Markov Chain Monte Carlo-Based Machine Unlearning: Unlearning What Needs to be Forgotten

Quoc Phong Nguyen
National University of Singapore
qphong@comp.nus.edu.sg

Ryutaro Oikawa
National University of Singapore
ryutaro.oikawa.1991@gmail.com

Dinil Mon Divakaran
Trustwave
dinil.divakaran@trustwave.com

Mun Choon Chan
National University of Singapore
chanmc@comp.nus.edu.sg

Bryan Kian Hsiang Low
National University of Singapore
lowkh@comp.nus.edu.sg

ABSTRACT

As the use of machine learning (ML) models is becoming increasingly popular in many real-world applications, there are practical challenges that need to be addressed for model maintenance. One such challenge is to 'undo' the effect of a specific subset of dataset used for training a model. This specific subset may contain malicious or adversarial data injected by an attacker, which affects the model performance. Another reason may be the need for a service provider to remove data pertaining to a specific user to respect the user's privacy. In both cases, the problem is to 'unlearn' a specific subset of the training data from a trained model without incurring the costly procedure of retraining the whole model from scratch. Towards this goal, this paper presents a Markov chain Monte Carlo-based machine unlearning (MCU) algorithm. MCU helps to effectively and efficiently unlearn a trained model from subsets of training dataset. Furthermore, we show that with MCU, we are able to explain the effect of a subset of a training dataset on the model prediction. Thus, MCU is useful for examining subsets of data to identify the adversarial data to be removed. Similarly, MCU can be used to erase the lineage of a user's personal data from trained ML models, thus upholding a user's "right to be forgotten". We empirically evaluate the performance of our proposed MCU algorithm on real-world phishing and diabetes datasets. Results show that MCU can achieve a desirable performance by efficiently removing the effect of a subset of training dataset and outperform an existing algorithm that utilizes the remaining dataset.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**.

ACM Reference Format:
Quoc Phong Nguyen, Ryutaro Oikawa, Dinil Mon Divakaran, Mun Choon Chan, and Bryan Kian Hsiang Low . 2022. Markov Chain Monte Carlo-Based Machine Unlearning: Unlearning What Needs to be Forgotten. In *AsiaCCS '22: ACM ASIA Conference on Computer and Communications Security, May 30th to June 3rd, 2022, Nagasaki, Japan.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3488932.3517406

AsiaCCS '22, May 30th to June 3rd, 2022, Nagasaki, Japan

ACM ISBN 978-1-4503-9140-5/22/05…$15.00
https://doi.org/10.1145/3488932.3517406

1 INTRODUCTION

While there has been a rapid increase in machine learning (ML) applications, they often require accurately labeled datasets to achieve competitive performance. This is also common in the security domain where supervised classifiers are built for threat detection (e.g., for detecting phishing attacks [9, 39–43], malware and malicious communications [10–12, 47, 48, 51, 55]). However, even with significant and costly human verification [1, 45], these datasets are prone to errors and poisoning attacks. For example, in the case of phishing, attackers often host the phishing pages on compromised sites [53] which are taken down and handed over to the website owners when detected. Thus, due to the short life cycle of phishing attacks [15, 50], not all URLs gathered from a feed (e.g., OpenPhish [6] or PhishTank [7]) correspond to valid phishing webpages. Such errors inadvertently corrupt the dataset with wrong labels. This undesirable outcome can also be caused by an adversary that deliberately injects malicious data into a training dataset [32, 38, 57] (e.g., see [56] for backdoor poisoning attack against malware classifiers), thereby degenerating the model performance. A straightforward solution is to retrain the model from scratch after removing the unwanted[1] (erroneous or malicious) data from the training dataset. But, such an approach is not practical as it incurs a significant amount of time and storage, especially when the *remaining dataset* (after removing the unwanted data) is large. Besides, as discussed below, since user privacy is of paramount importance, retaining data is not an option in all use cases [29].

While the above settings assume that the set of unwanted data is given, there is also a related and important problem of identifying the subset of training data that is malicious (or erroneous) by examining a number of subsets of data. This problem arises when the training dataset is formed using different subsets acquired from multiple sources: For example, phishing URLs and malicious domains can be obtained from multiple sources such as OpenPhish [6], PhishTank [7], APWG [5], URLhaus [8], Anomali [4], and other commercial as well as open source threat intelligence feeds. Among these subsets of training data, the objective is to examine whether there exists a subset of malicious data. We consider the approach of analyzing the model prediction after removing a subset of the training data. In particular, a subset of the training data is potentially malicious if removing it from the model increases the accuracy in

[1] We also refer to the unwanted dataset as the *erased dataset*.

the model prediction (e.g., on a test set). Due to the vast number of the subsets of a training dataset to be examined, it is not practical to retrain the model numerous times to identify the effect of removing each subset on the model prediction. This problem is also important when we want to apportion credit between different network analysts that label different parts of the data.

The problem of removing a subset of the training dataset from a trained model also arises in the domain of user privacy. In particular, a company or service provider needs to remove a user's personal data when she would like to exercise the "right to be forgotten" [58]. The data points are not only stored in the database (which can be easily deleted), but they are also used to train ML models, thus forming the *data lineage* [19]. Therefore, to respect the privacy of a user, it is important that her data's lineage is also erased from the trained ML models upon request. Since the number of users who demand their data to be removed (e.g., those who stop using a service from a company) is often much smaller than that of the remaining users (e.g., active users of the service), retraining the model from scratch by excluding the erased data is prohibitively expensive. Furthermore, in some cases, the majority of the user data may not be accessible due to storage limitations or policies restricting the retention period of data.

Note that, while there exist approaches to learn with noisy labels [13, 31, 52], they address a problem different from what we have discussed above. The above scenarios highlight a common problem of removing the effect of a subset of the training dataset from a model without retraining the model from scratch, which is called *machine unlearning* (MU) [19, 24, 28, 30, 49]. When the accuracy of the model after being unlearned is not crucial, the approach of analyzing the model prediction after removing a subset of the training data is also investigated in the *interpretable ML* literature [37, 38].

Contribution: In this work, we address the problem of unlearning, under the assumption that there are samples of the model parameters which estimate the posterior belief of the model parameters well. In particular, we employ a *Markov chain Monte Carlo* (MCMC) algorithm to draw these samples. Hence, our novel proposed approach is named *MCMC-based machine unlearning* (MCU). It can efficiently unlearn a trained model from different subsets of the training dataset. The key component of MCU is a *candidate set* of the model parameters such that the parameters of the model retrained on the remaining dataset (i.e., the naively *retrained model*) are close to a candidate in the candidate set (Section 4.1).

We also assume that the subset of the training data to be erased is small relative to the whole training dataset, which means that the change in model parameters by removing the subset is small. Hence, the posterior probability of the retrained model parameters given the training dataset should be sufficiently large. This is the rationale behind our selection of the candidate set as the set of MCMC samples drawn from the posterior belief of the model parameters given the training dataset. The candidate set is equipped with auxiliary values based on the importance sampling technique that are useful in performing unlearning (Section 4.2). We further relax our assumption on the small change of the model parameters by proposing an *enlarged candidate set* based on a flattened distribution (Section 4.3). Additionally, MCU can be used to explain the effect of a subset of training data on the model prediction (Section 5). We also show that MCU does not suffer from an important pitfall of *catastrophic unlearning* (Section 6) known to affect MU algorithms.

We empirically demonstrate the advantage of MCU with the enlarged candidate set in three scenarios—removing user's personal data, removing unwanted data, and interpreting the model prediction with respect to subsets of the training data. Note, since MCMC faces difficulty in high dimensional problems [14, 16], our experiments are conducted on problems of moderate dimensions where the above assumption holds, i.e., the MCMC samples estimate the posterior distribution of the model parameters well. Nonetheless, we perform extensive experiments using both synthetic and real-world datasets, the latter of which include the Pima Indians diabetes dataset and a phishing webpage detection dataset (Section 7). Results show that our proposed enlarged candidate set improves the performance of MCU significantly. Furthermore, while not using the remaining dataset during unlearning, MCU with an enlarged candidate set can outperform an existing MU algorithm [26, 27] that however utilizes the remaining dataset in its unlearning procedure. To the best of our knowledge, MCU is the only work on MU for MCMC algorithms without using the remaining dataset.

2 RELATED WORKS

The pioneering work [19] addresses the MU problem for statistical query learning [36] which includes several ML algorithms such as naive Bayes classifier, support vector machine, and k-means clustering. However, they are required to have a summation form. To perform unlearning, the erased data are simply subtracted from the summations and the model is updated. The work of [17] proposes to divide the training dataset into multiple shards, each of which is trained with a different model. By assuming that the erased dataset belongs to a small number of shards, the computation required in retraining models is reduced. Another class of research works [26, 27, 33] utilize the notion of an influence function [23, 38] to unlearn a model from a datum (i.e., an individual data point). In particular, the work of [33] can unlearn logistic regression models, while that of [26, 27] can unlearn Bayesian models such as those obtained from MCMC algorithms. However, since the influence function is based on the first-order Taylor approximation, these unlearning algorithms are only accurate when there is a small change in the model (e.g., erasing a datum from a large training dataset), which makes them less practical for real-world use. There are also other works that focus on unlearning specific learning algorithms such as k-means clustering [30] and variational inference [49].

3 BACKGROUND

3.1 Machine Unlearning

We focus on supervised learning problems where the training dataset consists of input and class-label pairs. ML models are used to capture the conditional probability $p(y|\mathbf{x}, \boldsymbol{\theta})$ of a label y given an input $\mathbf{x}$, and the model parameters $\boldsymbol{\theta}$. Let the training data be denoted by $\mathcal{D} \triangleq \{(\mathbf{x}_i, y_i)\}_{i=1}^n$. Given the training dataset, we would like to find the *maximum a posteriori* (MAP) estimate of the model parameters. In other words, the optimal model parameters $\boldsymbol{\theta}_\mathcal{D}$ maximize the *posterior* probability $p(\boldsymbol{\theta}|\mathcal{D})$ of the model parameters

given the training data:

$$\boldsymbol{\theta}_{\mathcal{D}} \triangleq \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \log p(\boldsymbol{\theta}|\mathcal{D}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left(\log p(\mathcal{D}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\right) \quad (1)$$

where $p(\mathcal{D}|\boldsymbol{\theta})$ and $p(\boldsymbol{\theta})$ are the *likelihood* and the *prior*, respectively. The data points are assumed to be conditionally independent given the model parameters, which holds for many ML models such as regression models and feed-forward neural networks:

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \log \prod_{i=1}^{n} p(y_i|\mathbf{x}_i, \boldsymbol{\theta}) = \sum_{i=1}^{n} \log p(y_i|\mathbf{x}_i, \boldsymbol{\theta}) \ . \quad (2)$$

Let us consider the *machine unlearning* scenario where we would like to remove the effect of an *erased dataset*, denoted by $\mathcal{D}_e \subset \mathcal{D}$, from a trained model specified by $\boldsymbol{\theta}_{\mathcal{D}}$ [19]. Recall that $\boldsymbol{\theta}_{\mathcal{D}}$ is the MAP estimate of the model parameters given the training data $\mathcal{D}$. It is obtained by maximizing the log posterior probability of $\boldsymbol{\theta}$ given $\mathcal{D}$ (Eq. (1)). When the erased dataset $\mathcal{D}_e$ is relatively large compared to the *remaining dataset*, denoted by $\mathcal{D}_r \triangleq \mathcal{D} \setminus \mathcal{D}_e$ (obtained by removing the erased dataset $\mathcal{D}_e$ from $\mathcal{D}$), retraining the model from scratch using the remaining dataset $\mathcal{D}_r$ is a possible solution, i.e., finding

$$\boldsymbol{\theta}_{\mathcal{D}_r} \triangleq \operatorname{argmax}_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\mathcal{D}_r) \ .$$

In most practical scenarios, the erased dataset $\mathcal{D}$ is small relative to the size of entire dataset $\mathcal{D}$. For example, whether it is applying unsupervised deep learning models to detect anomalies in network traffic [48] or removal of user data (lineage) from existing trained models, $\mathcal{D}_r$ is typically much larger than $\mathcal{D}_e$. Therefore, retraining the model on $\mathcal{D}_r$ from scratch can be inefficient in terms of computational time and impractical in terms of storing data indefinitely due to both storage constraints and/or regulatory policies. Also, note that when $\mathcal{D}_e$ is small, the difference between the parameters $\boldsymbol{\theta}_{\mathcal{D}}$ of the model trained on $\mathcal{D}$ vs. the parameters $\boldsymbol{\theta}_{\mathcal{D}_r}$ of the model trained on $\mathcal{D}_r$ is less drastic. So, it is possible for MU algorithms to achieve a more efficient solution without resorting to the naive approach of retraining with the remaining data $\mathcal{D}_r$ [19].

