Comparative Analysis of Gibbs and Metropolis-Hastings Sampling for Bayesian Inference

Parker Walters

December 2024

1 Introduction

Metropolis-Hastings and Gibbs Sampling are two cornerstone algorithms in Markov Chain Monte Carlo (MCMC) methods, essential for tackling complex probabilistic models, particularly in Bayesian inference. These techniques enable sampling from high-dimensional and intractable probability distributions by constructing a Markov chain that converges to the target distribution. Widely used in Bayesian data analysis, these algorithms underpin inference tasks where direct sampling is infeasible, such as estimating posterior distributions for hierarchical models or integrating over multidimensional spaces.

Algorithm 1 Metropolis-Hastings Algorithm

Initialize $x \leftarrow$ initial value
for $i = 1$ to num_iterations **do**
  $x_{\text{proposed}} \leftarrow$ sample_from_q(x_{current})
  $\alpha \leftarrow \min\left(1, \frac{p(x_{\text{proposed}})q(x_{\text{current}}|x_{\text{proposed}})}{p(x_{\text{current}})q(x_{\text{proposed}}|x_{\text{current}})}\right)$
  $u \leftarrow$ uniform_random$(0, 1)$
  if $u < \alpha$ **then**
    $x_{\text{current}} \leftarrow x_{\text{proposed}}$
  end if
  samples$[i] \leftarrow x_{\text{current}}$
end for

The Metropolis-Hastings algorithm is a versatile method that proposes new samples from a user-defined proposal distribution, $q(x'|x)$, and accepts or rejects them based on an acceptance ratio. This acceptance probability is designed to ensure that the Markov chain converges to the target distribution over time, even when the proposal distribution differs significantly from the target. This flexibility makes Metropolis-Hastings applicable to a wide range of problems, though its efficiency can depend on the choice of the proposal distribution.

Algorithm 2 Gibbs Sampling Algorithm

Initialize $x_1, x_2, \ldots, x_d \leftarrow$ initial values
for $i = 1$ to num_iterations **do**
  for $j = 1$ to d **do**
    $x_j \leftarrow$ sample_from_conditional$(p(x_j|x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_d))$
  end for
  samples$[i, :] \leftarrow [x_1, x_2, \ldots, x_d]$
end for

In contrast, Gibbs Sampling is a special case of Metropolis-Hastings where all proposed samples are accepted because they are drawn directly from the conditional distributions of the target. By iteratively sampling each variable x_j from its conditional distribution given the others, $p(x_j|x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_d)$, Gibbs Sampling simplifies the sampling process and ensures automatic acceptance. While Gibbs Sampling can be more efficient when conditional distributions are easy to compute, it is limited to problems where these distributions are readily available. Together, these methods provide complementary tools for probabilistic modeling and simulation, addressing a wide array of challenges in computational statistics.

2 Methods

2.1 Bayesian Inference Simulation Using Gibbs Sampling and Metropolis-Hastings

The first wave of tests involves a Bayesian inference simulation conducted using both Gibbs Sampling and Metropolis-Hastings methods. The simulation focuses on estimating the posterior distribution for a normal distribution with an unknown mean, μ. The likelihood function is a normal distribution, $\mathcal{N}(\mu \mid 5, 1)$, and the prior for μ is defined as a uniform distribution over the interval $[0, 10]$.

The analysis will involve running four parallel chains, each with 1,000 iterations and no burn-in, to compare the resulting distributions generated by the two methods. The performance of both algorithms will be analyzed using various statistical tests and visualization tools to evaluate convergence, efficiency, and accuracy. This comparison will highlight the strengths and weaknesses of each sampling approach in capturing the posterior distribution and provide insights into their practical use in Bayesian inference.

2.2 Refining Metropolis-Hastings

For the second wave of tests, Metropolis-Hastings will be fine-tuned to optimize its performance. While the base distribution remains the same, key parameters such as the number of iterations, burn-in period, and other algorithmic settings will be adjusted to improve efficiency and accuracy.

These modifications aim to address potential shortcomings observed in the initial tests and enhance the ability of Metropolis-Hastings to converge to the true posterior distribution. By iteratively refining the method, the goal is to make Metropolis-Hastings a more competitive approach in this Bayesian inference scenario, especially when compared to Gibbs Sampling. The results from this second wave will provide a deeper understanding of how tuning can affect the method's performance and applicability.

2.3 Theoretical Posterior Distribution

The target posterior distribution for the Bayesian inference simulation is shown in Figure 1. The likelihood function is a normal distribution, $\mathcal{N}(\mu \mid 5, 1)$, and the prior for μ is defined as a uniform distribution over the interval $[0, 10]$. As a result, the posterior distribution resembles a truncated normal distribution centered at $\mu = 5$. This theoretical distribution serves as the reference for evaluating the results of the Metropolis-Hastings and Gibbs Sampling algorithms.

