Markov Chain Monte Carlo

In the previous chapters we learnt how to simulate a r.v. based on Inverse transform and acceptance-rejection method. In many cases and practical applications the random vector to be simulated is composed of a large number of dependent variables ie $P(\vec{X}) = p(X_1, X_2 \ldots X_n)$ where $X_1, X_2, \ldots X_n$ are dependent r.v.s. In this case using the previously learnt techniques are highly inefficient. Also sometimes the probability distribution is known only upto a multiplicative constant ie. $P(\vec{X}) = C\, g(\vec{X})$. In such cases we use the techniques of Markov Chains to simulate a Markov chain whose equilibrium distribution is the required distribution to be sampled from.

Suppose we want to generate a random variable X having probability mass function, $P\{X=j\} = P_j$, $j = 1, \ldots N$ we construct an irreducible, aperiodic Markov chain with limiting probabilities P_j

Suppose our objective was to estimate $E(h(X))$

for some function ~~*~~ h. Now an important property of Markov chains is

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} h(X_i) = \sum_{j=1}^{N} p_j\, h(j)$$

$\therefore$ $\mathbb{E}(h(X)) = \sum_{j=1}^{N} h(j)\, p_j$ can be estimated by the estimator $\frac{1}{n}\sum_{i=1}^{n} h(X_i)$.

However since the early states of the Markov chain are influenced by the initial state (it takes a while to reach equilibrium) we can disregard the first k states for some suitable k ie we use

$\frac{1}{n-k} \sum_{i=k+1}^{n} h(X_i)$ as estimator for $\mathbb{E}(h(X))$

Metropolis-Hastings Algorithm

Suppose we want to simulate a random variable with probability mass function $\pi(j) = b(j)/B$ $j = 1, \cdots m$ where B is difficult to calculate.

One way is to construct a Markov chain whose limiting probabilities are $\pi(j)$. The Metro-polis Hastings algorithm provides an approach. It constructs a time reversible Markov chain in the following way.

Let Q be an arbitrary irreducible Markov transition probability matrix. The matrix Q with entries $q(i,j)$ is called the Proposal matrix. Now define the Markov chain $\{X_n, n \geq 0\}$ as follows. When $X_n = i$, generate a random variable X such that $P(X=j \mid X_n = i) = q(i,j)$. If $X=j$ then X_{n+1} is set to j with probability $\alpha(i,j)$ where α is a matrix called the Acceptance matrix. X_{n+1} is set to i with probability $1-\alpha(i,j)$.

P_{ij} is the probability of transitioning from state i to j is given by

$$P_{ij} = \begin{cases} q(i,j).\,\alpha(i,j) & \text{if } i \neq j \\ q(i,i) + \sum_{k \neq i} q(i,k)\left(1-\alpha(i,k)\right) & \end{cases}$$

How do we determine matrices Q and α? The proposal matrix is chosen to that it is easy and cheap to simulate it. The acceptance matrix α is chosen so that the Markov chain is time reversible

ie $\quad \pi(i) P_{i,j} = \pi(j) P_{j,i} \quad$ for $j \neq i$

which is equivalent to

$$\pi(i) q(i,j) \alpha(i,j) = \pi(j) q(j,i) \alpha(j,i)$$

~~It is~~ If we take

$$\alpha(i,j) = \min\left(1, \frac{\pi(j)}{\pi(i)} \cdot \frac{q(j,i)}{q(i,j)}\right)$$ these

equations will be satisfied.

$$\alpha(i,j) = \min\left(1, \frac{b(j)}{b(i)} \frac{q(j,i)}{q(i,j)}\right).$$

(Note knowledge of constant C is not required)

The Algorithm is as follows:

1. Choose an irreducible Markov transition probability matrix Q. Choose some integer value k between 1 and m
2. Let $n=0$ $\quad X_0 = k$
3. Generate a random variable X such that $P\{X=j\} = q(X_n, j)$. Generate a random number U.

4. If $U < \dfrac{b(X)\,q(X, X_n)}{b(X_n)\,q(X_n, X)}$ then $NS = X$

else $NS = X_n$

5. $n = n+1$, $X_n = NS$

6. Goto 3

Example

Suppose we want to generate a random element from a large complicated "combinatorial" set ℓ. For example ℓ might be the set of all permutations $(x_1, x_2, \ldots x_n)$ of numbers $(1, \ldots, n)$ such that $\sum_{j=1}^{n} j x_j > a$ for a given constant a. We use the Metropolis-Hastings algorithm as follows. To find the matrix Q we introduce a concept of neighboring elements. Two permutations are neighbours if one results from the other by an interchange of two positions. For eg: $(1, 2, 3, 4)$ and $(1, 2, 4, 3)$ are neighbours.

