Nuevo MCMC Machine Unlearning Algorithm

Viral Chitlangia

Abstract

Machine unlearning addresses the critical challenge of removing specific data points from trained models without full retraining—a necessity for privacy compliance and adversarial data mitigation. Current Markov Chain Monte Carlo (MCMC)-based unlearning methods suffer from slow mixing and computational inefficiency when approximating post-removal posteriors. We introduce Nuevo MCMC, a novel algorithm that integrates Newton updates and Metropolis-Hastings sampling to accelerate unlearning in generalized linear models. By pre computing Hessians and gradients from the original dataset, Nuevo constructs a Laplace-approximated proposal distribution, enabling rapid exploration of the updated posterior after data removal. This improvement in time efficiency gets samples from an approximate distribution. This paper tries to look into the trade off.

1 Introduction

Machine Learning models are generally trained on large sets of data. The data is generally trained on user data, which can become problematic if the user wishes to deny access to their data. This can also be used if we find that certain data points are "poisonous" to the data, and mistrain the model. Training the whole model again can take a lot of time and computational power, which can be avoided, especially since we have already trained on the whole data. This is where Machine Unlearning comes in, as instead of training the new model, we start from the whole model, and unlearn the data which needs to be removed. This significantly increases the time efficiency of the MCMC process for getting samples of the data.
In this report, I propose using a Newton update[Guo et al., 2023] on the weight to formulate an efficient algorithm for the Machine Unlearning Algorithm[Nguyen et al., 2022]. I will test out the algorithm on a Poisson Regression and Negative Binomial Regression Model, and compare with other MCMC Sampling algorithms.

2 Literature Review

[Nguyen et al., 2022] introduced an MCMC based framework to unlearn data on request keeping and changing the model parameters, as per new data. It proposed to unlearn the model based on what has been removed rather than learning the model again based on the remaining data points. Here, we have assumed that each observation is independent of the other observations($Y_i|X_i \perp\!\!\!\perp Y_j|X_j$, where $i \neq j$). We also assume $X_i \perp\!\!\!\perp Y_j$, where $i \neq j$. It proposed the distribution for the unlearned model as follows

X	Total Features
Y	Total Outputs
X_r	Remaining Features
Y_r	Remaining Outputs
X_e	Erased Features
Y_e	Erased Outputs
$L(w; D)$	Loss of Dataset D = (X, Y) with weight w

Table 1: Notation

$$h(\theta) = \log(f(Y|X,\theta)) + \log(f(\theta)) \tag{1a}$$
$$\log(f(Y|X,\theta)) = \log(f(Y_e|X_e,\theta)) + \log(f(Y_r|X_r,\theta)) \tag{1b}$$
$$\log(f(Y_r|X_r,\theta)) = \log(f(Y|X,\theta)) - \log(f(Y_e|X_e,\theta)) \tag{1c}$$
$$\log(f(Y_r|X_r,\theta)) + \log(f(\theta)) - \log(f(Y_r|X_r)) = h(\theta) - \log(f(Y_e|X_e,\theta)) - \log(f(Y_r|X_r)) \tag{1d}$$
$$\log(f(\theta|X_r,Y_r)) = h(\theta) - \log(f(Y_e|X_e,\theta)) - \log(f(Y_r|X_r)) \tag{1e}$$

Notation is as followed in Table 1.
[Guo et al., 2023] proposed a method to unlearn certain rows of data using an alteration of Newton's Method, where L is a pre defined Loss Function, and the model has weight w -

$$w^* = \arg\min_w L(w; X, Y) \tag{2a}$$
$$\Delta(w^*) = \nabla L(w^*; X_e, Y_e) \tag{2b}$$
$$H(w^*) = \nabla^2 L(w^*; X_e, Y_e) \tag{2c}$$
$$w^- = w^* + H(w^*)^{-1}\Delta(w^*) \tag{2d}$$

3 Proposed Algorithm for a General Model

For the proposed algorithm, we merge the distribution function proposed in [Nguyen et al., 2022], and propose a Transition Kernel based on the work in [Guo et al., 2023].

