Reversible Jump Markov Chain Algorithm for Model selection in Linear Regression

Younghwan Cho

May 2024

Abstract

Model selection in linear regression is a crucial task, especially when dealing with a large number of potential predictor variables. The challenge lies in identifying the optimal subset of predictors that best explains the response variable, while also determining the appropriate number of predictors to include in the model. In this study, we explore the application of the Reversible Jump Markov Chain Monte Carlo (RJMCMC) algorithm as a Bayesian approach to model selection in linear regression. RJMCMC extends traditional MCMC methods by allowing the model to jump between parameter spaces of different dimensions, thereby facilitating the selection of both the optimal subset of predictors and the correct model dimension. Through a series of simulations, we evaluate the effectiveness of RJMCMC in various scenarios, including different feature space sizes and varying levels of noise. Our findings demonstrate that while RJMCMC is robust in small to moderately sized feature spaces, its performance can degrade as the feature space expands or as parameter variability increases. The results also suggest that the initial values in the algorithm are less critical than managing parameter variability, which has a significant impact on the algorithm's accuracy and convergence.

1. Introduction

1.1. Model Selection and variable selection in regression

Model selection in general refers to a procedure of choosing the most appropriate statistical or machine learning model for a given dataset. The aim is to find a model that best captures the relationship between the independent variables (predictors) and the dependent variable (response) without overfitting or underfitting. When it comes to regression, model selection can be selecting between linear regression, polynomial regression, ridge regression, lasso regression, or more complex algorithms like decision trees, support vector machines, or neural networks depending on the data structure and problem setup.

In this research we will confine our interest to linear regression setup. Traditional linear regression formulation is as follows: Given a data set $\{y_i, x_{i1}, \ldots, x_{ip}\}_{i=1}^n$ of n statistical units, a linear regression model assumes that the relationship between the dependent variable y and the vector of regressors $\mathbf{x}$ is linear. This relationship is modeled through a disturbance term or error variable ε - an unobserved random variable that adds "noise" to the linear relationship between the dependent variable and regressors. Thus the model takes the form

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^\top \boldsymbol{\beta} + \varepsilon_i, \quad i = 1, \ldots, n,$$

where $^\top$ denotes the transpose, so that $\mathbf{x}_i^\top \boldsymbol{\beta}$ is the inner product between vectors $\mathbf{x}_i$ and $\boldsymbol{\beta}$.

Often these n equations are stacked together and written in matrix notation as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \quad \text{where} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}, \quad \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

Variable selection, also known as feature selection, involves selecting the most relevant predictors for use in a model. In linear regression setup as above, variable selection refers to selecting a subset of size m ($< p$) of $\{1, \ldots, p\}$, where the subset is called relevant features. With reordering of m if needed, updated formulation is:

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_m x_{im} + \varepsilon_i = \mathbf{x}_i^\top \boldsymbol{\beta} + \varepsilon_i, \quad i = 1, \ldots, n$$

1.2. Bayesian and non-Bayesian Approaches

Bayesian approach in statistics, as opposed to non-Bayesian (frequentist) approach, imposes probability distribution on the parameter. This is called prior distribution. If we denote the prior distribution by $\pi(\theta)$ and the sampling distribution by $f(\mathbf{x} \mid \theta)$, then the posterior distribution, the conditional distribution of θ given the sample, $\mathbf{x}$, is

$$\pi(\theta \mid \mathbf{x}) = f(\mathbf{x} \mid \theta)\pi(\theta)/m(\mathbf{x})$$

The strength of Bayesian approach is its ability to sequentially update the beliefs about the parameters as new data become available. This is done by treating the posterior probabilities obtained from previous data as the new priors. Many algorithms have been developed on this Bayesian setting, among which MCMC (Markov Chain Monte Carlo) is the most popular method. MCMC includes algorithms such as Metropolis-Hastings, Gibbs Sampling, Kalman Filtering and Sequential Monte Carlo.

1.3. Metropolis-Hastings Algorithm

Metropolis-Hastings Algorithm is one of the most popularM MCMC technique used to approximate sampling from complicated distributions. The proposed RJMCMC, which will be explained later, is a specific application of and an extension to Metropolis-Hastings Algorithm, so I will introduce specifics of this algorithm here.

