Statistics 572
Homework 7

Alejandro Robles

November 27th, 2018

1 Proportion of Defective Item using MCMC

Let parameter θ be the proportion of defective items. Let the prior be $\theta \sim Uniform(0,1)$. Also, let $x_1, ..., x_n \sim Bernoulli(\theta)$ with a Likelihood of $L(\theta; x) = \theta^{\sum x_i}(1-\theta)^{n-\sum x_i}$.

θ: Use a M-H sampler to generate Monte Carlo of size 2,000 whose invariant distribution is given by the posterior distribution of $\theta|x$.

1. Generate a sample of size 100 from $Bernoulli(0.2)$
2. Construct Monte Carlo and burn-in the first 500. Use the later 1500 Monte Carlo to calculate the mean and the variance of the chain. Plot the chain and histogram of the chain.
3. Given we know the posterior distribution is $\theta|x \sim Beta(\sum x_i + 1, n - \sum x_i + 1)$. Generate a Monte Carlo from $Beta(\alpha, \beta)$ using Random-Walk Metropolis Sampler and calculate the mean and variance of the chain. Plot chain and histogram.

Note for Step 2, randomly initiate θ_0 and take the candidate θ^* from prior $Uniform(0,1)$.

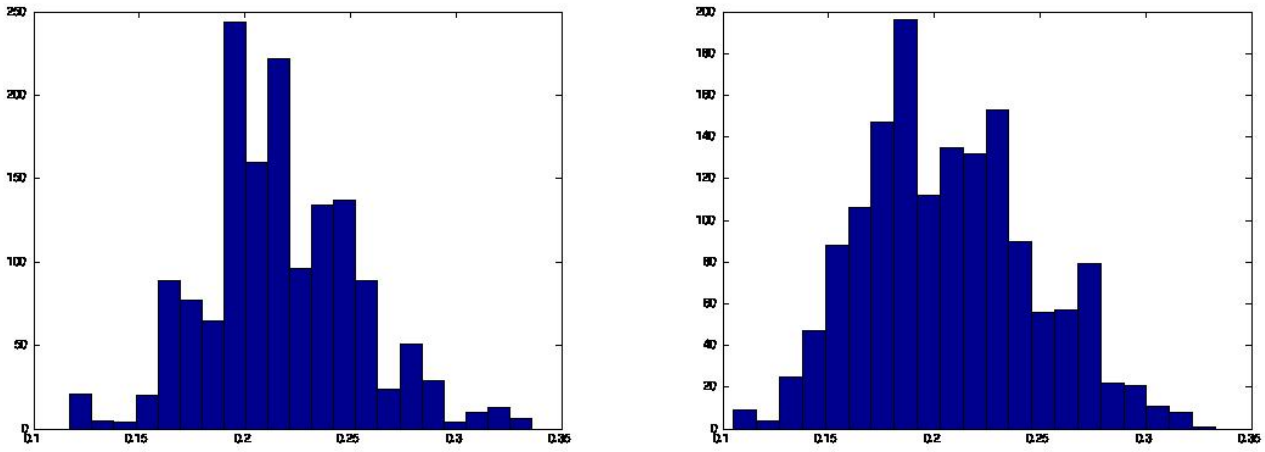
Calculate $\alpha = \frac{L(\theta^*;x)}{L(\theta_t;x)} = \frac{(\theta^*)^{\sum x_i}(1-\theta^*)^{n-\sum x_i}}{\theta_t^{\sum x_i}(1-\theta_t)^{n-\sum x_i}}$ and sample from $Uniform(0,1)$ for comparison.

MATLAB Code:

```
% ************* Part 1 *************
%  Parameters
n = 100; m = 2000; p = 0.2; burnin_rate = 0.25;

% Data Follows Binomial
data = binornd(1, p, n, 1);

% ************* Part 2 *************
% Generating posterior function
alpha = sum(data);
post = @(x) (x^alpha*(1-x)^(n-alpha));

% Proposal Distribution
generate_update = @(x) unifrnd(0,1);

% Monte Carlo
x = mcmc(m, generate_update, post, burnin_rate);
```

```
% Statistics
[mu1, variance1] = stats_mc(x);

% Plot Monte Carlo
plot_mc(x);

% ************* Part 3 *************
% Generating posterior function
alpha = sum(data) + 1;
beta = length(data) - sum(data) + 1;
post = @(x) (betapdf(x,alpha,beta));

% Proposal Distribution
generate_update = @(x) (x + (rand(1) - 0.5) / sqrt(12));

% Function to check if we need to update
update = @(new, delta, u) ~(new <= 0 || new >= 1 || u > delta);

% Monte Carlo
x = mcmc_general(m, generate_update, post, burnin_rate, update);

% Statistics
[mu2, variance2] = stats_mc(x);

% Plot Monte Carlo
plot_mc(x);
```