3.1 Distribution for Remaining Function

Let us say that the model is distributed with density function $f(Y|X,\beta)$. We can use this to get the distribution of the parameter β conditional on X_r and Y_r[Table 1] (C and E are constants):

$$f(\beta|X,Y) = \frac{f(Y|X,\beta)f(\beta)}{f(Y|X)}$$
$$f(\beta|X,Y) \propto f(Y|X,\beta)f(\beta)$$
$$\log(f(\beta|X,Y)) = C + \log(f(Y|X,\beta)) + \log(f(\beta))$$
$$\log(f(\beta|X_r,Y_r)) = E + h(\beta) - \log(f(Y_e|X_e,\beta))[\text{Equation } 1e]$$
$$h(\beta) = \log(f(Y|X,\beta)) + \log(f(\beta))[\text{Equation } 1a]$$

3.2 Proposal Function

We make the proposal function to move in direction of the unlearning step. We need to define L, which is the loss function for the unlearning. We need the loss to be less when the conditional density of β given the dataset is high. We use Equations 2b and 2c to define Δ and H,

$$L(\beta; X_e, Y_e) = C - \log(f(\beta|X_e,Y_e)) = -\log(f(Y_e|\beta,X_e)) - \log(f(\beta))$$
$$\Delta(\beta) = \nabla L(\beta; X_e, Y_e) = -\nabla(\log(f(Y_e|\beta,X_e))) - \nabla(\log(f(\beta)))$$
$$H(\beta) = \nabla^2 L(\beta; X_r, Y_r) = \nabla^2 L(\beta; X, Y) - \nabla^2 L(\beta; X_e, Y_e)$$

We start with $\beta^{(1)} = \hat{\beta} = \arg\min_\beta [\log(f(\beta|X,Y))]$
[Guo et al., 2023] proposed using a Newton Update to predict the mode of the updated distribution. We can use the distribution of the update to approximate the new proposal function. We can set a hyper parameter h to control the Newton Update. As $n \to \infty$, we have $\beta^{(n)} \sim f(.|X_r,Y_r)$. Using Equation 2d,

$$\beta^{(n+1)} = \beta^{(n)} + hH(\beta^{(n)})^{-1}\Delta(\beta^{(n)}) \tag{3}$$

Say, $j(.) = H(.)^{-1}\Delta(.)$ and $H^{-1}\Delta \sim G$. G is the distribution of the transformation of the approximate unlearned distribution.
Since, we don't have the exact distribution, we can use the idea of a Laplace Approximation [[Bergamin et al., 2023]] to approximate the distribution as a normal distribution. In our experiment, we center it at the initially simulated value.

$$\hat{\mu} = H(\beta^{(n)})^{-1}\Delta(\beta^{(n)}) \tag{4a}$$

$$|\frac{\partial j}{\partial \beta}|g(\beta) = f(\beta|X_r, Y_r) \tag{4b}$$

$$-\nabla^2 \log(g(\beta)) = -\nabla^2(\log(f(\beta|X_r, Y_r)) - \log(|\frac{\partial j}{\partial \beta}|)) \tag{4c}$$

$$-\nabla^2 \log(g(\beta)) \approx -\nabla^2(\log(f(\beta|X_r, Y_r))) \tag{4d}$$

$$\Sigma = -\nabla^2 \log(g(\beta^{(n)})) \approx -\nabla^2(\log(f(\beta^{(n)}|X_r, Y_r))) \tag{4e}$$

$$\beta^{(n+1)}|\beta^{(n)} \sim N(\beta^{(n)} + hH(\beta^{(n)})^{-1}\Delta(\beta^{(n)}), h^2H(\beta^{(n)})^{-1})[\text{Equation 3}] \tag{4f}$$

3.3 Pre Processing Data

We can't directly use the method in Sub Section 3.2, because the Proposal Function involves the Complete Data, which we can't access while running the MCMC algorithm, because of Time Inefficiency. As can be seen in Sub Section 3.1 and 3.2, we require the full dataset only for $h(\beta)$ and $\nabla^2 L(\beta|X, Y)$. We can pre compute the values for those variables from random samples from $f(\beta|X, Y)$. If we use MCMC to draw samples, we can also add some samples from a Multivariate Normal Distribution centered at the MLE or MAP estimate of β. (This is because while using MCMC, it is possible to repeat samples, reducing the number of unique samples taking per run)

3.4 Approximation Function

Suppose we are given a β and we need to approximate g(β), and we are given a set $\beta_1, ..., \beta_n$ and $g(\beta_1), ..., g(\beta_n)$. For approximation, we used the Kernel Approximation to approximate the function. We used the Kernel $K(x, y) = \frac{1}{||x-y||_2^2}$. We chose kernel as such because the Gaussian Kernel proved to be very unstable for larger distances. Using Algorithm 1, we can obtain the approximation for $H(\beta) = \nabla^2 L(\beta|X, Y)$ and $h(\beta)$.