We define the transition probability function as follows.

$$q(s, t) = \frac{1}{|N(s)|} \quad \text{~~where~~ if } t \in N(s)$$

where $N(s)$ is the set of neighbours of s

4. If $u < \dfrac{b(X)\, q(X, X_n)}{b(X_n)\, q(X_n, X)}$ then $NS = X$

else $NS = X_n$

5. $n = n+1$, $X_n = NS$

6. Goto 3

Example

Suppose we want to generate a random element from a large complicated "combinatorial" set ℓ. For example ℓ might be the set of all permutations $(x_1, x_2, \ldots x_n)$ of numbers $(1, \ldots, n)$ such that $\sum_{j=1}^{n} j x_j > a$ for a given constant a. We use the Metropolis-Hastings algorithm as follows. To find the matrix Q we introduce a concept of neighboring elements. Two permutations are neighbours if one results from the other by an interchange of two positions. For eg. $(1, 2, 3, 4)$ and $(1, 2, 4, 3)$ are neighbours. We define the transition probability function as follows.

$$q(s, t) = \frac{1}{|N(s)|} \quad \text{~~where~~ if } t \in N(s)$$

where $N(s)$ is the set of neighbours of s

That is the target next state, is equally likely to be any of its neigbours.

The desired limiting probabilites are $\pi(s) = C$.

$$\alpha(s,t) = \min\left(\frac{|N(s)|}{|N(t)|}, 1\right)$$

If the present state of the Markov chain is s then one of its neighbours t is chosen _ say t. If t has fewer neighbours than s ie $\frac{|N(s)|}{|N(t)|} \geq 1$ then the next state is chosen to be t. Otherwise a random variable U is generated and the next state is chosen to be t if $U < \frac{|N(s)|}{|N(t)|}$ otherwise the next state remains s. The limiting probabilites are $\pi(s) = \frac{1}{|\mathcal{S}|}$

Given the set of $n!$ permutations suppose you want to estimate the probability $\alpha = P\left(\sum_{j=1}^{n} j x_j > a\right) = P(t(\vec{x}) > a)$

where $\vec{x} = (x_1, x_2, \dots, x_n)$ is a permutation of $1, \dots, n$ and $t(\vec{x}) = \sum_{j=1}^{n} j x_j$

One way of estimating α would be to generate a large number of random permutation and the proportion of permutation that satisfy $t(\vec{x}) > a$ would be an estimate for α. However if α is very small this procedure would be inefficient as it would require a large number of sample points. We can write

$$P(t(\vec{x}) > a) = \prod_{i=1}^{n} P(t(\vec{x}) > a_i \mid t(\vec{x}) > a_{i-1})$$

where $a_0 = 0, a_1, \dots, a_n = a$ are some intermediate points such that $P(t(\vec{x}) > a_i \mid t(\vec{x}) > a_{i-1})$ is not too small. The probability $P(t(\vec{x}) > a_i \mid t(\vec{x}) > a_{i-1})$ is estimated by the M-H algorithm in the following way:

1. Set $J = N = n = 0$

2. Choose an initial state $X_0 = \vec{x}$ s.t $t(\vec{x}) > a_{i-1}$

3. Given the current state X_n the proposed next state Y is chosen uniformly among the neighbours of X_n ie $P(Y = \vec{y}) = \dfrac{1}{|N(X_n)|}$

4. If $t(Y) \leq a_{i-1}$ goto 3

5. Generate U

   If $U \leq \dfrac{|N(X_n)|}{|N(Y)|}$

   $n = n+1$, $X_n = Y$, $N = N+1$

6. If $t(Y) > a_i$ $J = J+1$, Goto

The proportion $\dfrac{J}{N}$ gives the probability $P(t(\vec{x}) > a_i \mid t(\vec{x}) > a_{i-1})$

Gibbs Sampling

A variation of the Metropolis-Hastings algorithm is known as the Gibbś sampler. Let $X = (X_1, X_2, \ldots, X_n)$ be a random vector with probability mass function $p(\vec{x})$ known upto a multiplicative constant ie $p(\vec{x}) = C\, g(\vec{x})$ (where $g(\vec{x})$ is known but C is not).