The setup is: we are interested in generating samples of a random variable X distributed according to the density $f(x)$. In addition to $f(x)$, we will assume having a density $q(y \mid x)$ that satisfies the following properties: 1. It is easy to sample from $q(\cdot \mid x)$ for all x. 2. The support of q contains the support of $f(x)$. 3. The functional form of $q(y \mid x)$ is known or $q(y \mid x)$ is symmetric in y and x. It is not necessary to know the normalizing constant in $q(y \mid x)$ as long as it does not depend upon x.

Given $f(x)$ and a choice of $q(y \mid x)$, that satisfies the above mentioned properties, the Metropolis-Hastings algorithm can be stated as follows:
$q(y \mid x)$ is called the Choose an initial condition X_0 in the support of $f(x)$. The Markov chain $X_1, X_2, \ldots, X_n$ is constructed iteratively according to the steps: 1. Generate a candidate $Y \sim q(y \mid X_t)$. 2. Update the state to X_{t+1} accroding to:

$$X_{t+1} = \begin{cases} Y & \text{with probability } \rho(X_t, Y) \\ X_t & \text{with probability } 1 - \rho(X_t, Y) \end{cases},$$

where $\rho(x,y) = \min\left\{\frac{f(y)q(x|y)}{f(x)q(y|x)}, 1\right\}$. proposal density and $\rho(x,y)$ is called the acceptance-rejection function.

Algorithm 1 Metropolis-Hastings Algorithm

Input: Proposal density q, target density f
Output: Sequence of samples x_t that approximate f
Initialize x_0
for $t = 1$ to n **do**
  Sample $u \sim \text{Uniform}(0,1)$
  Sample $Y \sim q(Y|x_t)$
  if $u < \min\left(1, \frac{f(Y)q(x_t|Y)}{f(x_t)q(Y|x_t)}\right)$ **then**
    $x_{t+1} \leftarrow Y$
  else
    $x_{t+1} \leftarrow x_t$
  end if
end for

1.3.1. Independent Metropolis Hastings

The Independent Metropolis Hastings chains are created by taking the proposal density of the $q(y|X) = q(y)$.This shows that the proposal density needs memoryless property such as exponential distribution. So, at this case the Metropolis Hastings algorithm will be more simpler than the ordinary MH algorithm.

Algorithm 2 Independent Metropolis-Hastings Algorithm

Input: Proposal density q, target density f
Output: Sequence of samples x_t that approximate f
Initialize x_0
for $t = 1$ to n **do**
  Sample $u \sim \text{Uniform}(0,1)$
  Sample $Y \sim q(Y)$
  if $u < \min\left(1, \frac{f(Y)q(x_t)}{f(x_t)q(Y)}\right)$ **then**
    $x_{t+1} \leftarrow Y$
  else
    $x_{t+1} \leftarrow x_t$
  end if
end for

1.3.2. Random walk Metropolis Hastings

The Random walkMetropolisHastings chains are created by taking the proposal density of the $q(y-x) = q(x|y)$. This shows that the proposal density is symmetric in certain location parameter, and normal distribution is one of case. So, at this case the Metropolis Hastings algorithm will be more simpler than the ordinary MH algorithm.

Algorithm 3 Random Walk Metropolis-Hastings Algorithm

Input: Proposal density q, target density f
Output: Sequence of samples x_t that approximate f
Initialize x_0
for $t = 1$ to n **do**
  Sample $u \sim \text{Uniform}(0, 1)$
  Sample $Y \sim q(Y - x_t)$
  if $u < \min\left(1, \frac{f(Y)}{f(x_t)}\right)$ **then**
    $x_{t+1} \leftarrow Y$
  else
    $x_{t+1} \leftarrow x_t$
  end if
end for

2. Approach or Methodology

2.1 Goal

We are interested in solving for the regression coeffcients in a standard linear model but the selection of predictors is not known. In other words, we are given a large number, say m, of predictors and we have to select an appropriate subset to obtain the optimal model. This is a model/variable selection problem which was explained in introduction. We will restrict to a smaller problem where the optimal subset is simply the first n predictors, we just don't know what n is.