Given the likelihood function and the prior distribution in Eq. (1), let us review a stochastic approximation method of the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ of the model parameters given the training dataset called the *Markov chain Monte Carlo* (MCMC) sampling. It will be used to construct the *candidate set* in our proposed MCU algorithm later in Section 4.1.

3.2 Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) algorithms are used to draw samples from a *target distribution* when directly sampling from it is difficult and only a function that is proportional to its density function is available [18]. These algorithms are particularly useful in drawing a set of samples to approximate the posterior distribution of a random variable in Bayesian models. In the context of our work, MCMC algorithms allow us to acquire a set of model parameters that are likely to be close to the retrained model parameters $\boldsymbol{\theta}_{\mathcal{D}_r}$ without the knowledge of the erased dataset $\mathcal{D}_e$ (or $\mathcal{D}_r$), as explained later in Section 4.1.

Consider the ML model in the previous section that is parameterized with parameters $\boldsymbol{\theta}$. Given the likelihood of the training dataset $p(\mathcal{D}|\boldsymbol{\theta})$ and the prior distribution $p(\boldsymbol{\theta})$, the posterior distribution of $\boldsymbol{\theta}$ given the training data $\mathcal{D}$ is obtained with the Bayes' rule:

$$p(\boldsymbol{\theta}|\mathcal{D}) = p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})/p(\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \ . \quad (3)$$

While the prior distribution $p(\boldsymbol{\theta})$ and the likelihood $p(\mathcal{D}|\boldsymbol{\theta})$ are often available from the model, it is difficult to evaluate the marginal likelihood (or the evidence) $p(\mathcal{D}) = \int p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}$, e.g., when $p(\boldsymbol{\theta})$ is not a conjugate prior for the likelihood function $p(\mathcal{D}|\boldsymbol{\theta})$. On the other hand, MCMC algorithms are still able to generate the samples of this target distribution $p(\boldsymbol{\theta}|\mathcal{D})$ using $p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$ (which does not involve $p(\mathcal{D})$) as this function is proportional to the posterior distribution density $p(\boldsymbol{\theta}|\mathcal{D})$, as seen in Eq. (3).

MCMC algorithms construct a Markov chain whose equilibrium distribution is the target distribution, e.g., $p(\boldsymbol{\theta}|\mathcal{D})$. A classic MCMC method is the Metropolis-Hastings (M-H) algorithm described in Algorithm 1 [34, 44]. The algorithm constructs a Markov chain starting from an initial sample $\boldsymbol{\theta}_0$ that can be selected arbitrarily. The *proposal density* $g(\boldsymbol{\theta}'|\boldsymbol{\theta})$ is used to draw the next sample given the current sample. It is chosen as a symmetric distribution centered at the current sample such as a Gaussian distribution. Then the next sample is accepted with the probability $\min(\beta, 1)$ where β is the acceptance ratio in Eq. (4). As the target distribution is only approximated well by the equilibrium distribution of the Markov chain, a number of initial samples are often discarded (which are called *burn-in* samples).

The downside of the M-H algorithm is that it requires a large number of samples to approximate the target density well if the acceptance ratio is low. There have been several MCMC methods that improve the sampling efficiency for a target density of a moderately high dimensional random variable by utilizing the Hamiltonian dynamics such as the *Hamiltonian Monte Carlo* (HMC) [16, 46] and the no-u-turn sampler [35]. There are also methods scalable to large datasets by relying on the stochastic gradients [20, 59, 60] and a symmetric splitting integration scheme for HMC [22]. It is noted that we do not focus on addressing these issues of MCMC in this work. The main focus of our work is to design a MU algorithm given a set of MCMC samples that approximate the posterior distribution well.

Algorithm 1 Metropolis-Hastings Algorithm

Input: $f(\boldsymbol{\theta}) = p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$, proposal density $g(\boldsymbol{\theta}'|\boldsymbol{\theta})$, initial sample $\boldsymbol{\theta}_0$, number of samples M
for $i = 1, 2, \ldots, M$ **do**
  Draw a proposed sample $\boldsymbol{\theta}'$ from $g(\boldsymbol{\theta}|\boldsymbol{\theta}_{i-1})$
  Evaluate the acceptance ratio:

$$\beta = \frac{f(\boldsymbol{\theta}')}{f(\boldsymbol{\theta}_{i-1})} = \frac{p(\mathcal{D}|\boldsymbol{\theta}')p(\boldsymbol{\theta}')}{p(\mathcal{D}|\boldsymbol{\theta}_{i-1})p(\boldsymbol{\theta}_{i-1})} \ . \quad (4)$$

  Draw a uniform random number $u \in [0, 1]$.
  if $u \leq \beta$ **then** ▷ Accept sample with prob. $\min(\beta, 1)$
    $\boldsymbol{\theta}_i = \boldsymbol{\theta}'$
  else ▷ Reject proposed sample
    $\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1}$ ▷ Reuse previous sample
  end if
end for
return $\{\boldsymbol{\theta}\}_{i=1}^{M}$

While MCMC algorithms have been popular methods to draw samples from a target distribution in many ML applications, there are also certain cases such as those in our MCU approach where we would like to obtain a set of samples representing a distribution from another set of samples representing a similar yet different distribution, by assigning weights to these samples. This technique is called *importance sampling*. In particular, we will employ importance sampling to transform the set of MCMC samples representing the distribution $p(\boldsymbol{\theta}|\mathcal{D})$ to an approximation of the distribution $p(\boldsymbol{\theta}|\mathcal{D}_r)$ in Section 4.2 and Section 4.4.

3.3 Importance Sampling

Suppose we are interested in estimating the mean of a random variable $f(\mathbf{x})$ where $\mathbf{x}$ follows a distribution specified by $p(\mathbf{x})$. It can be estimated by:

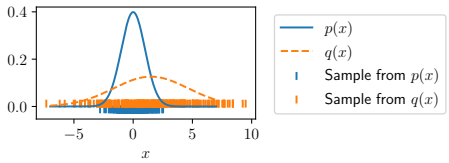
$$\mathbb{E}[f(\mathbf{x})] \triangleq \int f(\mathbf{x})p(\mathbf{x})\,\mathrm{d}\mathbf{x} \approx \frac{1}{|\mathcal{X}|}\sum_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$$

where $\mathcal{X}$ is a set of samples of $\mathbf{x}$ drawn from the distribution specified by $p(\mathbf{x})$.

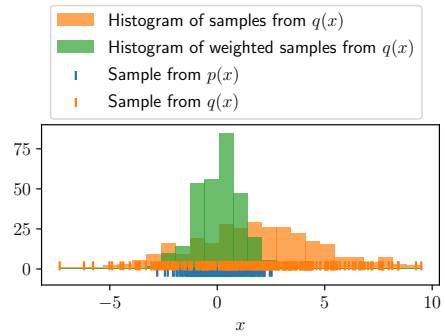
However, the problem arises when it is difficult to draw samples from the distribution specified by $p(\mathbf{x})$. In such case, if there exists a distribution specified by $q(\mathbf{x})$ from which samples of $\mathbf{x}$ can be drawn easily, then *importance sampling* is a popular technique to estimate $\mathbb{E}[f(\mathbf{x})]$ using a set $\mathcal{X}'$ of samples drawn from the distribution specified by $q(\mathbf{x})$:[2]

$$\mathbb{E}[f(\mathbf{x})] = \int q(\mathbf{x})\frac{p(\mathbf{x})}{q(\mathbf{x})}f(\mathbf{x})\,\mathrm{d}\mathbf{x} \approx \frac{1}{|\mathcal{X}'|}\sum_{\mathbf{x}\in\mathcal{X}'}\frac{p(\mathbf{x})}{q(\mathbf{x})}f(\mathbf{x})\,. \quad (5)$$

As an illustration, Figure 1 shows that the samples $x \in \mathcal{X}'$ with the weights defined as the density ratio $p(\mathbf{x})/q(\mathbf{x})$ are able to represent the distribution specified by $p(x)$. The plots of $p(x)$ and $q(x)$ are shown in Figure 1a. We observe that the two densities are different, so their samples are distributed differently. Suppose we cannot draw samples from $p(x)$ and we can only draw samples from $q(x)$. Using the importance sampling technique, we can obtain a set of weighted samples that represent the distribution specified by $p(x)$ from the set of samples of the distribution specified by $q(x)$. In particular, the samples from the distribution specified by $q(x)$ are weighted with the density ratio $p(x)/q(x)$ (in a similar fashion to Eq. (5)). We observe in Figure 1b that the histogram of the weighted samples obtained from importance sampling is able to represent the distribution specified by $p(x)$.



(a) Samples of $p(x)$ and $q(x)$.



(b) The weighted samples of $q(x)$ with importance sampling.

Figure 1: Plots of the weighted samples representing $p(x)$ obtained from the samples of $q(x)$ with importance sampling.

4 MCMC-BASED MACHINE UNLEARNING

We first define the threat model, and then explicitly and formally define the problem of machine unlearning. Common notations used in this paper are listed in Table 1.

Threat model. This work considers two classes of threats in ML systems. (a) A subset of the data used for training is malicious or poisoned by an adversary. This essentially means, part of the labels provided as ground truth is wrong. This can happen due to errors in data collection and labeling, but could also be the result of an adversarial attack [32, 38, 56, 57]; both are relevant as they affect model performance, in terms of classification accuracy. (b) The second class of threats comes from how an ML model needs to be maintained to respect user privacy. A user exercising her right to withdraw the use of her data for any purposes previously agreed to, requires the ML model to erase the data lineage corresponding to the user. To examine whether a model has unlearned a given user's data lineage, we need to evaluate the performance of the unlearned model on the user's data. Contrary to traditional objective of achieving higher accuracy, note that the unlearned model can have lower classification accuracy on the erased dataset.

Problem Definition. The problem is to remove the effects of a set $\mathcal{D}_e$ of unwanted data (also referred to as *erased dataset*) in a model trained on $\mathcal{D} \supset \mathcal{D}_e$, by finding an approximation of the model parameters $\boldsymbol{\theta}_{\mathcal{D}_r}$ without involving the costly procedure of retraining the model on $\mathcal{D}_r \triangleq \mathcal{D} \setminus \mathcal{D}_e$. We make the assumption that the size of erased dataset $\mathcal{D}_e$ is relatively small compared with that of the training dataset $\mathcal{D}$.