In the Gibb's Sampler we assume that for any i and values x_j $j \neq i$ we can cheaply simulate the conditional distribution

$$P(X_i = x \mid X_j = x_j,\ j \neq i)$$

The Gibbs sampler operates by using the Metro-polis-Hastings algorithm on a Markov chain with transition probabilities as follows. When the present state is $\vec{x} = (x_1, x_2, \ldots, x_n)$ a coordinate that is equally likely to be any of $1, \ldots, n$ is chosen. If coordinate i is chosen then a random variable X whose mass function $P(X=x) = P(X_i = x \mid X_j = x_j, j \neq i)$ is ~~chosen~~ generated. If $X = x$ then the state $\vec{y} = (x_1, x_2, \ldots, x_{i-1}, x, x_{i+1}, \ldots x_n)$ is considered as candidate next state.

In other words

$$q(\vec{x}, \vec{y}) = \frac{1}{n} P\{X_i = x \mid X_j = x_j, j \neq i\}$$

ie $q(\vec{x}, \vec{y}) = \dfrac{P(\vec{y})}{n\, P\{X_j = x_j, j \neq i\}}$

Since we want the limiting mass funclun to be p, the vector $\vec{y}$ is accepted with probability

$$\alpha(\vec{x}, \vec{y}) = \min\left(\frac{P(\vec{y})}{P(\vec{x})} \cdot \frac{q(\vec{y}, \vec{x})}{q(\vec{x}, \vec{y})}, 1 \right)$$

$$= \min\left(\frac{P(\vec{y})}{P(\vec{x})} \cdot \frac{P(\vec{x})}{P(\vec{y})}, 1 \right)$$

$$= 1.$$

Example: Suppose we want to generate n random points on the ~~__~~ circle of radius 1 centered at the origin, conditiuned on the event that no two points are within distance d of each other.

$\beta = P\{$ no two points are within d of each other $\}$

This can be accomplished by employing the Gibbs Sampler by starting with n points in the circle $x_1, x_2, \ldots, x_n$

such that no two are within d of each other. Then, Generate a random number U Let $I = \text{Int}(nU)+1$. Also generate a random pt. on the circle. If this point is not within d of any of the other $n-1$ points excluding X_1 then replace X_1 with this pt. otherwise generate a new point. After a large number of iterations the set of n-points will have the desired distribution.

Ising Model

The Ising model is a mathematical model of magnetism. The model consists on spins which can be in one of two possible states. The spins are arranged on a lattice or graph and interact only with its nearest neighbours.

Let $G = (V, E)$ be a finite graph with vertex set V and edge set E. Each vertex may be in two states $+1$ or -1. A configuration is a vector $\theta = \{\theta_v : v \in V\}$ lying in the state space $\boldsymbol{\Theta} = \{-1, 1\}^V$.

A configuration is assigned the probability

$$\pi(\theta) = \frac{1}{Z} \exp\left\{ \sum_{v \sim w} \theta_v \theta_w \right\}$$ (Boltz-mann Dist)

where $v \sim w$ is the relation that v and w are neighbours.

$$Z = \sum_{\theta \in \boldsymbol{\Theta}} \exp\left\{ \sum_{v \sim w} \theta_v \theta_w \right\}$$

Calculation of probabilities like for $t, u \in V$ chance that t and u have the same state can be very difficult

$$\sum_{\theta : \theta_t = \theta_v} \pi(\theta) = \sum_{\theta} \frac{1}{2} (\theta_t \theta_u + 1) \pi(\theta)$$

We can use the Gibbs sampler to sample from $\pi(\theta)$

We construct a Markov chain X with stationary distribution π.

We start with a particular configuration. To proceed we restrict ourselves to transitions which flip the value of the current state only at one coordinate $v \in V$. That is given $X_n = i \; \{i_\omega : \omega \in V\}$ we decide that X_{n+1} takes values $j = (j_\omega : \omega \in V)$ such that $j_\omega = i_\omega$ whenever $\omega \neq v$. How do we select v? Either select uniformly or & cycle through elements of V in some predetermined manner.

Let $\Theta_{i,v} = \{ j \in \Theta : j_\omega = i_\omega \text{ for } \omega \neq v \}$

then $$q_{ij} = \frac{\pi_j}{\sum_{k \in \Theta_{i,v}} \pi_k} \quad 3$$

ie the next proposal state is chosen from $\Theta_{i,v}$ according to conditional distribution given the other components i_ω $v \neq \omega$.

Simulated Annealing

Let $\mathcal{A}$ be a set of finite vectors and let $V(\vec{x})$ be a nonnegative function defined on $\vec{x} \in \mathcal{A}$.

Suppose we are interested in finding the maximum (or minimum) value of $V(\vec{x})$ and the argument $\vec{x}^*$ which acheives this maximum.