2.2. Problem Specification

We seek coefficients for the model

$$y = \sum_{i=1}^{n} x_i b_i + \epsilon$$

where $n < m, x_i$s are the predictors, y is the response variable, and ϵ is the measurement noise. We are given k independent measurements, denoted in bold by $\mathbf{y}, \mathbf{X}$ and ϵ. We will seek a Bayesian solution to the joint estimation of $\{n, b_1, \ldots, b_n\}$.
To setup a Bayesian formulation we need to define a joint posterior density of the type:

$$f\left(n, \mathbf{b}_n \mid \mathbf{y}\right) \propto f\left(\mathbf{y} \mid n, \mathbf{b}_n\right) f\left(\mathbf{b}_n \mid n\right) f(n).$$

We will use the notation $\mathbf{X}_n = \mathbf{X}(:, 1:n)$ and $\mathbf{b}_n = \{b_1, \ldots, b_n\}$. We will use the following terms:

- The likelihood function is given by: $f\left(\mathbf{y} \mid n, \mathbf{b}_n\right) = \left(\frac{1}{\sqrt{2\pi\sigma_0^2}}\right)^k e^{\frac{-1}{2\sigma_0^2}\|\mathbf{y}-\mathbf{X}_n\mathbf{b}_n\|^2}$.
- The prior on $\mathbf{b_n}$ given n is: $f\left(\mathbf{b}_n \mid n\right) = \left(\frac{1}{\sqrt{2\pi\sigma_p^2}}\right)^n e^{\frac{-1}{2\sigma_p^2}\|\mathbf{b}_n-\mu_b\|^2}$.
- The prior on n is simply uniform: $f(n) = \frac{1}{m}$.

2.3. Sampling from the Posterior with RJMCMC

Proposed algorithm to sample from the posterior using RJMCMC:
Let $(n, \mathbf{b}_n)$ be the current samples from the posterior.

(a) Select a candidate number n^* from the probability $f(n)$.

(b) If $n^* \geq n$, generate a random vector $\mathbf{u} \sim N(0, \sigma_r I_{n^*})$. The candidate coefficient vector is given by:

$$\mathbf{b}_{n^*} = \begin{bmatrix} \mathbf{b}_n \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix}$$

Compute the likelihoods:

$$h_1(\mathbf{u}) = \left(\frac{1}{\sqrt{2\pi\sigma_r^2}}\right)^{n^*} e^{\frac{-1}{2\sigma_r^2}\|\mathbf{u}\|^2}, \quad h_2(\mathbf{u}_1) = \left(\frac{1}{\sqrt{2\pi\sigma_r^2}}\right)^{n} e^{\frac{-1}{2\sigma_r^2}\|\mathbf{u}_1\|^2}.$$

(c) If $n^* < n$, generate a random vector $\mathbf{u}_1 \sim N(0, \sigma_r I_{n^*})$. The candidate coefficient vectc is given by:

$$\mathbf{b}_{n^*} = \mathbf{b}_n^1 + \mathbf{u}_1, \quad \mathbf{b}_n = \begin{bmatrix} \mathbf{b}_n^1 \\ \mathbf{b}_n^2 \end{bmatrix}$$

and form $\mathbf{u} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{b}_n^2 \end{bmatrix}$ Compute the likelihoods:

$$h_2(\mathbf{u}) = \left(\frac{1}{\sqrt{2\pi\sigma_r^2}}\right)^{n} e^{\frac{-1}{2\sigma_r^2}\|\mathbf{u}\|^2}, \quad h_1(\mathbf{u}_1) = \left(\frac{1}{\sqrt{2\pi\sigma_r^2}}\right)^{n^*} e^{\frac{-1}{2\sigma_r^2}\|\mathbf{u}_1\|^2}.$$

(d) Compute the acceptance-rejection function:

$$\rho = \min\left\{1, \frac{f(n^*, \mathbf{b}_{n^*} \mid \mathbf{y})\, h_2}{f(n, \mathbf{b}_n \mid \mathbf{y})\, h_1}\right\} = \min\left\{1, \frac{f(\mathbf{y} \mid n^*, \mathbf{b}_{n^*})\, f(\mathbf{b}_{n^*} \mid n^*)\, h_2}{f(\mathbf{y} \mid n^*, \mathbf{b}_n)\, f(\mathbf{b}_n \mid n)\, h_1}\right\}$$
$$= \min\left\{1, \frac{e^{-(E_1 - E_2)} \left(2\pi\sigma_p^2\right)^{(n-n^*)/2} e^{\frac{-1}{2\sigma_p^2}\left(\|\mathbf{b}_{n^*} - \mu_b\|^2 - \|\mathbf{b}_n - \mu_b\|^2\right)} h_2}{h_1}\right\}$$

where $E_1 = \frac{1}{2\sigma_0^2}\|\mathbf{y} - \mathbf{X}_{n^*}\mathbf{b}_{n^*}\|^2$ and $E_2 = \frac{1}{2\sigma_0^2}\|\mathbf{y} - \mathbf{X}_n\mathbf{b}_n\|^2$. (e) If $U \sim U[0,1]$ is less than ρ then set $(n, \mathbf{b}_n) = (n^*, \mathbf{b}_{n^*})$. Else, return to Step (a).