Both methods aim to approximate this posterior distribution using sampling techniques. The comparison will involve assessing how well each algorithm reproduces the target distribution, analyzing convergence, and evaluating their efficiency in capturing the true underlying characteristics of the posterior.

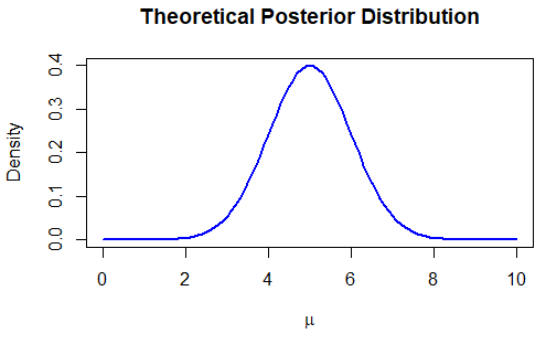

Figure 1: Theoretical Posterior Distribution. The posterior is a truncated normal distribution centered around $\mu = 5$.

3 Results

3.1 Gibbs

The Gibbs sampling process demonstrates strong performance across all key diagnostics. The RHAT value of 1.0049 is well below the convergence threshold of 1.05, confirming that the chains have mixed effectively and are sampling from the true posterior distribution. Furthermore, the effective sample size of 4000 highlights the sampler's ability to produce a large number of independent samples with minimal autocorrelation. These metrics collectively indicate a robust and reliable implementation of Gibbs sampling.

The density plot (Figure 2) shows that the sampled distributions closely align with the theoretical posterior, reflecting an excellent approximation. The trace plot (Figure 3) resembles a "fuzzy caterpillar," with no discernible trends or systematic patterns, confirming efficient exploration of the target space. Similarly, the autocorrelation plot (Figure 4) exhibits a sharp drop to near-zero values after a few lags, suggesting that samples are nearly independent. While minor deviations in the density plot tails and occasional extreme values in the trace plot could be explored further, these effects are minimal and do not detract significantly from the overall performance.

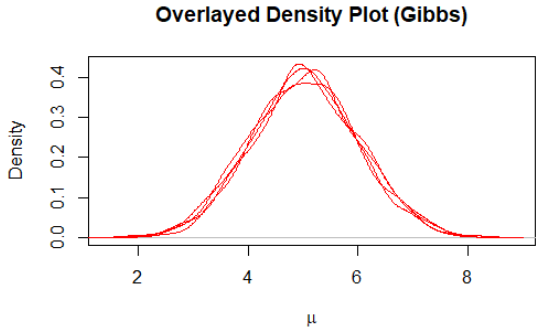

Figure 2: Overlayed Density Plot (Gibbs Sampling). The sampled distributions align closely with the theoretical posterior.

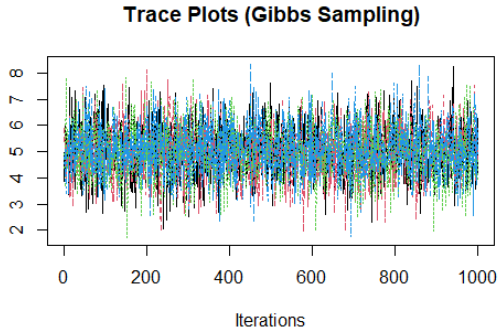

Figure 3: Trace Plots (Gibbs Sampling). The traces exhibit no discernible trends, confirming good mixing.

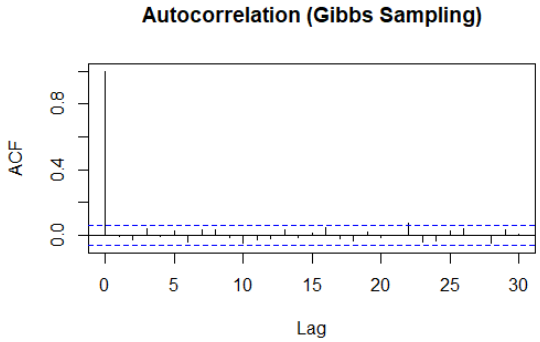

Figure 4: Autocorrelation Plot (Gibbs Sampling). The autocorrelation drops to near zero after a few lags, indicating nearly independent samples.