Algorithm 1 Approximating any function g

1: Sort $\beta_1, ..., \beta_n$ by $||\beta - \beta_i||_2^2$ in increasing order, and label it as $\beta_{(1)}, ..., \beta_{(n)}$.
2: Choose k as the k nearest neighbors to use to approximate $g(\beta)$.
3: $\hat{g}(\beta) = \frac{\Sigma_{i=1}^{k} K(\beta,\beta_{(i)})g(\beta_{(i)})}{\Sigma_{i=1}^{k} K(\beta,\beta_{(i)})}$

3.5 Nuevo MCMC Algorithm

We can use Algorithm 2 to sample from the unlearning distribution(Eq 1e). To sample from the unlearning distribution using Algorithm 2, we use the proposal function to use in the Metropolis Hasting Algorithm to sample from the new distribution ([Robert and Casella, 1999]). Currently, the tentative name for the algorithm is Nuevo MCMC Algorithm.

4 Logistic Regression

We will now try to frame the machine unlearning algorithm for a Logistic Regression Model. For a model like that of Logistic Regression, where a closed form solution is not possible, approximation techniques

Algorithm 2 Sampling from the Proposed Algorithm

1: Initialize Number of Observations n and Number of Points to consider for approximation k.
2: Initialize $\beta_1 = \hat{\beta}$, where $\hat{\beta}$ is the MLE solution for the Complete Data.
3: Set i = 1
4: Draw $Y_{(i+1)} \sim N(\beta_i + h\hat{H}(\beta_i)^{-1}\Delta(\beta_i), h^2\hat{H}(\beta_i)^{-1})$ [Equation 4f]
5: Let $\hat{h}(x)$ be the approximation of $h(x)$, and $\hat{H}(x)$ be the approximation for $H(x)$ as shown in Subsection 3.4.
6: $\Delta \log f = \log(f(Y_{i+1}|X_r, Y_r)) - \log(f(\beta_i|X_r, Y_r))$
7: $\Delta \log f = h(Y_{i+1}) - h(\beta_i) - \log(f(Y_{i+1}|X_e, Y_e)) + \log(f(\beta_i|X_e, Y_e))$
8: $\Delta \log f \approx \hat{h}(Y_{i+1}) - \hat{h}(\beta_i) - \log(f(Y_{i+1}|X_e, Y_e)) + \log(f(\beta_i|X_e, Y_e))$
9: $\Delta \log g = \log(N(\beta_i|Y_{i+1} + hH(Y_{i+1})^{-1}\Delta(Y_{i+1}))) - \log(N(Y_{i+1}|\beta_i + hH(\beta_i)^{-1}\Delta(\beta_i)))$
10: $\Delta \log g \approx \log(N(\beta_i|Y_{i+1} + h\hat{H}(Y_{i+1})^{-1}\Delta(Y_{i+1}))) - \log(N(Y_{i+1}|\beta_i + h\hat{H}(\beta_i)^{-1}\Delta(\beta_i)))$
11: $\alpha = \exp(\Delta \log f + \Delta \log g)$
12: Draw $X \sim \text{Bern}(\alpha)$.
13: If $X = 1$, set $\beta_{i+1} = Y_{i+1}$.
14: If $X = 0$, set $\beta_{i+1} = \beta_i$.
15: Set $i = i + 1$
16: If $i < n$, go back to Step 4. If $i = n$, stop.

are important to approximate the distribution for the parameters. In this section, we start by getting the distribution of the Parameters for the Remaining Dataset in terms of the Erased dataset, as that is computationally more efficient to compute. Then, we go on to get a Proposal Distribution.

4.1 Distribution for Remaining Dataset

We can get the distribution of the Y given X and β, and use that to get the conditional distribution of β for a Logistic Regression Model, as done in Subsection 3.1.

$$f(Y|X,\beta) = \Pi_{i=1}^{n} \frac{e^{y_i x_i^T \beta}}{1 + e^{x_i^T \beta}}$$
$$f(\beta|X,Y) = \frac{f(Y|X,\beta)f(\beta)}{f(Y|X)}$$
$$f(\beta|X,Y) \propto \Pi_{i=1}^{n} \frac{e^{y_i x_i^T \beta}}{1 + e^{x_i^T \beta}} f(\beta)$$
$$\log(f(\beta|X,Y)) = C + \Sigma_{i=1}^{n}(y_i x_i^T \beta - \log(1 + e^{x_i^T \beta})) + \log(f(\beta))$$
$$\log(f(\beta|X_r,Y_r)) = D + h(\beta) - \log(f(\beta|X_e,Y_e))[\text{Equation } 1a]$$
$$h(\beta) = \log(f(\beta|X,Y)) + \log(f(\beta))[\text{Equation } 1e]$$

We assume a uniform prior for this case.