2.4. Simulation with RJMCMC and output

2.4.1 Experiment

To verify the accuracy of the algorithm, we will do experiments on 10 different setups. In setup 1-5, we will impose different initial value and true value. In setup 6-8, we will impose different dimension of feature space. In setup 9 and 10, we will impose different variances.

$$\begin{array}{|c|}\hline m = 10 \\ n_0 = \text{ceil}(\text{rand} \cdot m); \\ k = 10 \\ \sigma_0 = 0.2 \\ \sigma_p = 0.3 \\ \mu_b = 2 \cdot \text{ones}(n_0, 1); \\ b = \mu_b + \sigma_p \cdot \text{randn}(n_0, 1); \\ X = 5 \cdot \text{randn}(k, m); \\ y = X(:, 1 : n_0) \cdot b + \sigma_0 \cdot \text{randn}(k, 1) \\ \hline \end{array}$$

Algorithm 4 Proposed Algorithm

Set a random integer n between 1 and m as the first candidate.
Construct the first coefficient β using the formula:

$$\beta = \mu_b + \sigma_p \begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix}^T \in \mathbb{R}^{n\times 1}$$

for iteration = 2 to maxiter **do**

Sample a random integer n^* between 1 and m, and let n be the element from the previous iteration.

if $n^* \geq n$ **then**

Generate a random vector $u \sim \mathcal{N}_{n^*}(0, \sigma^2 I_{n^*})$.
Construct the candidate coefficient vector β_{n^*}:

$$\beta_{n^*} = \begin{pmatrix} \beta_n \\ 0 \end{pmatrix} + \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \in \mathbb{R}^{n^*\times 1}, \quad u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

Compute the likelihood function:

$$h_1(u) = \left(\frac{1}{\sqrt{2\pi\sigma_r^2}}\right)^{n^*} \exp\left(-\frac{1}{2\sigma_r^2}\|u\|^2\right)$$

$$h_2(u) = \left(\frac{1}{\sqrt{2\pi\sigma_r^2}}\right)^{n} \exp\left(-\frac{1}{2\sigma_r^2}\|u_1\|^2\right)$$

else

Generate a random vector $u_1 \sim \mathcal{N}_{n^*}(0, \sigma^2 I_{n^*})$.
Construct the candidate coefficient vector β_{n^*} and likelihood function

end if

Compute the acceptance-rejection function:

$$\rho = \min\left\{1, \frac{f(n^*, \beta_{n^*}|y)h_2}{f(n, \beta_n|y)h_1}\right\} = \min\left\{1, \frac{f(y|n^*, \beta_{n^*})f(\beta_{n^*}|n^*)h_2}{f(y|n, \beta_n)f(\beta_n|n)h_1}\right\}$$

$$= \min\left\{1, \exp(E_1 - E_2)(2\pi\sigma_p^2)^{\frac{n-n^*}{2}} \exp\left(-\frac{1}{2\sigma_p^2}\left(\|\beta_{n^*} - \mu_\beta\|^2 - \|\beta_n - \mu_\beta\|^2\right)\right)\frac{h_2}{h_1}\right\}$$

Where:

$$E_1 = \frac{1}{2\sigma_0^2}\|y - X_{n^*}\beta_{n^*}\|^2, \quad E_2 = \frac{1}{2\sigma_0^2}\|y - X_n\beta_n\|^2$$

if $U \sim U(0,1)$ and $U \geq \rho$ **then**

Update to $(n, \beta_n) = (n^*, \beta_{n^*})$.

end if

If $U \geq \rho$, then preserve the (n, β_n)