3.2 Initial Metropolis-Hastings

The Metropolis-Hastings sampling process demonstrates several challenges in terms of performance and diagnostics. The RHAT value of 1.059021 is above the ideal threshold of 1.05, indicating poor convergence across the chains. Additionally, the effective sample size is only 175, which suggests significant autocorrelation and inefficiency in producing independent samples. These issues highlight that the chains have not mixed well, limiting the reliability of the posterior estimates. There was also an acceptance rate of 0.844 which in terms of running Metropolis-Hastings simulations is rather high as the goal is to be within 0.2 to 0.5 as that satisfies a better filling rate for accurate acceptance.

The density plot (Figure 5) shows notable deviations from the theoretical posterior, with inconsistencies in the shape across chains. The trace plot (Figure 6) shows some discernible trends and clustering, which indicates poor mixing and suggests that the chains may be stuck in local regions of the posterior space. The autocorrelation plot (Figure 7) demonstrates a slow decay of autocorrelation across lags, reflecting that samples are not independent and that many iterations may be required to achieve a representative posterior. These observations collectively suggest that the Metropolis-Hastings sampler might benefit from tuning the proposal distribution to improve efficiency and convergence.

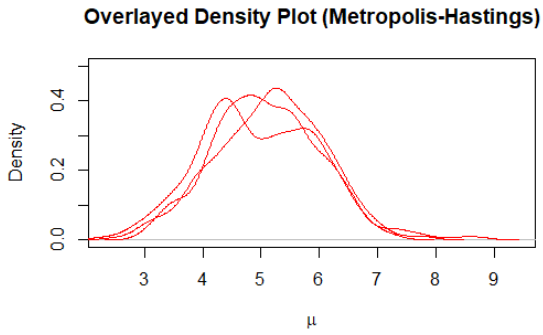

Figure 5: Overlayed Density Plot (Metropolis-Hastings). Deviations from the theoretical posterior indicate poor chain mixing.

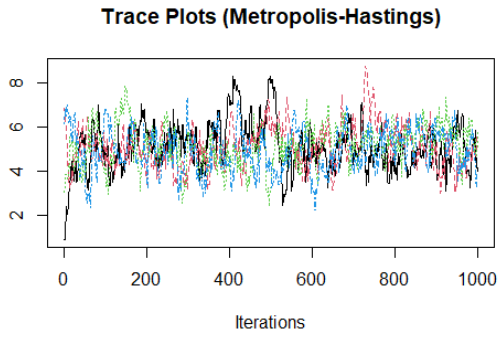

Figure 6: Trace Plots (Metropolis-Hastings). Discernible trends and clustering indicate poor mixing.

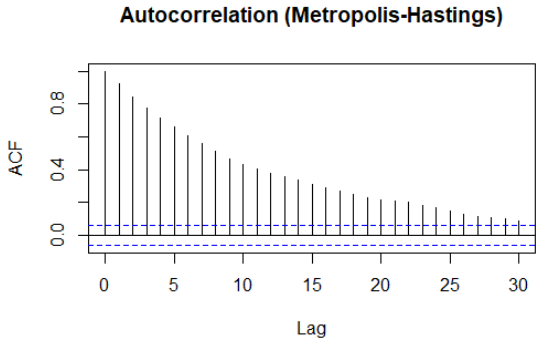

Figure 7: Autocorrelation Plot (Metropolis-Hastings). Slow decay indicates significant dependency between samples.

3.3 Optimized Metropolis-Hastings

The Optimized Metropolis-Hastings sampling process demonstrates significant improvements in performance and diagnostics compared to the initial implementation. The RHAT value of 1.001 is well within the convergence threshold of 1.05, indicating that the chains have mixed effectively. Additionally, the effective sample size of 2000 is a substantial improvement, highlighting the increased independence of the samples and the sampler's efficiency in approximating the posterior distribution. There was also an acceptance ratio of 0.7, which is an improvement to the prior test. Even though it still is higher than preferred it can partially be explained due to the simplicity of the proposal distribution being given.

The improvements were achieved by introducing a 1000-iteration burn-in, increasing the proposal standard deviation from 0.5 to 1, and extending the number of iterations per chain to 5000 instead of the original 1000. These adjustments ensured better exploration of the target space and reduced autocorrelation between samples. The density plot (Figure 8) closely aligns with the theoretical posterior, indicating robust sampling. The trace plot (Figure 9) resembles the desired "fuzzy caterpillar" with no discernible trends or clustering, confirming effective mixing. The autocorrelation plot (Figure 10) shows a rapid drop to near-zero values after a few lags, demonstrating the independence of the samples. Overall, the optimized Metropolis-Hastings sampler achieves results comparable to Gibbs sampling.