Overview of MCU. We approach the problem by constructing a *candidate set* $\boldsymbol{\Theta}$ of the model parameters that is close to the parameters $\boldsymbol{\theta}_{\mathcal{D}_r}$ without the knowledge of $\mathcal{D}_e$ (or $\mathcal{D}_r$) (Section 4.1). While the construction of $\boldsymbol{\Theta}$ involves running an MCMC algorithm, it is pre-computed before the unlearning procedure because it does not require the knowledge of $\mathcal{D}_e$. Therefore, given a dataset $\mathcal{D}_e$ to be erased from the trained model, the pre-computed $\boldsymbol{\Theta}$ (and its pre-computed auxiliary values) can be used to unlearn the model quickly (Section 4.2). We further enlarge the candidate set so that the chance it is closer to the parameters $\boldsymbol{\theta}_{\mathcal{D}_r}$ increases; this is achieved through the introduction of the *enlarged candidate set* by flattening the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ (Section 4.3). Due to the difference between the flattened distribution and $p(\boldsymbol{\theta}|\mathcal{D})$, importance sampling (Section 3.3) is utilized to perform unlearning with this enlarged candidate set (Section 4.4).

[2]Importance sampling requires the condition if $p(\mathbf{x}) > 0$ then $q(\mathbf{x}) > 0$ so that the density ratio $p(\mathbf{x})/q(\mathbf{x})$ is defined for all $\mathbf{x}$.

Table 1: Table of notations.

Notation	Definition
$\mathcal{D}$	The training dataset
$\mathcal{D}_e$	The erased dataset ($\mathcal{D}_e \subset \mathcal{D}$)
$\mathcal{D}_r$	The remaining dataset (i.e., $\mathcal{D}_r \triangleq \mathcal{D} \setminus \mathcal{D}_e$)
$\boldsymbol{\theta}_{\mathcal{D}}$	The parameters of the model trained on $\mathcal{D}$
$\boldsymbol{\theta}_{\mathcal{D}_r}$	The parameters of the model (re)trained on $\mathcal{D}_r$
$\boldsymbol{\Theta}$	The candidate set
$w(\boldsymbol{\theta})$	The weight of a candidate $\boldsymbol{\theta} \in \boldsymbol{\Theta}$
$\tilde{p}(\boldsymbol{\theta}\|\mathcal{D};\alpha)$	A flattened distribution of $p(\boldsymbol{\theta};\mathcal{D})$, defined as $\tilde{p}(\boldsymbol{\theta}\|\mathcal{D};\alpha) \propto (p(\boldsymbol{\theta})p(\mathcal{D}\|\boldsymbol{\theta}))^{\alpha}$ for $\alpha \in (0,1]$
$\widetilde{\boldsymbol{\Theta}}(\alpha)$	The enlarged candidate set at the scale α
$\tilde{\beta}(\alpha)$	The acceptance ratio in the M-H algorithm when applied to drawing samples from $\tilde{p}(\boldsymbol{\theta}\|\mathcal{D};\alpha)$
$\tilde{w}(\boldsymbol{\theta})$	The weight of a candidate $\boldsymbol{\theta} \in \widetilde{\boldsymbol{\Theta}}(\alpha)$

4.1 Candidate Set of Unlearned Parameters

Let us consider a discrete set $\boldsymbol{\Theta}$ which we refer to as the *candidate set* of unlearned model parameters. This set $\boldsymbol{\Theta}$ is constructed without the knowledge of the erased dataset $\mathcal{D}_e$. The intention in the design of $\boldsymbol{\Theta}$ is that, given an erased dataset $\mathcal{D}_e$, the parameters $\boldsymbol{\theta}_{\mathcal{D}_r}$ (unknown when $\boldsymbol{\Theta}$ is constructed) are close to a candidate in $\boldsymbol{\Theta}$.

As $\mathcal{D}_e$ is unknown during the construction of $\boldsymbol{\Theta}$, we rely on the assumption that $\mathcal{D}_e$ is relatively small compared with $\mathcal{D}$. Therefore, after unlearning the model (with parameters $\boldsymbol{\theta}_{\mathcal{D}}$) trained on $\mathcal{D}$ from $\mathcal{D}_e$, we assume that the obtained model parameters $\boldsymbol{\theta}_{\mathcal{D}_r}$ do not differ much from $\boldsymbol{\theta}_{\mathcal{D}}$. Furthermore, as $\boldsymbol{\theta}_{\mathcal{D}}$ is the MAP estimate of the model parameters given $\mathcal{D}$, $\boldsymbol{\theta}_{\mathcal{D}}$ is the mode of the posterior distribution of $\boldsymbol{\theta}$ given $\mathcal{D}$ (see Section 3.1). As a result, from our assumption that the model parameters $\boldsymbol{\theta}_{\mathcal{D}_r}$ do not differ much from $\boldsymbol{\theta}_{\mathcal{D}}$, $\boldsymbol{\theta}_{\mathcal{D}_r}$ does not differ much from the mode of the posterior distribution of $\boldsymbol{\theta}$ given $\mathcal{D}$. In other words, the posterior density $p(\boldsymbol{\theta}_{\mathcal{D}_r}|\mathcal{D})$ is sufficiently large.

Hence, in order to construct the candidate set $\boldsymbol{\Theta}$ that is likely to contain the parameters of the model unlearned from $\mathcal{D}_e$, we would like to construct $\boldsymbol{\Theta}$ as the set of model parameters $\boldsymbol{\theta}$ with high posterior probability densities $p(\boldsymbol{\theta}|\mathcal{D})$. To this end, we propose to use MCMC methods to draw samples from the posterior distribution of $\boldsymbol{\theta}$ given $\mathcal{D}$, e.g., applying Algorithm 1 with $f(\boldsymbol{\theta}) = p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$. Therefore, the candidate set $\boldsymbol{\Theta}$ is a set of samples drawn the distribution $p(\boldsymbol{\theta}|\mathcal{D})$.

4.2 MU with Candidate Set

While the constructed candidate set $\boldsymbol{\Theta}$ is likely to contain the unlearned model parameters if they do not differ much from $\boldsymbol{\theta}_{\mathcal{D}}$, we still need to construct the unlearned model parameters given this candidate set $\boldsymbol{\Theta}$ and an erased dataset $\mathcal{D}_e$.

Together with the candidate set $\boldsymbol{\Theta}$, we also store the values $h(\boldsymbol{\theta}) \triangleq \log p(\mathcal{D}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$ for all candidates $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. MCMC algorithms often require this value to evaluate the acceptance ratio (e.g., in Algorithm 1), so it does not incur additional computation to evaluate $\log p(\mathcal{D}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$. As a result, both the candidate set $\boldsymbol{\Theta}$ and these auxiliary values $h(\boldsymbol{\theta})$ can be pre-computed before the unlearning happens.

When there is a request to unlearn the trained model from an erased dataset $\mathcal{D}_e$, we make use of $h(\boldsymbol{\theta}) \triangleq \log p(\mathcal{D}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$ to efficiently evaluate the following value:

$$\begin{aligned}\log p(\boldsymbol{\theta}|\mathcal{D}_r) &= \log p(\mathcal{D}_r|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log p(\mathcal{D}_r) \\ &= h(\boldsymbol{\theta}) - \log p(\mathcal{D}_e|\boldsymbol{\theta}) - \log p(\mathcal{D}_r) = g(\boldsymbol{\theta}, \mathcal{D}_e) - \log p(\mathcal{D}_r) \quad (6)\end{aligned}$$

for $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, where $g(\boldsymbol{\theta}, \mathcal{D}_e) \triangleq h(\boldsymbol{\theta}) - \log p(\mathcal{D}_e|\boldsymbol{\theta})$ and we use the assumption that the data are conditionally independent given the model parameters (Eq. (2)) and $(\mathcal{D}_r, \mathcal{D}_e)$ is a partition of $\mathcal{D}$, i.e.,

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \log p(\mathcal{D}_r \cup \mathcal{D}_e|\boldsymbol{\theta}) = \log p(\mathcal{D}_r|\boldsymbol{\theta}) + \log p(\mathcal{D}_e|\boldsymbol{\theta}) \ .$$

In Eq. (6), $\log p(\mathcal{D}_r)$ is independent of $\boldsymbol{\theta}$, so it is treated as a constant. Besides, we note that i) we do not use $\mathcal{D}_r$ in the evaluation of $g(\boldsymbol{\theta}, \mathcal{D}_e)$ in Eq. (6); and ii) given the stored value $h(\boldsymbol{\theta}) \triangleq \log p(\mathcal{D}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$ and $\mathcal{D}_e$, the evaluation of $g(\boldsymbol{\theta}, \mathcal{D}_e)$ is efficient as we assume that $\mathcal{D}_e$ is small.

We recall that the ultimate goal of MU is to obtain the parameters that is close to the parameters $\boldsymbol{\theta}_{\mathcal{D}_r}$ of the model retrained on $\mathcal{D}_r$, i.e., the parameters that maximizes the log posterior probability $\log p(\boldsymbol{\theta}|\mathcal{D}_r)$. Note that the candidate set $\boldsymbol{\Theta}$ is likely to contain such parameters as explained in Section 4.1, and $g(\boldsymbol{\theta}, \mathcal{D}_e)$ differs with $\log p(\boldsymbol{\theta}|\mathcal{D}_r)$ by a constant $\log p(\mathcal{D}_r)$ for all $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. Therefore, we can choose the candidate in $\boldsymbol{\Theta}$ with the maximum posterior probability, given the remaining dataset by choosing the candidate $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ with the largest value of $g(\boldsymbol{\theta}, \mathcal{D}_e)$.

More importantly, we are also able to obtain an approximate distribution of the posterior distribution given the remaining dataset (i.e., $p(\boldsymbol{\theta}|\mathcal{D}_r)$) based on importance sampling. Recall that $\boldsymbol{\Theta}$ is constructed as MCMC samples from the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$, so we can assign weight $w(\boldsymbol{\theta})$ to each candidate $\boldsymbol{\theta}$ in $\boldsymbol{\Theta}$ as follows

$$w(\boldsymbol{\theta}) = \frac{p(\boldsymbol{\theta}|\mathcal{D}_r)}{p(\boldsymbol{\theta}|\mathcal{D})} = \frac{p(\mathcal{D}_r|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}\frac{p(\mathcal{D})}{p(\mathcal{D}_r)} = \frac{e^{g(\boldsymbol{\theta},\mathcal{D}_e)}}{e^{h(\boldsymbol{\theta})}}\frac{p(\mathcal{D})}{p(\mathcal{D}_r)} \quad (7)$$

where $p(\mathcal{D})/p(\mathcal{D}_r)$ is independent of $\boldsymbol{\theta}$, so it disappears after we normalize the weights for all $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. As illustrated in Figure 1, we can use this weighted set $\boldsymbol{\Theta}$ to approximate the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D}_r)$. Therefore, we can also use the weighted average of $\boldsymbol{\Theta}$ as the unlearned model parameters.