end for

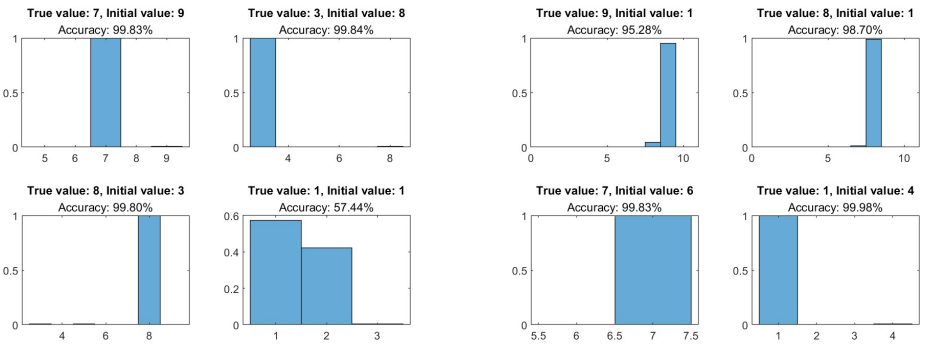
Setup 1: both true value and initial value are set to be a random numbers between 1 and 10.
Setup 2: target value is set to 1, initial value is set to be a random number between 1 and 10.
Setup 3: target value = initial value = ceil(rand · m)
Setup 4: target value = 10, initial value = 1
setup 5: target value = 1, initial value = 10
setup 6: we increase size of feature space m to 15, and both true value and initial value are set to be a random numbers between 1 and 15.
setup 7: we increase m to 50.
setup 8: we increase m to 100
setup 9: we increase σ_0 from 0.2 to 2, i.e. we increase the variability in y.
setup 10: we increase σ_p from 0.3 to 3, i.e. we increase the variability in parameter.

3. Experimental Results and conclusion

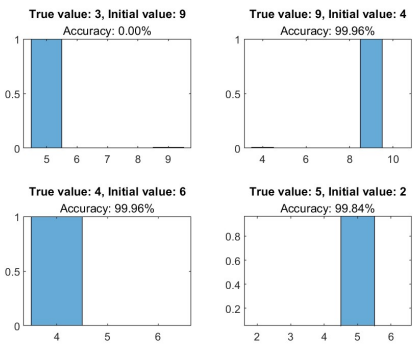
In this section, we will present results from the setups specified in section 2. Conclusions will be made after each results, collection of which will be our final conclusion.

3.1 Result for Setup 1

Setup 1: both true value and initial value are set to be a random numbers between 1 and 10. histogram of n values visited by the Markov chain:



(a) (b)

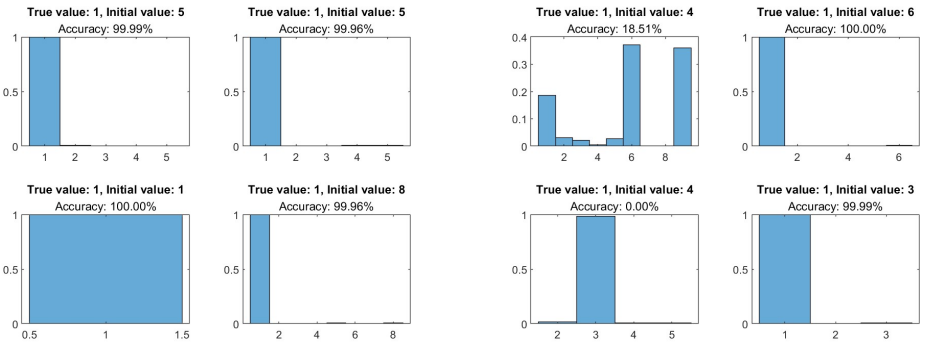


(c)

Conclusion: Overall accuracy (correct sampling percentage) is high for random initial value and random true value. However, we have case when true value=1 and initial value=1, the accuracy as 57%. Also, when true value=3 and initial value=2, n=5 with probability 1 which results in accuracy of 0. In this case, we can also note that converged n was between initial value and true value.

3.2 Result for Setup 2 and conclusion

Setup 2: target value is set to 1, initial value is set to be a random number between 1 and 10.
histogram of n values visited by the Markov chain:

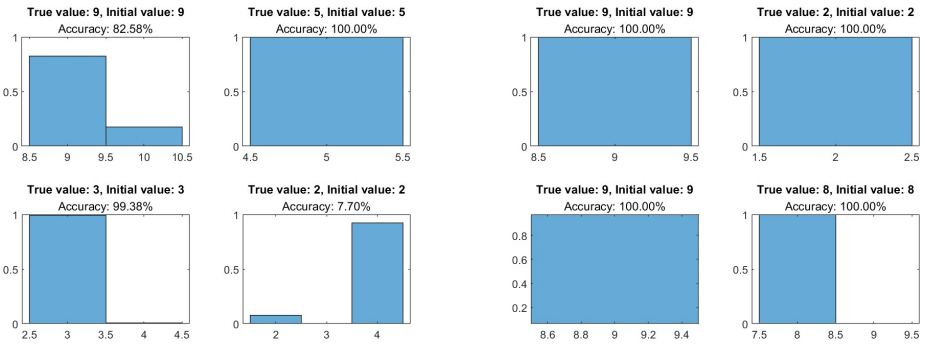


(d) (e)

Conclusion: Unlike what we have expected from setup1, there was a case where true value=1 and initial value=4 and converging n was not always between those two values.