4.2 Proposal Function

We make the proposal function to move in direction of the unlearning step. We use Equations 2b and 2c to define Δ and H,

$$L(\beta|X_e,Y_e) = \Sigma_{i=1}^{n_e}(\log(1 + e^{x_{ei}^T\beta}) - y_i x_{ei}^T \beta) \tag{5a}$$
$$\Delta(\beta) = \nabla L(\beta|X_e,Y_e) = \Sigma_{i=1}^{n_e}(\frac{e^{x_{ei}^T\beta}}{1 + e^{x_{ei}^T\beta}} x_i - y_i x_{ei}) \tag{5b}$$
$$H(\beta) = \nabla^2 L(\beta|X_r,Y_r) = \Sigma_{i=1}^{n}(\frac{e^{x_i^T\beta}}{(1 + e^{x_i^T\beta})^2} x_i x_i^T) - \Sigma_{i=1}^{n_e}(\frac{e^{x_{ei}^T\beta}}{(1 + e^{x_{ei}^T\beta})^2} x_{ei} x_{ei}^T) \tag{5c}$$

We can use Equation 5 to get the Proposal Distribution, from Equation 4f.

4.3 Experiment

4.3.1 Obtaining Data

We randomly generated 10^6 samples from $N_2(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix})$ for our independent variable x. We can obtain our Design Matrix X by adding in the intercept column. We choose our parameters for the Logistic Regression as $\beta = \begin{bmatrix}-0.5 & 1.2 & -0.8\end{bmatrix}^T$. We get our response variable Y as $Y_i \sim Bern(\frac{1}{1+e^{-\beta^T X_i}})$. We then randomly removed 100 points where the value of Y = 1.

4.3.2 Pre Processing Data

The steps for Pre Processing are laid out in Subsection 3.3. We can use that data, and the Approximation function in 3.4 to simulate the Nuevo MCMC Algorithm for Logistic Regression Data. For approximation, we chose k = 200.

4.4 Results

To evaluate the performance of the proposed algorithm, we compared it with the Random Walk (RW) Metropolis-Hastings algorithm on the erased dataset and MALA Unlearning Algorithm. The following metrics were assessed:

Benchmarking

Algorithm	Time Elapsed	Relative Time
Random Walk Algorithm	64.285	2.149
MALA Unlearning Algorithm	29.915	1.000
Nuevo Unlearning Algorithm	30.045	1.004

Parameter Estimates

- **Mean:**

$$\text{Random Walk Algorithm: } \begin{bmatrix}-0.50620201.2113512 & -0.8114267\end{bmatrix}^T$$

$$\text{Nuevo Unlearning Algorithm: } \begin{bmatrix}-0.4900239 & 1.2374902 & -0.8127756\end{bmatrix}^T$$

$$\text{MALA Unlearning Algorithm: } \begin{bmatrix}-0.50906511.2226965 & -0.8154638\end{bmatrix}^T$$

- **Variance:**

$$\text{Random Walk Algorithm: } \begin{bmatrix} 0.2562937 & -0.6131955 & 0.4107545 \\ -0.6131955 & 1.4674473 & -0.9829258 \\ 0.4107545 & -0.9829258 & 0.6584841 \end{bmatrix}$$

$$\text{Nuevo Unlearning Algorithm: } \begin{bmatrix} 0.2401249 & -0.6063997 & 0.3982794 \\ -0.6063997 & 1.5313833 & -1.0058016 \\ 0.3982794 & -1.0058016 & 0.6606060 \end{bmatrix}$$

$$\text{MALA Unlearning Algorithm: } \begin{bmatrix} 0.2591486 & -0.6224324 & 0.4151244 \\ -0.6224324 & 1.4949878 & -0.9970648 \\ 0.4151244 & -0.9970648 & 0.6649829 \end{bmatrix}$$