Let $V^* = \max_{x \in \mathcal{A}} V(\vec{x})$

and $M = \{ x \in \mathcal{A} : V(\vec{x}) = V^* \}$

we define a measure (Gibb's Measure) on the set $\mathcal{A}$. Let $\lambda > 0$ we attribute the following mass function on the set of values of $\mathcal{A}$

$$P_\lambda(\vec{x}) = \frac{e^{\lambda V(\vec{x})}}{\sum_{x \in \mathcal{A}} e^{\lambda V(\vec{x})}}$$

Multiplying the numerator and denominator by $e^{-\lambda V^*}$ we see that

$$P_\lambda(\vec{x}) = \frac{e^{\lambda(V(\vec{x}) - V^*)}}{|M| + \sum_{x \notin M} e^{\lambda(V(\vec{x}) - V^*)}}$$

Now since $V(\vec{x}) - V^* < 0$ for $\vec{x} \notin M$

we obtain that as $\lambda \to \infty$

$$P_\lambda(\vec{x}) = \frac{\delta(\vec{x}, M)}{|M|} \quad \begin{cases} \delta(\vec{x}, M) = 1 & \text{if } x \in M \\ = 0 & \text{else} \end{cases}$$

~~Therefe~~ Therefore if we let λ to be large and generate a Markov chain whose limiting distribution is $P_\lambda(\vec{x})$ then most of the mass will be concentrated on elements in M.

Now if we have a neighbourhood structure on the elements of $\mathcal{A}$. we start the chain in some state $\vec{x}$ then we let the next proposed state $\vec{y}$ to be equally likely to be equally likely to be any of its neighbours, and if neighbour ~~state~~ $\vec{y}$ is chosen then the next state becomes $\vec{y}$ with probability

$$\min\left\{1, \frac{e^{\lambda V(\vec{y})} / N(\vec{y})}{e^{\lambda(\vec{x})} / N(\vec{x})}\right\}$$

If each vector has the same number of neighbours, then when the state is $\vec{x}$ if one of its neighbours say $\vec{y}$ is proposed then if $V(\vec{y}) \geq V(\vec{x})$ then the chain moves to state $\vec{y}$, and if $V(\vec{y}) < V(\vec{x})$ the chain moves to state $\vec{y}$ with probability $e^{\lambda(V(\vec{y}) - V(\vec{x}))}$ or remains in state $\vec{x}$ otherwise.

One weakness of this algorithm is that if the chain enters a state $\vec{x}$ whose value V is greater than each of its neighbours then since λ is large it might take a long time to move out of state $\vec{x}$ (since it moves out with prob $e^{\lambda(V(\vec{y}) - V(\vec{x}))}$)

A variation of the preceding algorithm called simulated annealing. In this in the n^{th} state of the Markov chain is $\vec{x}$, then a neighbour is randomly selected. If it is $\vec{y}$ then the next state is $\vec{y}$ with probability

$$\min\left\{1, \frac{\exp\{\lambda_n V(\vec{y})/N(\vec{y})\}}{\exp\{\lambda_n V(\vec{x})/N(\vec{x})\}}\right\}$$

where λ_n is a prescribed set of values that start out small and then grow. A useful choise of $\lambda_n = C \log(1+n)$ $C > 0$.

Application - Travelling Salesman Problem

In the travelling salesman problem we are given a list of cities and a starting city 0 $1, \ldots, r$. There is a non-negative reward $v(i,j)$ associated whenever the salesman goes from city i to city j. The salesman has to visit all the cities $1, \ldots, r$ is some permutation $x_1, \ldots, x_r$ such that $\sum_{i=1}^{r} v(x_{i-1}, x_i)$ is maximized. $(x_0 = 0)$

We define our set $\mathcal{A}$ to be the set of $r!$ permutations where a permutation $x_1, x_2, \ldots, x_r$ means the salesman visits the cities in order $0 \to x_1 \to x_2 \ldots \to x_r$

Our goal is to find elements of $\mathcal{A}$ s.t.

$$V(\vec{x}) = \cancel{\max} \sum_{i=1}^{r} v(x_{i-1}, x_i)$$ is maximized.

we use simulated annealing procedure The neighbourhood structure is defined as before (two permutations are neighbours if one can be obtained from the other by a single transposition). we take

$\lambda_n = \log(1+n)$.

~~we start~~ If the chain in a state $X_n = \vec{X}$ The proposed next state $\vec{y}$ is chosen uniformly among neighbours of $\vec{X}$. If $V(\vec{y}) \geq V(X_n)$ set $X_{n+1} = \vec{y}$ else set $X_{n+1} = \vec{y}$ with propability $(1+n)^{(V(\vec{y}) - V(X_n))}$ otherwise set it to X_n.