3.3 Result for Setup 3

Setup 3: target value = initial value = ceil(rand · m)
histogram of n values visited by the Markov chain:

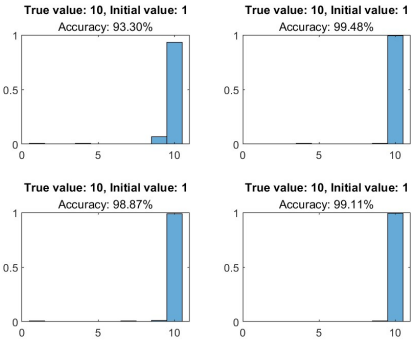


(f) (g)

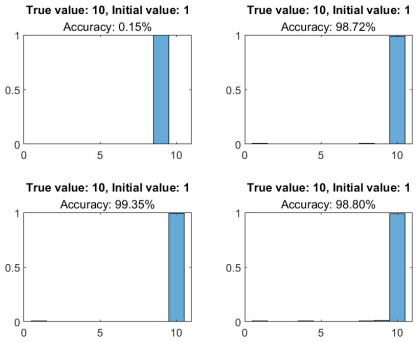
Conclusion: When initial value = true value, the algorithm showed high accuracy over all n. One simulation showed when true value = initial value = 2, n converged to 4 mostly and had accuracy 7.7%. However, in the other simulation in the same case showed 100% accuracy so that it is hard to deduce meanings out of this observation.

3.4 Result for Setup 4

Setup 4: target value = 10, initial value = 1
histogram of n values visited by the Markov chain:



(h)

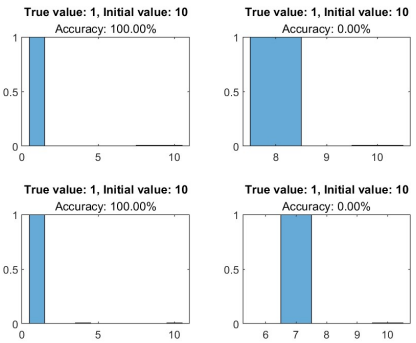


(i)

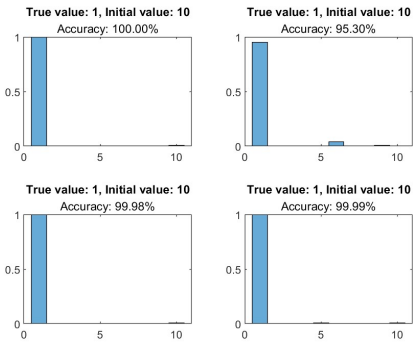
Conclusion: When initial value=1 and true value=10, most of the simulations ended up converging to true n. Even one simulation where accuracy was 0.15%, it converged to n=9 which is close to the true value 10. This implies distance of the initial value and true value is likely to be irrelevant when initial value $<$ true value.

3.5 Result for Setup 5

setup 5: target value = 1, initial value = 10
histogram of n values visited by the Markov chain:



(j)



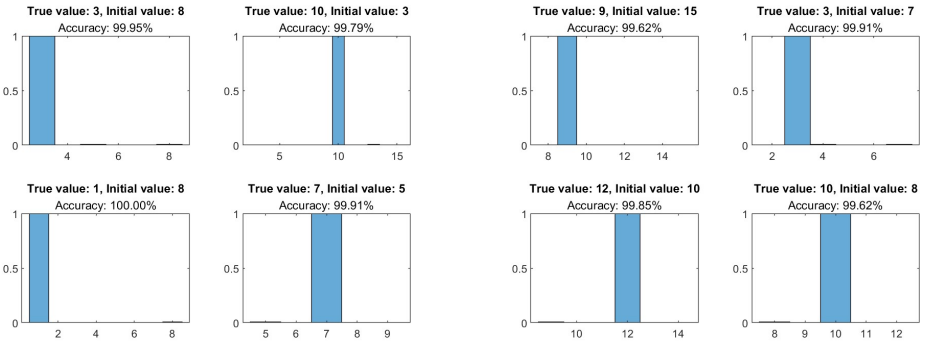
(k)

Conclusion: When initial value=10 and true value=1, accuracy was slightly worse than the opposite case as in setup 4. However this can be neglected if more simulations were run, as still accuracy in most cases were close to 100%. We might conclude that distance between initial value and true value is irrelevant to accuracy.