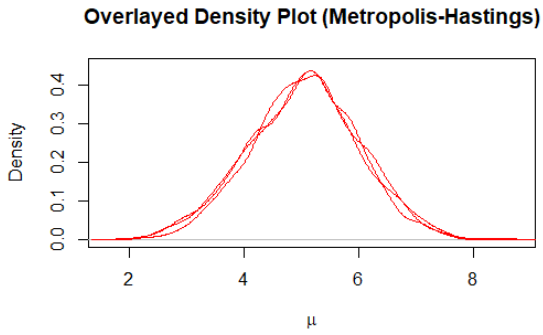

Figure 8: Overlayed Density Plot (Optimized Metropolis-Hastings). The sampled distributions align closely with the theoretical posterior.

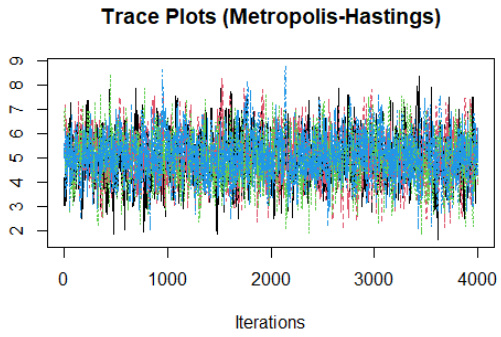

Figure 9: Trace Plots (Optimized Metropolis-Hastings). The traces exhibit no discernible trends, confirming good mixing.

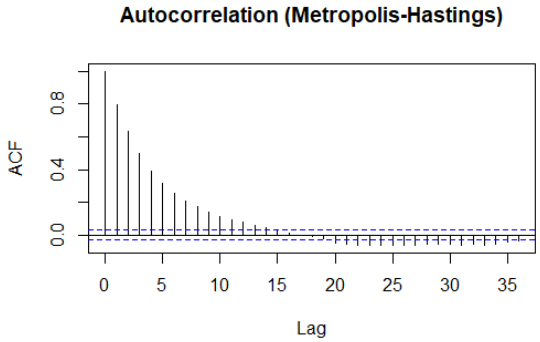

Figure 10: Autocorrelation Plot (Optimized Metropolis-Hastings). The autocorrelation drops to near zero after a few lags, indicating nearly independent samples.

4 Conclusion

In this study, Gibbs sampling demonstrated superior efficiency and accuracy for the given problem. The RHAT values and effective sample sizes confirmed its ability to converge quickly with minimal autocorrelation, yielding results that closely matched the theoretical posterior. However, Gibbs sampling's reliance on tractable conditional distributions limits its applicability to cases where such distributions can be derived. For problems involving complex or non-conjugate priors—such as a Cauchy prior for this Binomial likelihood— Gibbs becomes impractical or unusable due to its inability to directly sample from the required conditional distributions. As a result, while Gibbs is an excellent tool for problems that meet these requirements, its inherent limitations make it less flexible for more general cases.

Metropolis-Hastings, on the other hand, proved to be more challenging to optimize for this specific test case. The initial implementation exhibited slow convergence, poor mixing, and significant autocorrelation, requiring additional tuning, such as adjusting the proposal distribution, adding a burn-in period, and increasing the number of iterations. Even after these optimizations, its performance lagged behind Gibbs in terms of efficiency. However, the flexibility of Metropolis-Hastings—its ability to handle non-tractable or complex conditional distributions—makes it invaluable in scenarios where Gibbs sampling fails. Thus, while Gibbs was the more effective approach here, Metropolis-Hastings remains a powerful and adaptable alternative for more complex problems requiring greater sampling flexibility.

A Appendix

A.1 Gibbs Code

```
# Gibbs Sampling
library(coda) # Used for MCMC Diagnostics

n_iter <- 1000   # Number of iterations per trial
x <- 5           # Observed data
sigma2 <- 1      # Variance
mu_prior_min <- 0
mu_prior_max <- 10
n_chains <- 4
chains_gibbs <- list()  # To store all chains

for (c in 1:n_chains) {
  mu_samples <- numeric(n_iter)
  mu <- runif(1, mu_prior_min, mu_prior_max)  # Random initialization

  for (i in 1:n_iter) {
    mu <- rnorm(1, mean = x, sd = sqrt(sigma2))
    mu <- max(mu_prior_min, min(mu_prior_max, mu))  # Clip to respect Uniform prior bounds
    mu_samples[i] <- mu
  }

  chains_gibbs[[c]] <- mu_samples  # Store the chain
}

# Convert to MCMC list for diagnostics
gibbs_mcmc <- mcmc.list(lapply(chains_gibbs, mcmc))
```