4.3 Enlarged Candidate Set

Remark 1. The main rationale behind the construction of the candidate set $\boldsymbol{\Theta}$ is that if the erased dataset $\mathcal{D}_e$ is small compared with $\mathcal{D}$, the retrained model parameters $\boldsymbol{\theta}_{\mathcal{D}_r}$ should have a sufficient high posterior probability density $p(\boldsymbol{\theta}_{\mathcal{D}_r}|\mathcal{D})$. However, in practice, by only performing MCMC sampling for the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$, we may not be able to obtain a sample that is close to $\boldsymbol{\theta}_{\mathcal{D}_r}$. Figure 2a shows a hypothetical scenario: the contour plot of a Gaussian distribution centered at $(2, 4)$ represents the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ of the model parameters given $\mathcal{D}$ while the contour plot of a Gaussian distribution centered at $(1, 1)$ represents the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D}_r)$ of the model parameters given $\mathcal{D}_r$. As the change in the posterior distribution of the model parameters after unlearning (i.e., $p(\boldsymbol{\theta}|\mathcal{D}_r)$ vs. $p(\boldsymbol{\theta}|\mathcal{D})$) is sufficiently large, we observe that samples drawn from $p(\boldsymbol{\theta}|\mathcal{D})$ (which constitute our candidate set $\boldsymbol{\Theta}$) do not overlap with the region of high

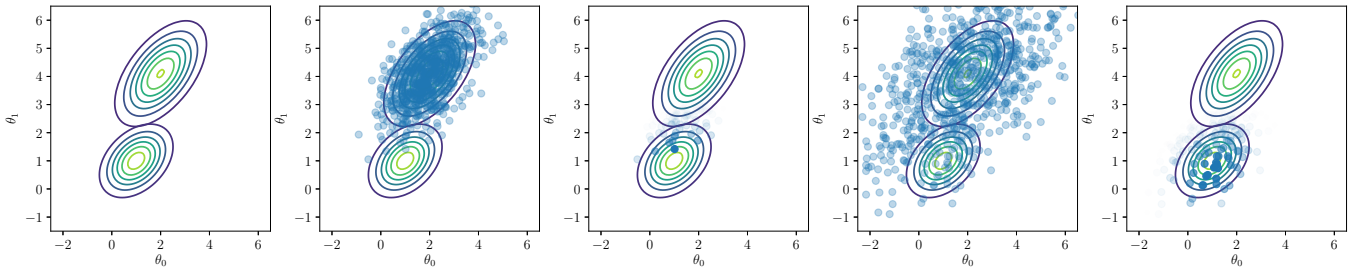


Figure 2: Plots of a hypothetical MU experiment with two parameters θ_0 and θ_1. (a) shows $p(\boldsymbol{\theta}|\mathcal{D})$ and $p(\boldsymbol{\theta}|\mathcal{D}_r)$ as Gaussian distributions centered at $(2, 4)$ and $(1, 1)$, respectively. (b) shows the samples drawn from $p(\boldsymbol{\theta}|\mathcal{D})$. (c) shows the weighted samples from those in (b) to represent $p(\boldsymbol{\theta}|\mathcal{D}_r)$. (d) shows the samples drawn from a flattened distribution $\tilde{p}(\boldsymbol{\theta}|\mathcal{D};\alpha)$. (e) shows the weighted samples from those in (d) to represent $p(\boldsymbol{\theta}|\mathcal{D}_r)$.

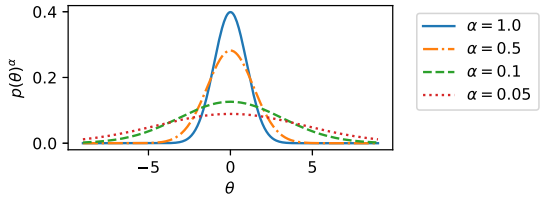


Figure 3: Distributions obtained by flattening a distribution with the density $p(\theta)$ through raising the density to a power $\alpha \in \{1.0, 0.5, 0.1, 0.05\}$ ($\alpha = 1$ does not flatten the distribution).

posterior probability $p(\boldsymbol{\theta}|\mathcal{D}_r)$ in Figure 2b. In fact, by plotting these samples/candidates with the weights from importance sampling in Eq. (7) in Figure 2c, we observe that they do not represent the distribution $p(\boldsymbol{\theta}|\mathcal{D}_r)$ well due to the lack of samples/candidates in the region of high posterior probability $p(\boldsymbol{\theta}|\mathcal{D}_r)$.

Consequently, based on Remark 1, we would like to enhance the capability of our approach to address the above issue in this section. We take the approach of enlarging the region where $\boldsymbol{\Theta}$ is constructed. It is done by sampling the candidate set $\boldsymbol{\Theta}$ from a *flattened distribution* of $p(\boldsymbol{\theta}|\mathcal{D})$, instead of from $p(\boldsymbol{\theta}|\mathcal{D})$. The flattened distribution is defined as follows.

Flattened distribution. Consider $p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta})$, we construct its flattened distribution as $\tilde{p}(\boldsymbol{\theta}|\mathcal{D};\alpha) \propto (p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta}))^\alpha$ for $\alpha \in (0, 1]$. While the exact density $\tilde{p}(\boldsymbol{\theta}|\mathcal{D};\alpha)$ of this flattened distribution is unknown, its proportional value $(p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta}))^\alpha$ is sufficient for us to draw its samples with MCMC algorithms (e.g., Algorithm 1). We call the candidate set constructed from this flattened distribution the *enlarged candidate set* at the scale α, denoted as $\widetilde{\boldsymbol{\Theta}}(\alpha)$. When $\alpha = 1$, $\tilde{p}(\boldsymbol{\theta}|\mathcal{D};\alpha = 1) = p(\boldsymbol{\theta}|\mathcal{D})$ and $\widetilde{\boldsymbol{\Theta}}(1) = \boldsymbol{\Theta}$.

The reason that raising the probability density of a distribution to a power $\alpha \in (0, 1]$ has the effect of flattening the distribution can be explained through the samples drawn from the flattened distribution with MCMC algorithms. Let us denote the acceptance ratio using the probability density raised to a power $\alpha \in (0, 1]$ as $\min(\tilde{\beta}(\alpha), 1)$ where $\tilde{\beta}(\alpha)$ is defined in the same manner as Eq. (4):

$$\tilde{\beta}(\alpha) = \left(p(\mathcal{D}|\boldsymbol{\theta}')p(\boldsymbol{\theta}')\right)^\alpha / (p(\mathcal{D}|\boldsymbol{\theta}_{i-1})p(\boldsymbol{\theta}_{i-1}))^\alpha \ .$$

We can express this acceptance ratio as a function of another acceptance ratio $\tilde{\beta}(\alpha')$ (i.e., with respect to the target distribution $\tilde{p}(\boldsymbol{\theta}|\mathcal{D};\alpha')$ for $\alpha' \in (0, 1]$).

$$\tilde{\beta}(\alpha) = \left(p(\mathcal{D}|\boldsymbol{\theta}')p(\boldsymbol{\theta}')\right)^\alpha / (p(\mathcal{D}|\boldsymbol{\theta}_{i-1})p(\boldsymbol{\theta}_{i-1}))^\alpha = \tilde{\beta}(\alpha')^{\alpha/\alpha'} \ .$$

We observe that $\tilde{\beta}$ is a decreasing function for $\alpha \in (0, 1)$ if its codomain is restricted to $(0, 1)$. For example, $\tilde{\beta}(\alpha) \geq \tilde{\beta}(\alpha')$ if $\alpha \leq \alpha'$ and $\tilde{\beta}(\alpha) \in (0, 1)$. Therefore,

$$\min(\tilde{\beta}(\alpha), 1) \geq \min(\tilde{\beta}(\alpha'), 1) \text{ if } \alpha \leq \alpha'$$

which means that the acceptance ratio in the M-H algorithm increases when α decreases. As the acceptance ratio increases, the samples drawn from the M-H algorithm cover a larger region, which is illustrated in Figure 3. A special case is when $\alpha \leq \alpha' = 1$, $\min(\tilde{\beta}(\alpha), 1) \geq \min(\tilde{\beta}(1), 1) = \min(\beta, 1)$. Thus, the samples drawn from $\tilde{p}(\boldsymbol{\theta}|\mathcal{D};\alpha)$ for $\alpha \in (0, 1)$ using MCMC algorithms cover a larger region of the domain (i.e., flattened) than those samples drawn from $p(\boldsymbol{\theta}|\mathcal{D})$.

Flattening the target distribution has also been investigated in the MCMC literature [54]. It is used to improve the exploration of the MCMC methods, e.g., to discover different modes of the target distribution. This is different our purpose which is to enlarge the candidate set such that it is close to the unlearned model parameters.

4.4 MU with Enlarged Candidate Set

Similar to Section 4.2, to construct the unlearned model parameters given the enlarged candidate set $\widetilde{\boldsymbol{\Theta}}(\alpha)$ and an erased dataset $\mathcal{D}_e$, we store the value $h(\boldsymbol{\theta}) \triangleq \log p(\mathcal{D}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$ for all candidates $\boldsymbol{\theta}$ in the enlarged candidate set $\widetilde{\boldsymbol{\Theta}}(\alpha)$. Then, we are able to obtain $g(\boldsymbol{\theta}, \mathcal{D}_e) \triangleq h(\boldsymbol{\theta}) - \log p(\mathcal{D}_e|\boldsymbol{\theta})$ that only differs from $\log p(\boldsymbol{\theta}|\mathcal{D}_r)$ by a constant independent of $\boldsymbol{\theta}$ (Eq. (6)).

However, because the enlarged candidate set $\widetilde{\boldsymbol{\Theta}}(\alpha)$ is drawn from the flattened distribution $\tilde{p}(\boldsymbol{\theta}|\mathcal{D};\alpha)$, the weights in Eq. (7) cannot be used to construct an approximation to the posterior distribution of $p(\boldsymbol{\theta}|\mathcal{D}_r)$. Therefore, we make a modification to Eq. (7) to obtain the weights for each candidate $\boldsymbol{\theta} \in \widetilde{\boldsymbol{\Theta}}(\alpha)$:

$$\tilde{w}(\boldsymbol{\theta}) = \frac{p(\boldsymbol{\theta}|\mathcal{D}_r)}{\tilde{p}(\boldsymbol{\theta}|\mathcal{D};\alpha)} = \frac{p(\mathcal{D}_r|\boldsymbol{\theta})\tilde{p}(\boldsymbol{\theta};\alpha)}{(p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}))^\alpha} \frac{\tilde{p}(\mathcal{D};\alpha)}{p(\mathcal{D}_r)} = \frac{e^{g(\boldsymbol{\theta},\mathcal{D}_e)}}{e^{\alpha h(\boldsymbol{\theta})}} \frac{\tilde{p}(\mathcal{D};\alpha)}{p(\mathcal{D}_r)}$$

where $\tilde{p}(\mathcal{D};\alpha) \triangleq \int (p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}))^\alpha \, \mathrm{d}\boldsymbol{\theta}$. Note that $\tilde{p}(\mathcal{D};\alpha)/p(\mathcal{D}_r)$ is independent of $\boldsymbol{\theta}$, so it disappears after we normalize the weights for all $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. We illustrate the enlarged candidate set $\widetilde{\boldsymbol{\Theta}}(\alpha)$ in Figure 2d and its corresponding weighted candidates in Figure 2e.

We can observe that the set of weighted candidates in the enlarged $\widetilde{\Theta}(\alpha)$ is able to approximate the posterior distribution $p(\theta|\mathcal{D}_r)$ much better than the weighted candidates in the candidate set Θ in Figure 2c. Therefore, we can also use the weighted average of $\widetilde{\Theta}(\alpha)$ as the unlearned model parameters.

5 EXPLAINING THE EFFECT OF A SUBSET OF TRAINING DATA ON MODEL PREDICTION

While MU has been mainly about removing the effect of a specific subset of training data from the model, we explore a new application of our MU approach MCU in explaining the effect of training data on the model prediction.

Let us consider a scenario that an ML model is trained to detect phishing webpages from data $\mathcal{D}$ collected from a number of sources. Let $\mathcal{D} = \cup_{i=1}^{n} \mathcal{D}_i$ where $\mathcal{D}_i$ is the data labeled by the source i. We would like to examine if there exists a source that contributes malicious/adversarial training data to the ML model. A solution is to train a model (A) on the data $\mathcal{D}$ collected from all sources. To check if a source i contributes malicious/adversarial training data $\mathcal{D}_i$, we train a separate model (B_i) on the training data $\mathcal{D} \setminus \mathcal{D}_i$ (i.e., excluding those from the source i). If the accuracy of the model B_i on a test set (different from the training data) increases significantly compared with that of the model A, then we can say that the data $\mathcal{D}_i$ labeled by the source i has a negative effect on our model, i.e., they are potentially malicious/adversarial data.

