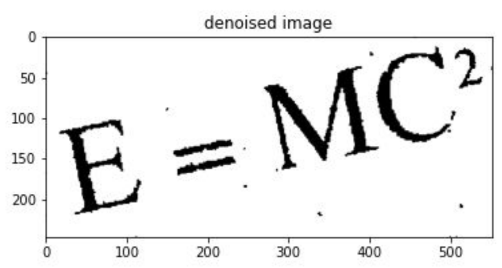
$$-logP(X|Y) - logP(X) = -\eta \sum_{i=1}^{N}\sum_{j=1}^{M} x_{ij}y_{ij} - \beta \sum_{i'j' \in N(ij)} y_{ij}y_{i'j'}$$

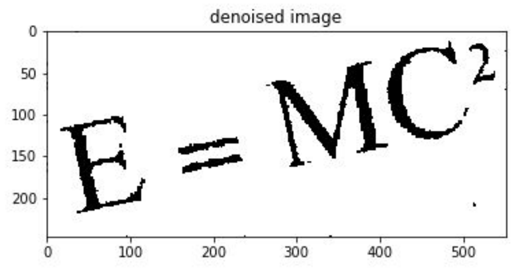
Once sampling is completed, the Monte Carlo method is used to get the posterior probabilities, which is essentially dividing the aggregated values of Y by the number of total samples.

Now after the posteriors are calculated, threshold them at 0.5, to obtain the denoised image.



Noisy Image





Denoise Image with num_samples=100 Denoised Image with num_samples=1000

Simulated Annealing

Problem Solved : Traveling Salesman Problem

Simulated Annealing is a stochastic global optimization algorithm. Considering the traveling salesman problem, the energy distribution for this problem is easy to express in a format suitable for SA, P (path) = $e^{-dist(path)/T}$

where path is the ordered list of cities to visit, dist is the function that computes path distance. The main trick to solve traveling salesman with SA is to pick a proper proposal distribution. The common proposal policy for this problem is the following

1. Pick two cities in the path
2. Exchange their positions in the path
3. Return the new proposed path

Simulated Annealing is compared for four different values of annealing rate: 0.9 (fast), 0.95 (middle), 0.99 (slow) and 0.997 (very slow).Below is the result tabulated with the annealing rate and the iteration to obtain optimum distance.

Annealing Rate	# of iterations	Optimal Distance
0.9	255	339.0598276269681
0.92	534	302.5393179454663
0.95	943	222.782426426424
0.99	1342	185.85391225516986
0.997	1674	161.5361237293849