Comparison with Mode

- Mode for Remaining Dataset:

$$\begin{bmatrix} -0.5052315 & 1.2103321 & -0.8100823 \end{bmatrix}^T$$

  Mode for Full Dataset:

$$\begin{bmatrix} -0.5025292 & 1.2098063 & -0.8097571 \end{bmatrix}^T$$

- Random Walk Algorithm:

$$\begin{bmatrix} -0.5060576 & 1.2052346 & -0.8122596 \end{bmatrix}^T$$

  Nuevo Unlearning Algorithm:

$$\begin{bmatrix} -0.4898499 & 1.2377373 & -0.8126076 \end{bmatrix}^T$$

  MALA Unlearning Algorithm:

$$\begin{bmatrix} -0.5090211 & 1.2228329 & -0.8167732 \end{bmatrix}^T$$

Comparison with Geweke Diagonal

- Random Walk Algorithm:

var1	var2	var3
1.4393	0.3051	0.9203

- Nuevo Unlearning Algorithm:

var1	var2	var3
-0.633616	-0.005534	-2.374102

- MALA Unlearning Algorithm:

var1	var2	var3
0.04597	-0.74326	-0.35311

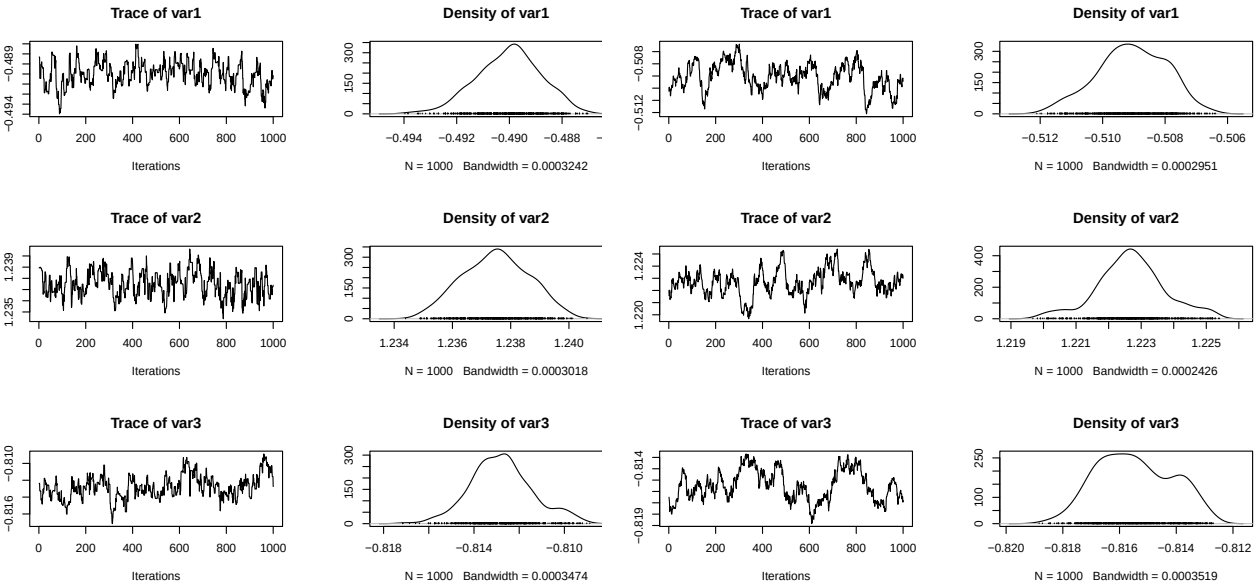
Comparison with Minimum Effective Sample Size

- Random Walk Algorithm: 93.82361
- Nuevo Unlearning Algorithm: 35.56561
- MALA Unlearning Algorithm: 11.1361

Comparison with Minimum Effective Sample Size per Unit Time

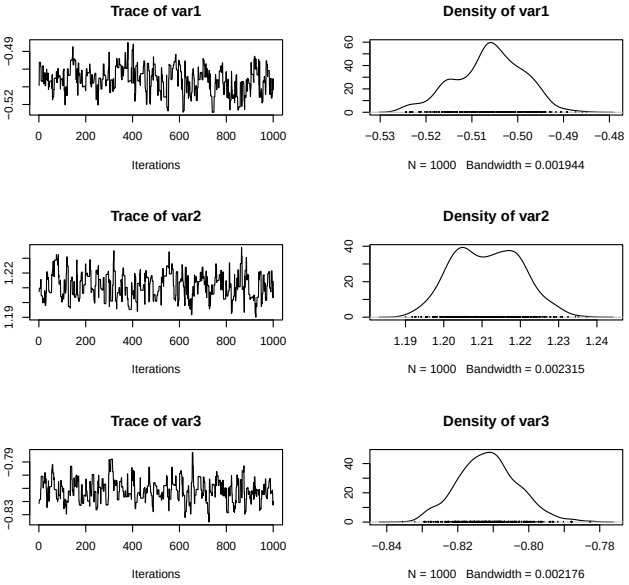
- Random Walk Algorithm: 0.1074911
- Nuevo Unlearning Algorithm: 0.09865141
- MALA Unlearning Algorithm: 0.0307684