A.2 Initial Metropolis-Hastings Code

```
# Metropolis-Hastings
library(coda) # Used for MCMC Diagnostics

n_iter <- 1000   # Number of iterations per trial
x <- 5           # Observed data
sigma2 <- 1      # Variance
mu_prior_min <- 0
mu_prior_max <- 10
proposal_sd <- 0.5
n_chains <- 4
chains_mh <- list()  # To store all chains
acceptance_rates <- numeric(n_chains)  # Store acceptance rates for each chain

for (c in 1:n_chains) {
  mu_samples <- numeric(n_iter)
  mu <- runif(1, mu_prior_min, mu_prior_max)  # Random initialization
  accepted <- 0

  for (i in 1:n_iter) {
    mu_proposed <- rnorm(1, mean = mu, sd = proposal_sd)
    acceptance_ratio <- posterior(mu_proposed, x, sigma2) / posterior(mu, x, sigma2)

    if (runif(1) < acceptance_ratio) {
      mu <- mu_proposed
      accepted <- accepted + 1
    }

    mu_samples[i] <- mu
  }
```

```
  acceptance_rates[c] <- accepted / n_iter  # Store acceptance rate for this chain
  chains_mh[[c]] <- mu_samples  # Store the chain
}

print(mean(acceptance_rates))

# Convert to MCMC list for diagnostics
mh_mcmc <- mcmc.list(lapply(chains_mh, mcmc))
```

A.3 Optimized Metropolis-Hastings Code

```
# Metropolis-Hastings Sampling with Burn-in
library(coda) # Used for MCMC Diagnostics

n_chains <- 4
n_burn_in <- 1000
n_iter <- 5000  # Total iterations, including burn-in
proposal_sd <- 1
chains_mh <- list()  # To store all chains

for (c in 1:n_chains) {
  mu_samples <- numeric(n_iter)
  mu <- runif(1, mu_prior_min, mu_prior_max)  # Random initialization
  accepted <- 0

  for (i in 1:n_iter) {
    mu_proposed <- rnorm(1, mean = mu, sd = proposal_sd)
    acceptance_ratio <- posterior(mu_proposed, x, sigma2) / posterior(mu, x, sigma2)

    if (runif(1) < acceptance_ratio) {
      mu <- mu_proposed
      accepted <- accepted + 1
    }

    mu_samples[i] <- mu
  }

  # Discard the burn-in samples
  acceptance_rates[c] <- accepted / n_iter  # Store acceptance rate for this chain
  chains_mh[[c]] <- mu_samples[(n_burn_in + 1):n_iter]
}

print(mean(acceptance_rates))

# Convert to MCMC list for diagnostics
mh_mcmc <- mcmc.list(lapply(chains_mh, mcmc))
```

A.4 Diagnostics Test Code

```
library(coda) # Used for MCMC Diagnostics

# Gibbs Sampling Trace Plots
traceplot(gibbs_mcmc, main = "Trace␣Plots␣(Gibbs␣Sampling)")

# Metropolis-Hastings Trace Plots
traceplot(mh_mcmc, main = "Trace␣Plots␣(Metropolis-Hastings)")

# Overlayed Density Plots
plot(density(chains_gibbs[[1]]), col = "red", main = "Overlayed␣Density␣Plot
␣␣␣␣(Gibbs)", xlab = expression(mu))
```

```
for (c in 2:n_chains) {
  lines(density(chains_gibbs[[c]]), col = "red")
}

plot(density(chains_mh[[2]]), col = "red", main = "Overlayed␣Density␣Plot
␣␣␣␣(Metropolis-Hastings)", xlab = expression(mu), ylim = c(0, 0.45))
for (c in 2:n_chains) {
  lines(density(chains_mh[[c]]), col = "red")
}

# Autocorrelation for Gibbs Sampling (First Chain)
acf(chains_gibbs[[1]], main = "Autocorrelation␣(Gibbs␣Sampling)")

# Autocorrelation for Metropolis-Hastings (First Chain)
acf(chains_mh[[1]], main = "Autocorrelation␣(Metropolis-Hastings)")

# R-hat Diagnostics
cat("R-hat␣for␣Gibbs␣Sampling:\n")
print(gelman.diag(gibbs_mcmc)$psrf)

cat("R-hat␣for␣Metropolis-Hastings:\n")
print(gelman.diag(mh_mcmc)$psrf)

# Effective Sample Size
cat("Effective␣Sample␣Size␣(Gibbs␣Sampling):\n")
print(effectiveSize(gibbs_mcmc))

cat("Effective␣Sample␣Size␣(Metropolis-Hastings):\n")
print(effectiveSize(mh_mcmc))
```