As the number of sources contributing to the training data can be large, the amount of time incurred to retrain different models B_i from scratch is prohibitively expensive. Therefore, instead of retraining from scratch, we would like to quickly estimate model B_i using model A and the corresponding data labeled by the source i. Then, obtaining model B_i in this approach can be viewed as unlearning model A from the data labeled by the source i, which is precisely the MU problem in the previous section.

Note that, as the number of sources increases, the approach of retraining becomes more and more expensive. Yet, at the same time, the data labeled by a source is likely to become smaller relative to the whole training dataset. As a result, the approach of MU becomes more practical. In Section 7.3, we demonstrate the effectiveness of MCU in explaining the positive impact of subsets of correctly labeled training data and the negative impact of subsets of incorrectly labeled training data with experiments on a phishing webpage detection dataset.

6 PITFALL: CATASTROPHIC UNLEARNING

We describe an important pitfall in MU approaches.

Remark 2 (Catastrophic unlearning)**.** It is noted that we should not unlearn a model from an erased dataset $\mathcal{D}_e$ through 'reversing' the training procedure by minimizing (instead of maximizing in the learning) the log posterior probability of the erased dataset:

$$\mathrm{argmin}_{\theta} \log p(\theta|\mathcal{D}_e) \ .$$

This is because it may lead to *catastrophic unlearning/forgetting* [24]: the log posterior probability of the remaining dataset $\mathcal{D}_r$ is decreased unnecessarily by unlearning. Although the work of [24] mitigates this issue by imposing a lower bound on $\log p(\theta|\mathcal{D}_e)$ when minimizing $\log p(\theta|\mathcal{D}_e)$, it does not propose a principled way of setting this lower bound value. Figure 4 shows a synthetic experiments with a linear regression problem. The dataset consists of tuples (x, y_x). The erased dataset $\mathcal{D}_e$ are plotted as orange dots and the remaining dataset $\mathcal{D}_r$ are plotted as blue dots. The model prediction of the model trained on $\mathcal{D}$ is shown as the dashed green line while that of the model retrained on the remaining dataset $\mathcal{D}_r$ is shown as the dashed purple line in the figure. We observe that by removing the orange dots (the erased dataset $\mathcal{D}_e$), the purple curve shifts away from the orange dots while it still fits the blue dots (the remaining dataset $\mathcal{D}_r$). However, if we unlearn the model from $\mathcal{D}_e$ by minimizing the log posterior probability of the erased dataset $\mathcal{D}_e$, the unlearned model produces entirely incorrect function values (shown as the red curves) even at the blue dots (the remaining dataset $\mathcal{D}_r$). This is the *catastrophic unlearning/forgetting* phenomenon mentioned above. Even when the unlearning (i.e., the minimization of $\log p(\theta|\mathcal{D}_e)$) is reduced by stopping after some number of iterations (e.g., after only 100 iterations), catastrophic unlearning still happens.

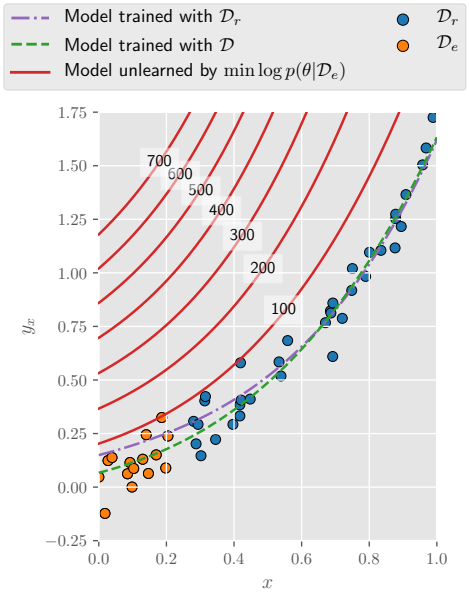


Figure 4: Illustration of the performance of the model unlearned by minimizing $p(\theta|\mathcal{D}_e)$. The red curves show the predicted function values of the unlearned models with the number of training iterations shown on the curve.

As illustrated in the above remark, the approach of minimizing $\log p(\theta|\mathcal{D}_e)$ is prone to catastrophic unlearning. There also exists approaches based on the influence function [26, 27] which performs only a small number of updates using Newton approximation. Thus, it is not prone to catastrophic unlearning as shown in our experiments in Section 7. However, the method is based on a first-order Taylor approximation, so it is only accurate for a very small change in the model after unlearning. Our experiments in Section 7 show that our proposed MCU approach outperforms this approach even when the unwanted data set is relatively small.

For MCU, catastrophic unlearning is mitigated from two angles: i) the (enlarged) candidate set restricts the unlearned model parameters to be within a region of high posterior probability density

$p(\boldsymbol{\theta}|\mathcal{D})$; ii) $g(\boldsymbol{\theta}, \mathcal{D}_e)$ is a good surrogate of the posterior probability density $p(\boldsymbol{\theta}|\mathcal{D}_r)$. The latter is because $g(\boldsymbol{\theta}, \mathcal{D}_e)$ differs from $p(\boldsymbol{\theta}|\mathcal{D}_r)$ by a constant (see Eq. (6)).

7 PERFORMANCE EVALUATION

We empirically illustrate the performance of our proposed MCU approach with a binary classification dataset, the Pima Indians diabetes dataset, and the phishing webpage detection dataset. The synthetic dataset is deliberately constructed to ease the clutter in the plots (by choosing a sufficiently small training dataset with a low input dimension) while still allowing the effect of unlearning to be easily visualized. On the other hand, the last two experiments demonstrate the empirical performance of MCU in larger real-world datasets and higher input dimensions. It is noted that we aim to design a MU algorithm given a set of MCMC samples that estimate the posterior belief of the model parameters well. Besides, it is well-known that performing MCMC sampling for high dimensional distributions is notoriously challenging [14]. Thus, we choose logistic regression models to correctly compare the unlearning performance of MCU with that of existing baselines without worrying about the performance of MCMC algorithms.

We study MCU in both cases, i) of using a candidate set $\boldsymbol{\Theta}$ and ii) of using an enlarged candidate set $\widetilde{\boldsymbol{\Theta}}(\alpha)$, to highlight the advantage of the latter approach. As explained in Section 4.3, the choice of α should depend on how much the model change after removing the erased dataset. In practice, we suggest setting aside a validation set of erased datasets to tune the value of α, i.e., selecting the value of α that the unlearned model obtained from MCU with $\widetilde{\boldsymbol{\Theta}}(\alpha)$ has the closest performance to the retrained model given an erased dataset in this validation set. For our experiments here, we choose the values of α to demonstrate the effect of flattening on MCU.

Furthermore, we compare MCU with an existing MU algorithm that is also based on MCMC samples, called *Bayesian inference forgetting* (BIF) [26, 27]. It is worth noting that, BIF utilizes the remaining dataset $\mathcal{D}_r$ in the procedure of unlearning, while MCU does not. Thus, BIF may not be feasible due to the unavailability of $\mathcal{D}_r$ or incur substantial overhead as the size of $\mathcal{D}_r$ is typically large. Therefore, we expect BIF to outperform our MCU approach in the experiments. However, due to the first-order approximation in BIF, we will observe that MCU outperforms BIF empirically in several experiments.

7.1 Synthetic Binary Classification Dataset

In this experiment, we train a logistic regression model on a binary classification dataset. We model the log ratio of the probabilities of an input x in the two classes 0 and 1 as a polynomial function: $\log\left(P(y_x = 1)/P(y_x = 0)\right) = \sum_{i=0}^{4} a_i x^i$ where $\{a_i\}_{i=0}^{4}$ are the model parameters. Equivalently, the class probabilities of an input x can be expressed as follows:

$$P(y_x = 1) = \exp\left(\sum_{i=0}^{4} a_i x^i\right) / \left(1 + \exp\left(\sum_{i=0}^{4} a_i x^i\right)\right) \quad (8)$$

and $P(y_x = 0) = 1 - P(y_x = 1)$. The prior distribution of each model parameter is a Gaussian distribution with mean 0 and variance 5.

The training dataset $\mathcal{D}$ and the erased dataset $\mathcal{D}_e$ consist of 50 and 8, respectively. Similar to the previous linear regression experiment, we deliberately choose $\mathcal{D}_e$ to be a cluster of 8 data points such that the unlearned model can be easily interpreted. We construct the candidate set and the enlarged candidate set (of the 5 model parameters $\{a_i\}_{i=0}^{4}$) using 3000 MCMC samples.

Without flattening the posterior belief $p(\boldsymbol{\theta}|\mathcal{D})$ in the construction of the candidate set $\boldsymbol{\Theta}$, Figure 5a and 5b show the model predictions of unlearned models with BIF and our MCU approach with $\alpha = 1$, respectively. We observe that the predicted probabilities $P(y_x = 1)$ of these approaches (plotted as red curves) differ from that of the model retrained on $\mathcal{D}_r$ (plotted as dashed purple curves), especially in the input region $[0.7, 1.0]$ (i.e., the location of the erased dataset $\mathcal{D}_e$). However, we also observe that these unlearning approaches are able to shift the predicted probabilities $P(y_x = 1)$ away from that of the model trained on $\mathcal{D}$ (plotted as dashed green curves) and towards that of the model retrained on $\mathcal{D}_r$ (plotted as dashed purple curves).

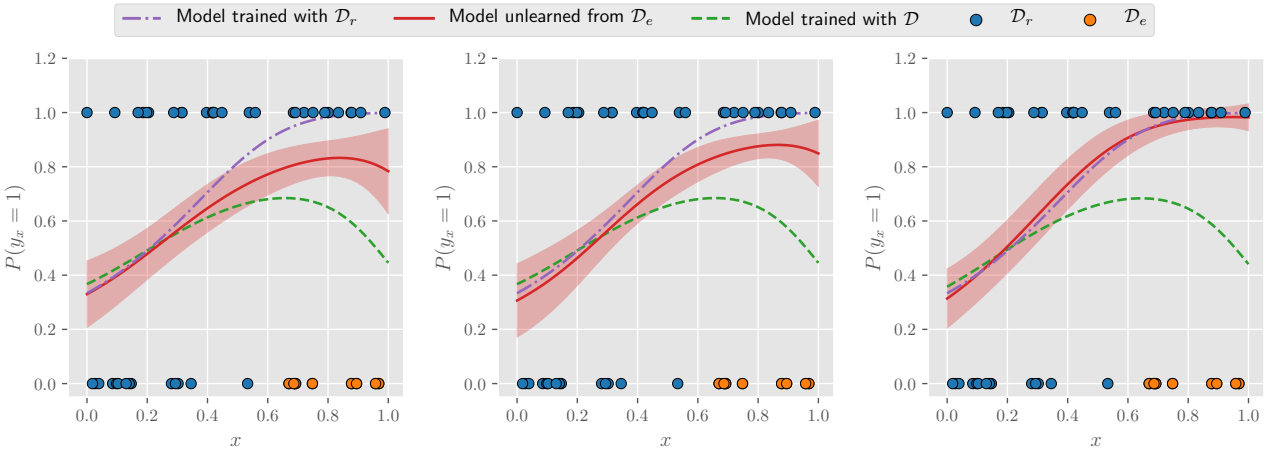
When the enlarged candidate set $\widetilde{\boldsymbol{\Theta}}(\alpha)$ is constructed by flattening the posterior belief $p(\boldsymbol{\theta}|\mathcal{D})$ with $\tilde{p}(\boldsymbol{\theta}|\mathcal{D}; \alpha = 0.1)$, the unlearning results are shown in Figure 5c. Compared with Figure 5a and 5b for unlearning results with BIF and $\alpha = 1.0$, the predicted $P(y_x = 1)$ of the unlearned model in Figure 5c (plotted as a dashed green curve) is closer to that of the model retrained on $\mathcal{D}_r$ (plotted as a dashed purple curve). The evaluation shows that our proposed MCU method with enlarged $\widetilde{\boldsymbol{\Theta}}(\alpha)$ outperforms the other two methods in this experiment. Again, MCU outperforms BIF which uses the remaining dataset in the unlearning procedure.