Comparison of Trace Plots The Trace Plots for the different algorithms are compared in Figure 1.



((a)) Nuevo Unlearning Algorithm

((b)) MALA Unlearning Algorithm



((c)) Random Walk Algorithm

Figure 1: Trace Plot for Logistic Regression Data

5 Negative Binomial Regression

We will now try to frame the machine unlearning algorithm for a Negative Binomial Regression Model. For a model like that of Negative Binomial Regression, where a closed form solution is not possible, approximation techniques are important to approximate the distribution for the parameters. In this section, we start by getting the distribution of the Parameters for the Remaining Dataset in terms of the Erased dataset, as that is computationally more efficient to compute. Then, we go on to get a Proposal Distribution.

5.1 Distribution for Remaining Dataset

We can get the distribution of the Y given X and β, and use that to get the conditional distribution of β for a Negative Binomial Regression Model, as done in Subsection 3.1.

$$
\begin{aligned}
f(Y|X,\beta) &= \Pi_{i=1}^{n} \binom{Y_i + r - 1}{r} \left(\frac{r}{r + e^{\beta^T X_i}}\right)^r \left(\frac{e^{\beta^T X_i}}{r + e^{\beta^T X_i}}\right)^{Y_i} \\
f(\beta|X,Y) &= \frac{f(Y|X,\beta) f(\beta)}{f(Y|X)} \\
f(\beta|X,Y) &\propto \Pi_{i=1}^{n} \binom{Y_i + r - 1}{r} \left(\frac{r}{r + e^{\beta^T X_i}}\right)^r \left(\frac{e^{\beta^T X_i}}{r + e^{\beta^T X_i}}\right)^{Y_i} f(\beta) \\
\log(f(\beta|X,Y)) &= C + \Sigma_{i=1}^{n} Y_i \beta^T X_i - (r + Y_i) \log(r + e^{\beta^T X_i}) + \log(f(\beta)) \\
\log(f(\beta|X_r,Y_r)) &= D + h(\beta) - \log(f(\beta|X_e,Y_e)) [\text{Equation } 1a] \\
h(\beta) &= \log(f(\beta|X,Y)) + \log(f(\beta)) [\text{Equation } 1e]
\end{aligned}
$$

We assume a uniform prior for this case, and take r = 5.

5.2 Proposal Function

We make the proposal function to move in direction of the unlearning step. We use Equations 2b and 2c to define Δ and H,

$$L(\beta|X_e,Y_e) = \Sigma_{i=1}^{n_e} ((r + Y_{ei}) \log(r + e^{\beta^T X_{ei}}) - Y_{ei} \beta^T X_{ei}) \tag{6a}$$

$$\Delta(\beta) = \nabla L(\beta|X_e,Y_e) = \Sigma_{i=1}^{n_e} \frac{(r + Y_{ei}) e^{\beta^T X_{ei}}}{r + e^{\beta^T X_{ei}}} X_{ei} \tag{6b}$$

$$H(\beta) = \nabla^2 L(\beta|X_r,Y_r) = \Sigma_{i=1}^{n} \frac{r(r + Y_i) e^{\beta^T X_i}}{(r + e^{\beta^T X_i})^2} X_i X_i^T - \Sigma_{i=1}^{n_e} \frac{r(r + Y_{ei}) e^{\beta^T X_{ei}}}{(r + e^{\beta^T X_{ei}})^2} X_{ei} X_{ei}^T \tag{6c}$$

We can use Equation 6 to get the Proposal Distribution, from Equation 4f.

5.3 Experiment

5.3.1 Obtaining Data

We randomly generated 10^5 samples from $N_2(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix})$ for our independent variable x. We can obtain our Design Matrix X by adding in the intercept column. We choose our parameters for the Logistic Regression as $\beta = \begin{bmatrix}-0.5 & 1.2 & -0.8\end{bmatrix}^T$. We get our response variable Y as $Y_i \sim \text{NB}(e^{\beta^T X_i}, r)$. We then randomly removed 100 points where the value of Y = 1.