3.6 Result for Setup 6

setup 6: we increase size of feature space m to 15, and both true value and initial value are set to be a random numbers between 1 and 15.

histogram of n values visited by the Markov chain:

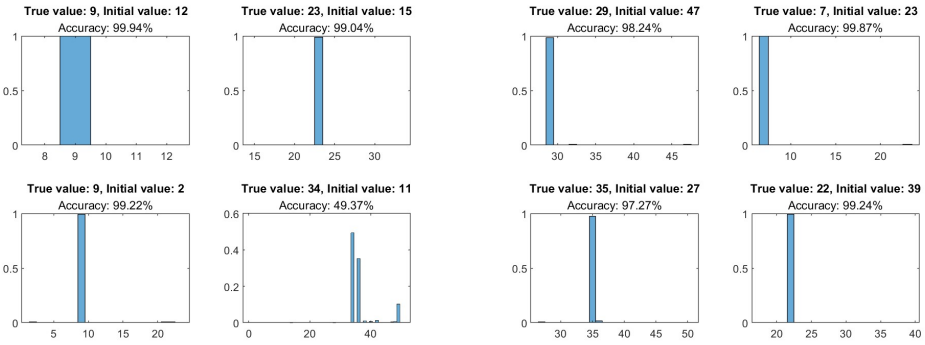


(l) (m)

Conclusion: When feature space m was increased to 15 from 10, the accuracy in simulation was not harmed. We might proceed to increase feature space even larger. This will be done ins setup 7 and 8.

3.7 Result for Setup 7

setup 7: we increase m to 50.
histogram of n values visited by the Markov chain:

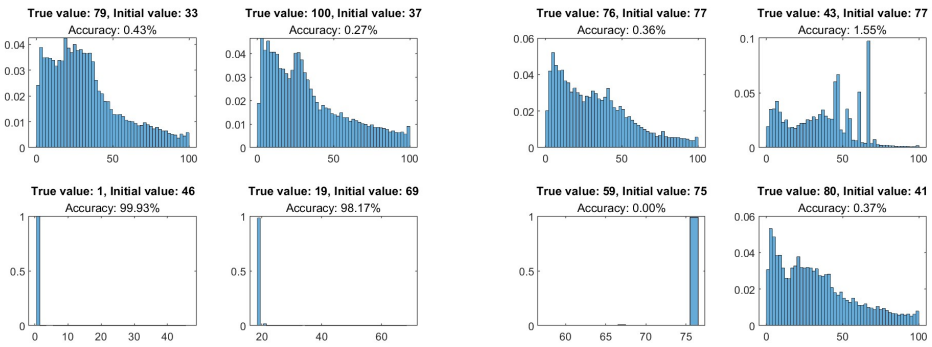


(n) (o)

Conclusion: When feature space m was increased to 50, the accuracy in simulation was slightly worse than the original case, as there are several 92-98 rages, 50% and 2% accuracies observed. However, given the high increase in feature space, this still needs further analysis.

3.8 Result for Setup 8

setup 8: we increase m to 100
histogram of n values visited by the Markov chain:

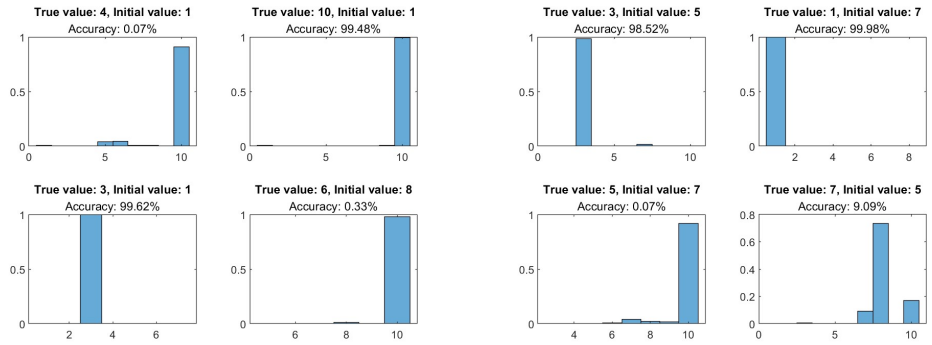


(p) (q)

Conclusion: As we increase the sample to 100, we can note that the distribution of n oftentimes follows left-skewed distribution. Also it is interesting to note that the distribution shows this pattern regardless of whether true value is small or large, and the procedure is likely to shrink the feature space. This might indicate the limit of the proposed algorithm. The algorithm will significantly fail after certain feature size.