7.2 Pima Indians Diabetes Dataset

In this experiment, we make use of the Pima Indians diabetes dataset [25] to construct the scenario when a number of patients would like to remove their personal medical data from a classification model that is trained on their data. There are a total of 768 patients in the dataset who are Pima Indian females of age 21 and above. The record of each of them includes 8 pieces of information: the number of times that they are pregnant, the plasma glucose concentration over 2 hours in an oral glucose tolerance test, the blood pressure, the triceps skinfold thickness, the amount of 2-hour serum insulin, the body mass index, the diabetes pedigree function (based on the family history), and the age. These 8 features are used to predict whether a woman is diabetic. We use a logistic regression model for this task which has 9 parameters including the weights for the 8 features and a bias.

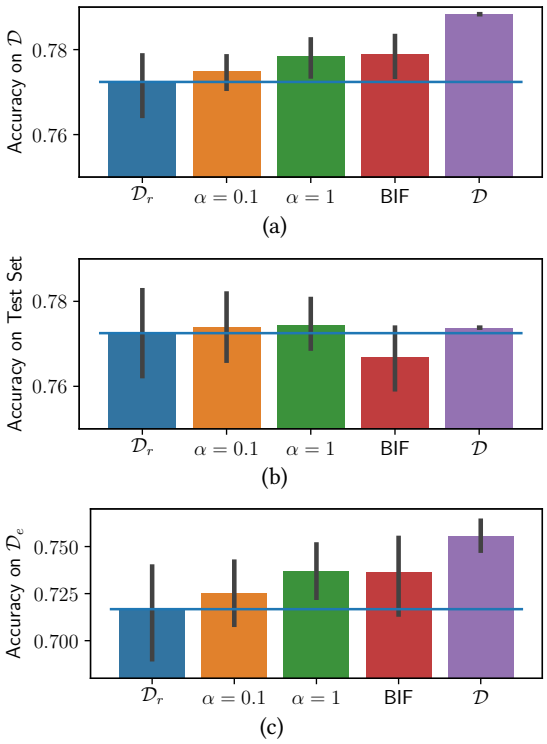
We look at a scenario where 200 patients (i.e., $|\mathcal{D}_e| = 200$) decide to withdraw their information from the trained model. Suppose that the medical data are confidential, so after training the model on the training dataset $\mathcal{D}$, only the trained model is kept (i.e., $\boldsymbol{\theta}_{\mathcal{D}}$) and the training data is not accessible anymore (except for the erased data $\mathcal{D}_e$ that are provided by the patients withdrawing their information). We use the MCU approach which utilizes $\mathcal{D}_e$ and the candidate set to remove $\mathcal{D}_e$ from the trained model.

For evaluation purposes, we reserve 178 data points from $\mathcal{D}$ to construct a test set that is not used in training the model. We select 23 different erased datasets $\mathcal{D}_e$ (each of size 200) such that removing each of them can cause a change in the model. The aim is to ease the visualization of the performance difference between unlearning methods which include MCU and BIF. The average and the 95% confidence interval of the accuracy of the unlearned models, the



(a) BIF. (b) MCU with $\alpha = 1$. (c) MCU with $\alpha = 0.1$.

Figure 5: Model prediction in experiments on the synthetic binary classification dataset.



(a) (b) (c)

Figure 6: Accuracy of the model trained on $\mathcal{D}$ (labeled as $\mathcal{D}$), retrained on $\mathcal{D}_r$ (labeled as $\mathcal{D}_r$), unlearned using BIF, MCU with $\alpha = 1$ (labeled as $\alpha = 1$) and $\alpha = 0.1$ (labeled as $\alpha = 0.1$). The accuracy is evaluated on (a) the training dataset $\mathcal{D}$, (b) the test set, and (c) the erased dataset $\mathcal{D}_e$. The bar plots show the average accuracy over 23 unlearning tasks with 23 different erased datasets $\mathcal{D}_e$. The black solid lines show the 95% confidence interval of the accuracy. The blue line shows the desirable accuracy (i.e., the accuracy of the retrained model) of the unlearned model.

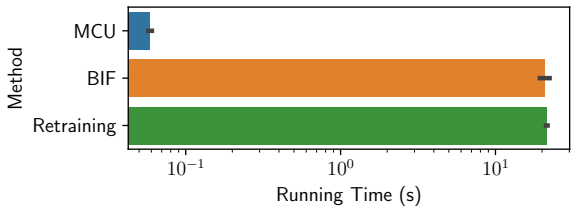


Figure 7: Running time (in seconds) for MCU, BIF, and retraining methods.

model trained on $\mathcal{D}$, and the model retrained on $\mathcal{D}_r$ over these 23 unlearning scenarios are reported.

It is important to note in this scenario, higher accuracy is not better. Instead, we want the model to 'forget' the erased data $\mathcal{D}_e$. Hence, we would like the accuracy of the unlearned model to match that of the model retrained on $\mathcal{D}_r$.

The prior distribution of each model parameter is a Gaussian distribution with mean 0 and variance 3. We construct the candidate set and the enlarged candidate set (of the 9 model parameters) using 10,000 MCMC samples.

To empirically show the unlearning performance of MCU with $\alpha = 1$ (not flattening), $\alpha = 0.1$ (flattening), and BIF, Figure 6 plots the accuracy of the unlearned models using these methods on the training dataset $\mathcal{D}$, the test set, and the erased dataset $\mathcal{D}_e$. As explained above, a good unlearning method should be able to achieve an accuracy similar to that of the model retrained on $\mathcal{D}_r$ (labeled as $\mathcal{D}_r$ in Figure 6). Intuitively, by unlearning the model trained on $\mathcal{D}$ from the erased dataset $\mathcal{D}_e$, the accuracy of the unlearned model on the training dataset $\mathcal{D}$ and the erased dataset $\mathcal{D}_e$ should decrease. It can be interpreted as the unlearned model 'forgetting' the erased dataset $\mathcal{D}_e$. This trend can be clearly observed in the case of the retrained model (by comparing the bar plot labeled as $\mathcal{D}_r$ with the bar plot labeled as $\mathcal{D}$) in Figures 6a and 6c. Between our proposed MCU methods and BIF, our MCU methods, especially the one with an enlarged candidate set using a flattened distribution with $\alpha = 0.1$, outperform BIF — their accuracies on the training dataset $\mathcal{D}$ and the erased dataset $\mathcal{D}_e$ are closer to that of the model

retrained on the remaining dataset $\mathcal{D}_r$. As for the accuracy on the test set in Figure 6b, while the unlearned model obtained with MCU is able to maintain similar accuracy to that of the retrained model, the accuracy of the unlearned model obtained with BIF drops significantly compared with the desirable accuracy of the retrained model. In Figure 7, we also observe that MCU incurs much less time than BIF and retraining methods as it does not use the remaining dataset during the unlearning procedure.

7.3 Phishing Webpage Detection Dataset

In this experiment, we consider the phishing webpage dataset [2] which is used in [41] for phishing detection with supervised learning. While the phishing URL were obtained by crawling PhishTank feed [7], the benign pages were crawled randomly from top 300,000 websites as ranked by Alexa [3]. We construct a 'clean training set', denoted as $\mathcal{D}^{(c)} = \{\mathbf{x}_i, y_i\}_{i=1}^{100000}$, of size 100,000 and a test set of size 30,127, where $y_i = 1$ if $\mathbf{x}_i$ corresponds to (the feature vector of) a phishing webpage and $y_i = 0$ otherwise. The total number of features in the processed dataset is 52. We also carry out experiments with a smaller subset of 5 features that are selected based on its importance score obtained from the trained model. A logistic regression model is employed to classify the phishing webpages.

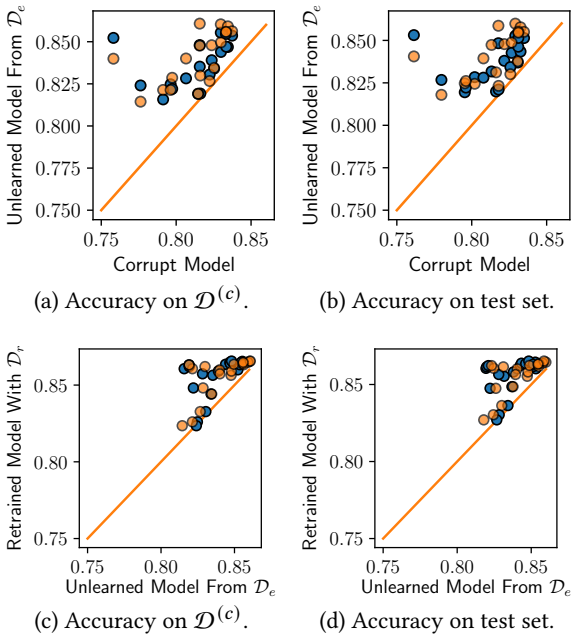
7.3.1 Unlearning Erroneous Data. We consider a scenario where the training dataset $\mathcal{D}$ contains a subset of erroneous data (i.e., the erased data $\mathcal{D}_e \subset \mathcal{D}$) that has a negative impact on the model performance. We called this training dataset the 'corrupt training dataset' to distinguish from the clean training dataset ($\mathcal{D}^{(c)}$) that contains all correct labels. The goal is to unlearn the model trained on $\mathcal{D}$ from the erased data $\mathcal{D}_e$.

To evaluate the performance of MCU, we perform experiments on 20 different corrupt datasets and their corresponding erased datasets. Each corrupt dataset $\mathcal{D}$ is constructed from the clean training dataset $\mathcal{D}^{(c)}$ as follows:

(1) A data point (i.e., a vector of feature values extracted from a webpage) $\mathbf{x}_i$ is randomly selected from the clean training dataset $\mathcal{D}^{(c)}$. Then, a set $\mathcal{N}(\mathbf{x}_i)$ is formed, which consists of 200 nearest neighbors of $\mathbf{x}_i$ in $\mathcal{D}_c$ (including $\mathbf{x}_i$). As the erased dataset $\mathcal{D}_e$ is assumed to be erroneous, $\mathcal{N}(\mathbf{x}_i)$ is chosen to form $\mathcal{D}_e$, such that each point's label in $\mathcal{D}_e$ is $1 - y$ if it is labeled as y in $\mathcal{D}^{(c)}$. In other words, all data points in $\mathcal{D}_e$ are incorrectly labeled.
(2) The remaining dataset $\mathcal{D}_r$ is defined as the largest subset of $\mathcal{D}^{(c)}$ that does not have any common data points with $\mathcal{N}(\mathbf{x}_i)$. It implies $\mathcal{D}_r \cap \mathcal{D}_e = \emptyset$.
(3) The corrupt training dataset is defined as $\mathcal{D} \triangleq \mathcal{D}_r \cup \mathcal{D}_e$ which consists of the clean data points from $\mathcal{D}_r$ and the erroneous data points from $\mathcal{D}_e$.