5.3.2 Pre Processing Data

The steps for Pre Processing are laid out in Subsection 3.3. We can use that data, and the Approximation function in 3.4 to simulate the Nuevo MCMC Algorithm for Negative Binomial Regression Data. For approximation, we chose k = 150.

5.4 Results

To evaluate the performance of the proposed algorithm, we compared it with the Random Walk (RW) Metropolis-Hastings algorithm on the erased dataset and MALA Unlearning Algorithm. The following metrics were assessed:

Benchmarking

Algorithm	Time Elapsed	Relative Time
Random Walk Algorithm	79.004	2.288
MALA Unlearning Algorithm	34.525	1.000
Nuevo Unlearning Algorithm	36.689	1.063

Parameter Estimates

- **Mean:**

$$\text{Random Walk Algorithm: } \begin{bmatrix} -0.5134762 & 1.1995666 & -0.8054334 \end{bmatrix}^T$$

$$\text{Nuevo Unlearning Algorithm: } \begin{bmatrix} -0.5169664 & 1.1981641 & -0.8030984 \end{bmatrix}^T$$

$$\text{MALA Unlearning Algorithm: } \begin{bmatrix} -0.5235552 & 1.1646848 & -0.7658632 \end{bmatrix}^T$$

- **Variance:**

$$\text{Random Walk Algorithm: } \begin{bmatrix} 0.2636902 & -0.6159657 & 0.4135809 \\ -0.6159657 & 1.4389781 & -0.9661741 \\ 0.4135809 & -0.9661741 & 0.6487403 \end{bmatrix}$$

$$\text{Nuevo Unlearning Algorithm: } \begin{bmatrix} 0.2673177 & -0.6194041 & 0.4151623 \\ -0.6194041 & 1.4355988 & -0.9622455 \\ 0.4151623 & -0.9622455 & 0.6449699 \end{bmatrix}$$

$$\text{MALA Unlearning Algorithm: } \begin{bmatrix} 0.2741855 & -0.6096203 & 0.4008272 \\ -0.6096203 & 1.3569183 & -0.8923368 \\ 0.4008272 & -0.8923368 & 0.5868531 \end{bmatrix}$$

Comparison with Mode

- Mode for Remaining Dataset:

$$\begin{bmatrix} -0.5134810 & 1.1994306 & -0.8055406 \end{bmatrix}^T$$

  Mode for Full Dataset:

$$\begin{bmatrix} -0.5018035 & 1.1985711 & -0.8052665 \end{bmatrix}^T$$

- Random Walk Algorithm:

$$\begin{bmatrix} -0.5102020 & 1.1974100 & -0.8048505 \end{bmatrix}^T$$

  Nuevo Unlearning Algorithm:

$$\begin{bmatrix} -0.5204285 & 1.1982793 & -0.8028384 \end{bmatrix}^T$$

  MALA Unlearning Algorithm:

$$\begin{bmatrix} -0.5322786 & 1.1442481 & -0.7495094 \end{bmatrix}^T$$

Comparison with Geweke Diagonal

- Random Walk Algorithm:

var1	var2	var3
-2.2868	2.6902	0.9165

- Nuevo Unlearning Algorithm:

var1	var2	var3
7.488	1.032	-3.211

- MALA Unlearning Algorithm:

var1	var2	var3
11.716	6.478	-12.630

Comparison with Minimum Effective Sample Size

- Random Walk Algorithm: 43.03326
- Nuevo Unlearning Algorithm: 1.517416
- MALA Unlearning Algorithm: 1.254615

Comparison with Minimum Effective Sample Size per Unit Time

- Random Walk Algorithm: 0.06157883
- Nuevo Unlearning Algorithm: 0.004674394
- MALA Unlearning Algorithm: 0.004177607

Comparison of Trace Plots The Trace Plots for the different algorithms are compared in Figure 2. All code for this experiment is available on Github.

6 Work to be Done

This model gives good results if we initialize with the optimal beta for the complete dataset. I wish to extend it for more vast initialization points. I would also like to explore how to fine tune the Hyperparameter h. I would also like to test methods to tune hyperparameters for each dimensional parameters.