Result:

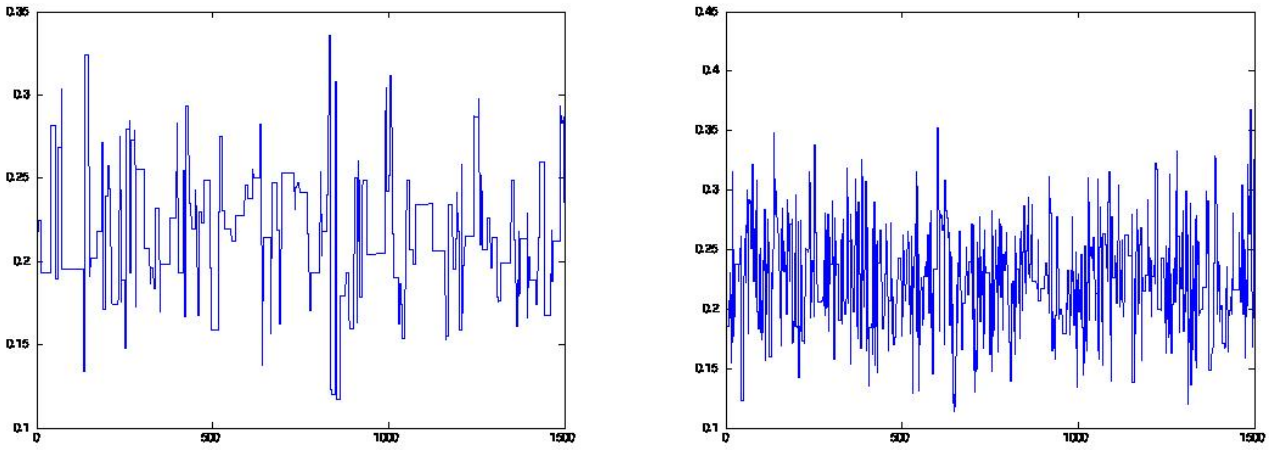


(a) MetropolisHastings

(b) Random Walk

Figure 1: Histograms

```
% From MCMC
Mean of MC: 0.2104
Variance of MC: 0.0015
% From Theoretical
Mean of MC: 0.2067
Variance of MC: 0.0016
```



(a) MetropolisHastings

(b) Random Walk

Figure 2: Markov Chains

Discussion:

As we can see from Figure 1, both follow very similar distributions for MetropolisHastings and Random Walk which peak around 0.2 which is the value of p. From Figure 2, we see that the Markov Chains are centered closely to p. The mean and variances are also very close to each other.

2 User Defined Functions

MCMC

```
function x = mcmc(m, generate_update, posterior, burnin)
% Create Space for Random Sample
x = zeros(1,m);
x(1) = randn(1); % Initiate first guess

for i=2:m
    y = generate_update(x(i-1));
    u = rand(1);
    delta = min([1 posterior(y)/posterior(x(i-1))]);
    if u <= delta
        x(i) = y;
    else
        x(i) = x(i-1);
    end
end

n = burnin * m;
x = x(n + 1 : m);
```

MCMC General

```
function x = mcmc_general(m, generate_update, posterior, burnin, bool_update)
% Create Space for Random Sample
x = zeros(1,m);
x(1) = randn(1); % Initiate first guess

for i=2:m
    y = generate_update(x(i-1));
    u = rand(1);
    delta = min([1 posterior(y)/posterior(x(i-1))]);
    if bool_update(y, delta, u);
        x(i) = y;
    else
        x(i) = x(i-1);
    end
end

n = burnin * m;
x = x(n + 1 : m);
```

Plot MCMC

```
function [] = plot_mc(x)
m = length(x);
h = 3.5 * std(x) * m^(-1/3);
t0 = min(x)-.1;
tm = max(x)+.1;
rng = tm-t0;
nbin = ceil(rng/h);
bins = t0:h:(nbin*h+t0);
vk = histc(x,bins);
fhat = vk/(m*h);
fhat(end) = [];
tm = max(bins);
bc = (t0+h/2):h:(tm-h/2);
figure
bar(bc,fhat,1)
figure
plot(x)
```

Statistics

```
function [mu, sigma_squared] = stats_mc(x)
mu = mean(x);
sigma_squared = var(x);
sprintf('Mean of MC: %g', mu)
sprintf('Variance of MC: %g', sigma_squared)
```