3.9 Result for Setup 9

setup 9: we increase σ_0 from 0.2 to 2, i.e. we increase the variability in y.
histogram of n values visited by the Markov chain:

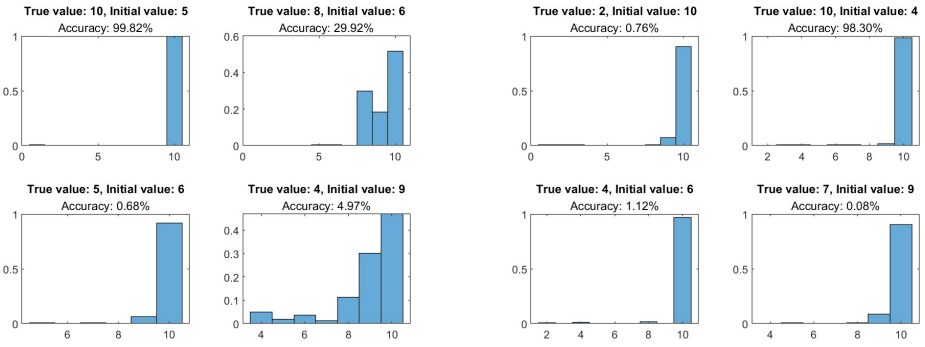


(r) (s)

Conclusion: After increasing σ_0 from 0.2 to 2.0, there was a considerable drop in accuracy compared to setup1. 4 out of 8 simulations completely missed true value, while others converged close to 100%.

3.10 Result for Setup 10

setup 10: we increase σ_p from 0.3 to 3, i.e. we increase the variability in parameter.
histogram of n values visited by the Markov chain:



(t) (u)

Conclusion: After increasing σ_p from 0.3 to 3, the accuracy was more terrible. Not only accuracy was low in most cases, the distribution of n showed right-skewed pattern regardless of true and initial value. We can conclude that regulating the variability in the parameter is more important than regulating variability in the likelihood.

4. Conclusion and Future Work

The Reversible Jump Markov Chain Monte Carlo (RJMCMC) algorithm, an extension of the Metropolis-Hastings algorithm, demonstrates its utility in variable selection for linear regression models, particularly in scenarios with a relatively small feature space. Throughout this project, we observed that RJMCMC performs robustly when the feature space is small to moderately sized, specifically when the number of features is less than 50. However, the algorithm's effectiveness begins to deteriorate as the feature space expands, with the distribution of selected features becoming increasingly predetermined and less accurate.

An interesting finding of this study is the minimal impact of initial values on the algorithm's performance, regardless of their proximity to the true value. This suggests that RJMCMC's convergence properties are relatively stable across different starting points. Nevertheless, the variability in parameters, particularly when increased, significantly hampers the algorithm's ability to accurately select the correct model. This limitation indicates that managing parameter variability is crucial for maintaining the accuracy and reliability of RJMCMC in model selection tasks.

Overall, while RJMCMC offers a valuable approach for model selection in linear regression, especially in smaller feature spaces, its limitations become evident as the complexity of the feature space and parameter variability increase. These insights highlight the importance of carefully considering the conditions under which RJMCMC is applied to ensure optimal performance.

References

[1] Casella, G., Berger, R. (2001). *Statistical Inference*. Duxbury Resource Center, June 2001.

[2] Chib, S., Greenberg, E. (1995). Understanding the metropolis-hastings algorithm. *The American Statistician*, 49(4), 327–335.

[3] Fan, Y., Sisson, S. A. (2011). Reversible jump mcmc. *Handbook of Markov Chain Monte Carlo*, 67–92.

[4] Liang, F., Liu, C., Carroll, R. (2010). Advanced markov chain monte carlo methods: Learning

from past samples. *Advanced Markov Chain Monte Carlo Methods: Learning from Past Samples*, July 2010.

[5] Metropolis, N., Ulam, S. (1949). The monte carlo method. *Journal of the American Statistical Association*, 44(247), 335–341.

[6] Rencher, A. C., Schaalje, G. B. (2008). *Linear Models in Statistics*. John Wiley Sons.

[7] Robert, C., Casella, G. (2011). A short history of markov chain monte carlo: Subjective recollections from incomplete data. *Statistical Science*, 26(1), February 2011.

[8] Weisberg, S. (2005). *Applied Linear Regression*, volume 528. John Wiley Sons.