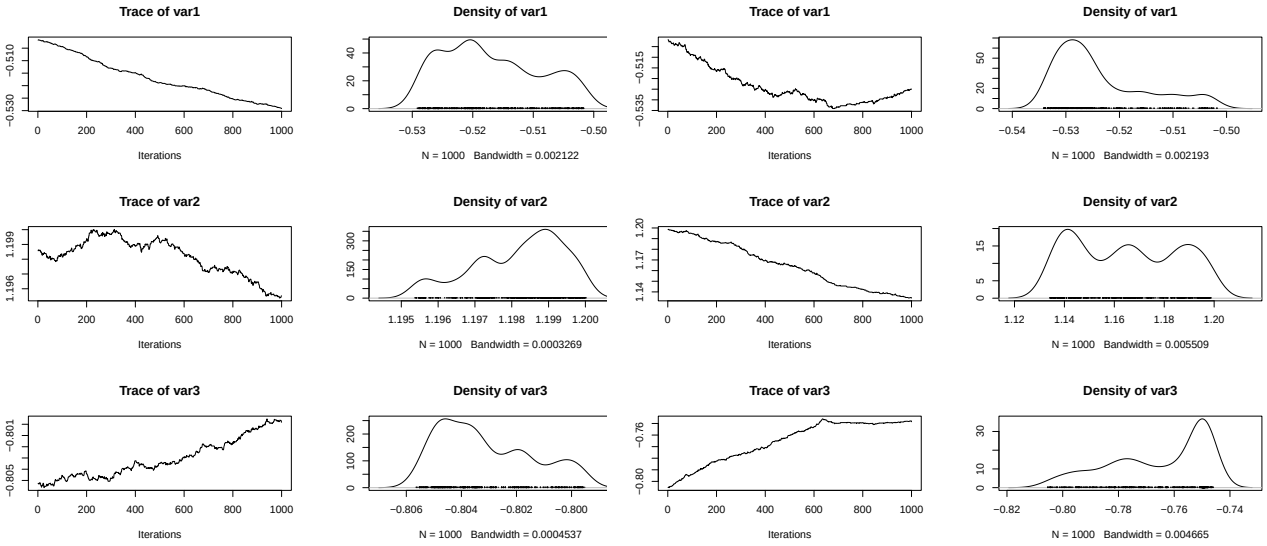
References

[Bergamin et al., 2023] Bergamin, F., Moreno-Muñoz, P., Hauberg, S., and Arvanitidis, G. (2023). Riemannian laplace approximations for bayesian neural networks.

[Guo et al., 2023] Guo, C., Goldstein, T., Hannun, A., and van der Maaten, L. (2023). Certified data removal from machine learning models.

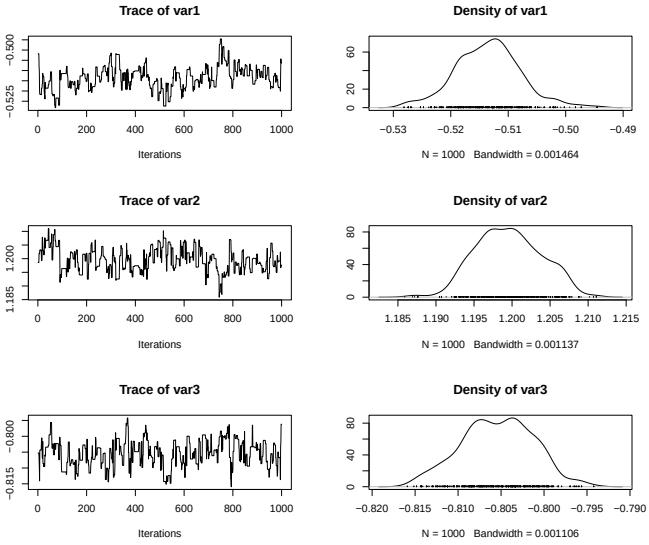
[Nguyen et al., 2022] Nguyen, Q. P., Oikawa, R., Divakaran, D. M., Chan, M. C., and Low, B. K. H. (2022). Markov chain monte carlo-based machine unlearning: Unlearning what needs to be forgotten. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '22, page 351–363, New York, NY, USA. Association for Computing Machinery.

[Robert and Casella, 1999] Robert, C. P. and Casella, G. (1999). *The Metropolis—Hastings Algorithm*, pages 231–283. Springer New York, New York, NY.



((a)) Nuevo Unlearning Algorithm

((b)) MALA Unlearning Algorithm



((c)) Random Walk Algorithm

Figure 2: Trace Plot for Negative Binomial Data