Each dot in Figure 8 shows the result of the experiment with one corrupt dataset. The result of BIF is shown with orange dots and that of our proposed approach MCU with $\alpha = 0.01$ is shown with blue dots. We have the following observations for the experiments on the phishing webpage detection dataset with 5 features:

- Accuracy of the unlearned model vs. that of the corrupt model: Figure 8a plots the accuracy on $\mathcal{D}^{(c)}$ of the corrupt model (i.e., the model trained on $\mathcal{D}$) against the accuracy on $\mathcal{D}^{(c)}$ of the model unlearned from $\mathcal{D}_e$. As $\mathcal{D}_e$ is the erroneous part of the $\mathcal{D}$, we expect that the accuracy of the unlearned model should improve after unlearning. This is observed in Figure 8a for both MCU with $\alpha = 0.01$ and BIF (the orange and blue dots are above the orange line where the accuracy of the corrupt model is equal to that of the unlearned model). Similarly, the same observation can be made from Figure 8b for the accuracy on the test set.
- Accuracy of the unlearned model vs. that of the retrained model: Figure 8c plots the accuracy on $\mathcal{D}^{(c)}$ of the model unlearned from $\mathcal{D}_e$ against the accuracy on $\mathcal{D}^{(c)}$ of the model retrained on $\mathcal{D}_r$. The better the unlearning approach is, the closer the accuracy of the unlearned model is to that of the naively retrained model. In Figure 8c, we observe that MCU and BIF have similar performance (note the distance between the blue and orange dots to the orange line where the accuracy of the unlearned model is equal to that of the retrained model). We have a similar observation from Figure 8d for the accuracy on the test set.

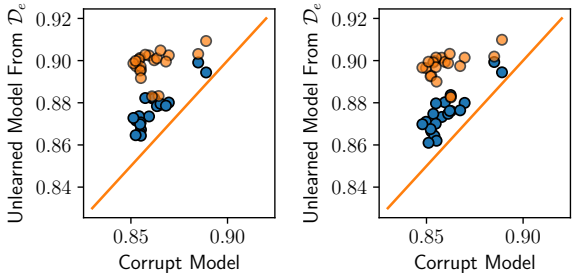


(a) Accuracy on $\mathcal{D}^{(c)}$. (b) Accuracy on test set.

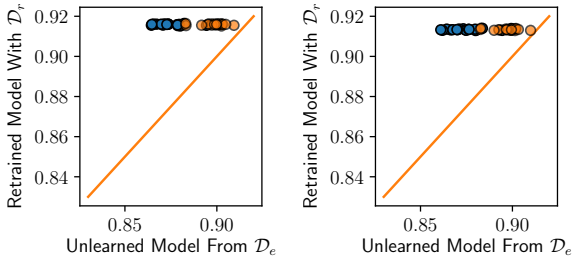
(c) Accuracy on $\mathcal{D}^{(c)}$. (d) Accuracy on test set.

Figure 8: Performances of MCU approach with $\alpha = 0.01$ (shown as blue dots) and BIF (shown as orange dots), in the experiments on the phishing webpage detection dataset with 5 features. The orange line is where the accuracy in the horizontal axis is equal to that in the vertical axis.

Figure 9 shows similar performance (comparing the accuracy of the corrupt and the retrained models on $\mathcal{D}^{(c)}$ and the test set) for the experiments on the phishing webpage detection dataset with 52 features. While BIF performance is better than MCU in this figure, recall that BIF uses the remaining dataset $\mathcal{D}_r$ in the unlearning procedure while MCU does not. Furthermore, our MCU approach demonstrates a reasonable performance as the accuracy of the unlearned model improves over that of the corrupt model (in Figures 9a and 9b).



(a) Accuracy on $\mathcal{D}^{(c)}$. (b) Accuracy on test set.



(c) Accuracy on $\mathcal{D}^{(c)}$. (d) Accuracy on test set.

Figure 9: Plots of the performances of our MCU approach with $\alpha = 0.01$ (shown as blue dots) and BIF (shown as orange dots) in the experiments on the phishing webpage detection dataset with 52 features. The orange line is where the accuracy in the horizontal axis is equal to that in the vertical axis.

7.3.2 Explaining the Effect of a Subset of Training Data on the Model Prediction. Consider another scenario where we would like to examine a number of subsets of training data to identify a subset that may have a negative impact on the model performance. These subsets may come from different sources such as different companies or different crowdworkers (see Section 5). We make use of our unlearning approach MCU to measure the effect of a subset of training data on the model prediction, by comparing the accuracy of the model before and after the unlearning procedure. We expect that the accuracy of the model after unlearning from a subset of erroneous data should improve, while the accuracy of the model after unlearning from a subset of clean data should drop.

To construct the ground truth dataset, we make use of the above corrupt training dataset $\mathcal{D}$. For a corrupt training dataset $\mathcal{D}$, the corresponding erased dataset $\mathcal{D}_e$ is the subset of the data that contains erroneous labels, i.e., $\mathcal{D}_e$ has a negative impact on the model performance. Erasing $\mathcal{D}_e$ should improve the model performance. We further draw randomly 30 subsets of the same size as $\mathcal{D}_e$ (200 data points) from $\mathcal{D}_r$. These 30 subsets contain correct labels. Erasing such subsets of data should make the model performance drop.

In Figure 10, there are 4 plots corresponding to 4 corrupt training datasets that are randomly generated. Given a corrupt training dataset, the plot shows the accuracies of models unlearned from $\mathcal{D}_e$ (shown as plus markers) and from random subsets of $\mathcal{D}_r$ (shown as purple dots). The results show that $\mathcal{D}_e$ has a negative impact on the model prediction and its removal improves accuracy (from cross to plus). On the other hand, removing correctly label data results in a drop in accuracy (from cross to dots). Furthermore, by observing

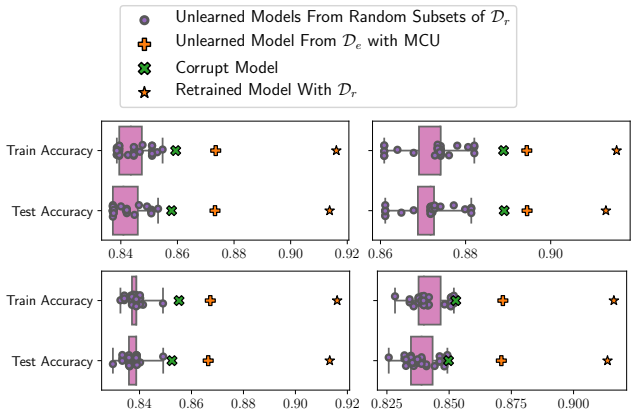


Figure 10: Plots of the accuracies of the models unlearned from 4 different $\mathcal{D}_e$ and random subsets of $\mathcal{D}_r$. They are computed on the clean training set $\mathcal{D}^{(c)}$ (labeled as train accuracy) and the test set (labeled as test accuracy) of the phishing webpage detection dataset with 52 features.

the boxplot of the accuracies of models unlearned from random subsets of $\mathcal{D}_r$ (shown as a purple box-and-whisker diagram in the plots), we observe that the accuracies of both the corrupt model and the model unlearned from $\mathcal{D}_e$ are significantly higher than that of the model unlearned from random subsets of $\mathcal{D}_r$ (the plus and cross markers are outside the boxplot).

In summary, MCU is able to identify subset of data which has a negative or positive impact on the original dataset by observing how its removal impacts accuracy. This capability of MCU is useful in detecting, identifying and (subsequently) removing adversarial data injected by an attacker.

8 CONCLUSION

We propose a new machine unlearning approach, named MCU, to remove the effect of a specific subset of training data on the trained model. It has important applications in ensuring the "right to be forgotten" in the context of user privacy, erasing a subset of malicious or adversarial data from the model, and explaining the effect of a subset of training data on the model. The approach shows promising empirical performance on both synthetic datasets and real-world datasets (the Pima Indians diabetes dataset and a phishing webpage detection dataset). As next steps, we plan to consider combining this approach with the influence function, developing an online unlearning variant [21], and also investigating these problems for unsupervised learning models.

ACKNOWLEDGEMENT

This research is supported by the National Research Foundation, Singapore under its Strategic Capability Research Centres Funding Initiative, and the National Research Foundation, Prime Minister's Office, Singapore under its Corporate Laboratory@University Scheme, National University of Singapore, and Singapore Telecommunications Ltd. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

REFERENCES

[1] 2018. *MalwareGuard: FireEye's Machine Learning Model to Detect and Prevent Malware.* https://www.fireeye.com/blog/products-and-services/2018/07/malwareguard-fireeye-machine-learning-model-to-detect-and-prevent-malware.html

[2] 2019. *Phishing webpage detection dataset.* https://github.com/JehLeeKR/phishing-madweb/

[3] [Accessed: Nov. 2021]. *Alexa top rank websites.* https://www.alexa.com/topsites

[4] [Accessed: Nov. 2021]. *Anomali.* https://www.anomali.com/marketplace/threat-intelligence-feeds

[5] [Accessed: Nov. 2021]. *APWG.* https://apwg.org/

[6] [Accessed: Nov. 2021]. *OpenPhish.* https://www.openphish.com/

[7] [Accessed: Nov. 2021]. PhishTank. https://www.phishtank.com/

[8] [Accessed: Nov. 2021]. *URLhaus.* https://urlhaus.abuse.ch/api/

[9] Sahar Abdelnabi, Katharina Krombholz, and Mario Fritz. 2020. VisualPhishNet: Zero-Day Phishing Website Detection by Visual Similarity. In *Proc. ACM CCS.* 1681–1698.

[10] Blake Anderson and David McGrew. 2016. Identifying Encrypted Malware Traffic with Contextual Flow Data. In *Proc. ACM Workshop on Artificial Intelligence and Security (AISec '16).* 35–46.

[11] Blake Anderson, Daniel Quist, Joshua Neil, Curtis Storlie, and Terran Lane. 2011. Graph-based malware detection using dynamic analysis. *Journal in computer Virology* 7, 4 (2011), 247–258.

[12] Hyrum S. Anderson and Phil Roth. 2018. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. *CoRR* abs/1804.04637 (2018). arXiv:1804.04637 http://arxiv.org/abs/1804.04637

[13] Dana Angluin and Philip Laird. 1988. Learning from noisy examples. *Machine Learning* 2, 4 (1988), 343–370.

[14] Andrei-Cristian Barbos, Francois Caron, Jean-François Giovannelli, and Arnaud Doucet. 2017. Clone MCMC: Parallel High-Dimensional Gaussian Gibbs Sampling. In *In Proc. NeurIPS*, Vol. 30. Curran Associates, Inc.

[15] Simon Bell and Peter Komisarczuk. 2020. An Analysis of Phishing Blacklists: Google Safe Browsing, OpenPhish, and PhishTank. In *Proc. Australasian Computer Science Week Multiconference.*

[16] Michael Betancourt. 2017. A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434* (2017).

[17] Lucas Bourtoule, Varun Chandrasekaran, Christopher Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2019. Machine Unlearning. *arXiv preprint arXiv:1912.03817* (2019).

[18] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. 2011. *Handbook of Markov chain Monte Carlo.* CRC press.

[19] Yinzhi Cao and Junfeng Yang. 2015. Towards making systems forget with machine unlearning. In *Proc. IEEE Symposium on Security and Privacy.* 463–480.

[20] Tianqi Chen, Emily Fox, and Carlos Guestrin. 2014. Stochastic gradient Hamiltonian Monte Carlo. In *Proc. ICML.* 1683–1691.

[21] Yizhou Chen, Shizhuo Zhang, and Bryan Kian Hsiang Low. 2022. Near-Optimal Task Selection for Meta-Learning with Mutual Information and Online Variational Bayesian Unlearning. In *Proc. AISTATS.*

[22] Adam D Cobb and Brian Jalaian. 2020. Scaling Hamiltonian Monte Carlo inference for Bayesian neural networks with symmetric splitting. *arXiv preprint arXiv:2010.06772* (2020).

[23] R Dennis Cook and Sanford Weisberg. 1982. *Residuals and influence in regression.* New York: Chapman and Hall.

[24] Min Du, Zhi Chen, Chang Liu, Rajvardhan Oak, and Dawn Song. 2019. Lifelong Anomaly Detection Through Unlearning. In *Proc. ACM CCS.* 1283–1297.

[25] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[26] Shaopeng Fu, Fengxiang He, and Dacheng Tao. 2021. Bayesian inference forgetting. *arXiv preprint arXiv:2101.06417* (2021).

[27] Shaopeng Fu, Fengxiang He, and Dacheng Tao. 2022. Knowledge removal in sampling-based Bayesian inference. In *Proc. ICLR.* (Part of Shaopeng Fu, Fengxiang He, and Dacheng Tao. "Bayesian inference forgetting.").

[28] Sanjam Garg, Shafi Goldwasser, and Prashant Nalini Vasudevan. 2020. Formalizing Data Deletion in the Context of the Right to be Forgotten. *arXiv preprint arXiv:2002.10635* (2020).

[29] GDPR.EU. 2016. Everything you need to know about the "Right to be forgotten". https://gdpr.eu/right-to-be-forgotten

[30] Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. 2019. Making AI forget you: Data deletion in machine learning. In *Advances in Neural Information Processing Systems.* 3513–3526.

[31] Jacob Goldberger and Ehud Ben-Reuven. 2017. Training deep neural-networks using a noise adaptation layer. In *Proc. ICLR.*

[32] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[33] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens van der Maaten. 2019. Certified Data Removal from Machine Learning Models. *arXiv preprint arXiv:1911.03030* (2019).

[34] W Keith Hastings. 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* 57, 1 (1970), 97–109.

[35] Matthew D Hoffman, Andrew Gelman, et al. 2014. The no-u-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.* 15, 1 (2014), 1593–1623.

[36] Michael Kearns. 1998. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)* 45, 6 (1998), 983–1006.

[37] Pang Wei Koh, Kai-Siang Ang, Hubert Teo, and Percy S Liang. 2019. On the accuracy of influence functions for measuring group effects. In *Proc. NeurIPS.*

[38] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proc. ICML.* 1885–1894.

[39] Hung Le, Quang Pham, Doyen Sahoo, and Steven CH Hoi. 2018. URLNet: Learning a URL representation with deep learning for malicious URL detection. *arXiv preprint arXiv:1802.03162* (2018).

[40] Jehyun Lee, Farren Tang, Pingxiao Ye, Fahim Abbasi, Phil Hay, and Dinil Mon Divakaran. 2021. D-Fence: A Flexible, Efficient, and Comprehensive Phishing Email Detection System. In *IEEE European Symposium on Security and Privacy (IEEE EuroS&P).* 578–597.

[41] Jehyun Lee, Pingxiao Ye, Ruofan Liu, Dinil Mon Divakaran, and Mun Choon Chan. 2020. Building robust phishing detection system: an empirical analysis. In *NDSS Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb).*

[42] Yun Lin, Ruofan Liu, Dinil Mon Divakaran, Jun Yang Ng, Qing Zhou Chan, Yiwen Lu, Yuxuan Si, Fan Zhang, and Jin Song Dong. 2021. Phishpedia: A Hybrid Deep Learning Based Approach to Visually Identify Phishing Webpages. In *30th USENIX Security Symposium.*

[43] Samuel Marchal, Kalle Saari, Nidhi Singh, and N Asokan. 2016. Know your phish: Novel techniques for detecting phishing sites and their targets. In *Proc. IEEE 36th International Conference on Distributed Computing Systems (ICDCS).* 323–333.

[44] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. 1953. Equation of state calculations by fast computing machines. *The journal of chemical physics* 21, 6 (1953), 1087–1092.

[45] Tyler Moore and Richard Clayton. 2008. Evaluating the Wisdom of Crowds in Assessing Phishing Websites. In *Financial Cryptography and Data Security.* 16–30.

[46] Radford M Neal et al. 2011. MCMC using Hamiltonian dynamics. In *Handbook of Markov chain Monte Carlo.* CRC Press, Chapter 5.

[47] Ido Nevat, Dinil Mon Divakaran, Sai Ganesh Nagarajan, Pengfei Zhang, Su Le, Ko Li Ling, and Vrizlynn Thing. 2018. Anomaly Detection and Attribution in Networks With Temporally Correlated Traffic. *IEEE/ACM Transactions on Networking* 26, 1 (Feb. 2018), 131–144.

[48] Quoc Phong Nguyen, Kar Wai Lim, Dinil Mon Divakaran, Kian Hsiang Low, and Mun Choon Chan. 2019. GEE: A gradient-based explainable variational autoencoder for network anomaly detection. In *Proc. IEEE Conf. on Commun. and Network Security (CNS).* 91–99.

[49] Quoc Phong Nguyen, Bryan Kian Hsiang Low, and Patrick Jaillet. 2020. Variational Bayesian unlearning. In *Proc. NeurIPS.*

[50] Adam Oest, Penghui Zhang, Brad Wardman, Eric Nunes, Jakub Burgis, Ali Zand, Kurt Thomas, Adam Doupé, and Gail-Joon Ahn. 2020. Sunrise to Sunset: Analyzing the End-to-end Life Cycle and Effectiveness of Phishing Attacks at Scale. In *29th USENIX Security Symposium.* 361–377.

[51] Lucky Onwuzurike, Enrico Mariconti, Panagiotis Andriotis, Emiliano De Cristofaro, Gordon Ross, and Gianluca Stringhini. 2019. MaMaDroid: Detecting Android Malware by Building Markov Chains of Behavioral Models (Extended Version). *ACM Trans. Priv. Secur.* 22, 2, Article 14 (April 2019).

[52] G. Patrini, A. Rozza, A. Krishna Menon, R. Nock, and L. Qu. 2017. Making deep neural networks robust to label noise: a loss correction approach. In *Proc. CVPR.* 1944–1952.

[53] PHISHLABS. 2021. Most Phishing Attacks Use Compromised Domains and Free Hosting. https://www.phishlabs.com/blog/most-phishing-attacks-use-compromised-domains-and-free-hosting/

[54] Christian P Robert, Víctor Elvira, Nick Tawn, and Changye Wu. 2018. Accelerating MCMC algorithms. *Wiley interdisciplinary reviews: computational statistics* 10, 5 (2018), e1435.

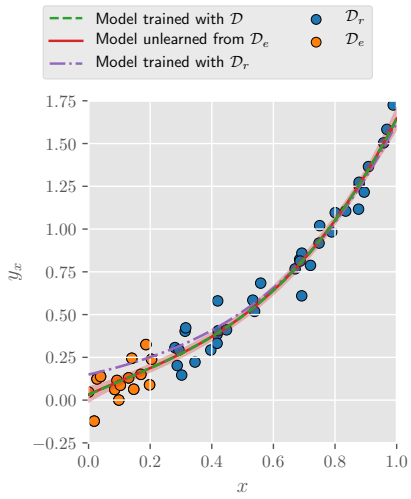
[55] Joshua Saxe and Konstantin Berlin. 2015. Deep neural network based malware detection using two dimensional binary program features. In *Proc. 10th International Conference on Malicious and Unwanted Software (MALWARE).* 11–20.

[56] Giorgio Severi, Jim Meyer, Scott Coull, and Alina Oprea. 2021. Explanation-Guided Backdoor Poisoning Attacks Against Malware Classifiers. In *30th USENIX Security Symposium.*

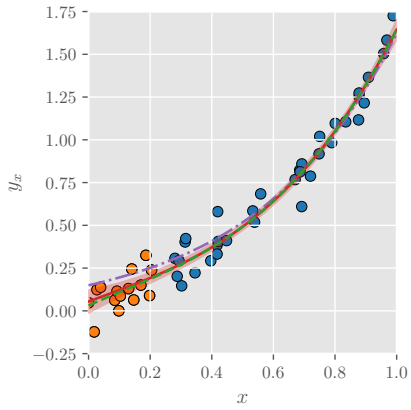
[57] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

[58] Eduard Fosch Villaronga, Peter Kieseberg, and Tiffany Li. 2018. Humans forget, machines remember: Artificial intelligence and the Right to Be Forgotten. *Computer Law & Security Review* 34, 2 (2018), 304–313.

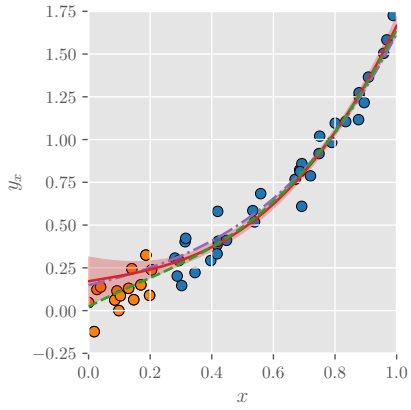
[59] Max Welling and Yee W Teh. 2011. Bayesian learning via stochastic gradient Langevin dynamics. In *Proc. ICML.* 681–688.



(a) Model prediction from unlearned model obtained by BIF.



(b) Model prediction from unlearned model obtained by MCU with $\alpha = 1$.



(c) Model prediction from unlearned model obtained by MCU with $\alpha = 0.08$.

Figure 11: Model prediction obtained from (a) BIF and MCU with (b) $\alpha = 1$ and (c) $\alpha = 0.08$ in experiments with the synthetic linear regression dataset.

[60] Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. 2020. Cyclical stochastic gradient MCMC for Bayesian deep learning. In *Proc. ICLR*.

APPENDIX

Synthetic Linear Regression

In this experiment, we would like to fit a polynomial curve $y_x = \sum_{i=0}^{4} a_i x^i$ to a linear regression dataset where the output is perturbed by a Gaussian noise of mean 0 and variance 0.01. The prior distribution of each model parameter is a Gaussian distribution of mean 0 and variance 4.

The training dataset $\mathcal{D}$ and the erased dataset $\mathcal{D}_e$ consist of 50 and 15 data points, respectively. We observe that unlearning the model from 15 data points randomly selected in $\mathcal{D}$ often does not change the model much, i.e., the difference between $\boldsymbol{\theta}_{\mathcal{D}}$ and $\boldsymbol{\theta}_{\mathcal{D}_r}$ is small. Therefore, we deliberately choose $\mathcal{D}_e$ to be a cluster of 15 data points with small values of x as shown in Figure 11. We construct the candidate set and the enlarged candidate set (of the 5 model parameters $\{a_i\}_{i=0}^{4}$) using 3000 MCMC samples.

When the enlarged candidate set $\widetilde{\boldsymbol{\Theta}}(\alpha)$ is constructed by flattening the posterior belief $p(\boldsymbol{\theta}|\mathcal{D})$ with $\tilde{p}(\boldsymbol{\theta}|\mathcal{D}; \alpha = 0.08)$, the unlearning results are shown in Figure 11c. Compared with Figure 11a and 11b for unlearning by BIF and MCU with $\alpha = 1.0$, the model prediction of the unlearned model in Figure 11c (plotted as a dashed green curve) is similar to that of the model retrained on $\mathcal{D}_r$ (plotted as a dashed purple curve). Furthermore, we can observe that the uncertainty of the model prediction increases at the erased data (i.e., $x \in (0.0, 0.2)$) after unlearning. It can be explained by the fact that the training data in this region are erased. As a result, we observe that our proposed MCU method with enlarged $\widetilde{\boldsymbol{\Theta}}(\alpha)$ outperforms the other two methods in this experiment. Notably, while not using the remaining dataset, MCU with an enlarged candidate set outperforms BIF which uses the remaining dataset in the unlearning procedure.