Universitat de Barcelona

Fundamental Principles of Data Science Master's Thesis

Regularization-Based Machine Unlearning

Author:
Arnau Jutglar Puig

Supervisors:
Nahuel Statuto, Julio C. S. Jacques Junior

A thesis submitted in partial fulfillment of the requirements for the degree of MSc in Fundamental Principles of Data Science

in the

Facultat de Matemàtiques i Informàtica

June 30, 2025

UNIVERSITAT DE BARCELONA

Abstract

Facultat de Matemàtiques i Informàtica

MSc

Regularization-Based Machine Unlearning

by Arnau Jutglar Puig

This work treats the unlearning problem in machine learning (ML). This is the process to make ML models forget some subset of their training data. We restrict this study to deep learning architectures.

We propose a metric to assess different unlearning algorithms. We design a new unlearning algorithm, *Regret*, and compare its performance with respect to *Fine-tuning* and our implementation of *Fanchuan*. We test them on four datasets and two different architectures.

The experiments reveal that *Regret* outperforms *Fine-tuning* by a small margin. Moreover, our implementation of *Fanchuan* is the best-performing algorithm and surpasses the other two clearly.

Contents

Acknowledgements

Vull expressar el meu agraïment a l'Eudald i al Marc pel suport que m'han donat mentre desenvolupava aquest treball. També vull donar les gràcies a en Julio Jacques i a en Nahuel Statuto pels seus consells i guia en aquesta feina.

Chapter 1

Introduction

In this section we explain what *unlearning* (cf. [7]) means in the context of machine learning and why it is relevant. We proceed to justify what motivates this work and what are we trying to achieve: a performant unlearning algorithm that does not rely on an initial *erase* [1] phase. We end by stating our hypothesis and clarifying the purpose of this work.

1.1 Problem definition

In the last ten years, there has been a surge in the usage of Deep Learning techniques to develop artificial intelligence tools. These artifacts rely on heavy loads of data collected from various sources, some of them may be sensitive to privacy issues related to their owners. For instance, an image processing model might have been trained on photographs involving minors obtained automatically, and their parents may subsequently request their removal. A generative AI model might have been trained on works of art of an artist which afterwards wants its works to not be present in the model's database anymore. These are just some examples that lead to consider the unlearning problem on neural networks.

But why *unlearning*? Of course if some data needs to be removed from the training set, one could always just do it and then train the model again on this retain set. However, for large models, this is unfeasible both for time and economical constraints. For instance, OpenAI's GPT-3 model's training required 3.64×10^3 PF-days of computational effort and is estimated to have cost between 2 and 4 million dollars (see [3]). Then, it is evident that some method is needed to make the models look as if they had been retrained with only the retain set but at a fraction of cost. *Unlearning* in the context of deep learning refers to this process.

1.2 Motivation

Most state of the art unlearning algorithms[2], as Fanchuan (cf. Chapter 2 §2.4.1), Kookmin (cf. [10]) or Seif (cf. [10]), rely on an initial phase of destruction followed by a repair one. This has the drawback that the models are operational only after some repair steps have been made and the network's weights have converged again. If the removal of some training samples were a legal requirement for the model to be kept up and running and the model was large enough, this would be a handicap for its developers, meaning that they would not have it usable after an extended period and a large cost.

[1] We explain what *erase* phase means in Chapter 2, §2.4

[2] Existing approaches will be explained in further detail in Chapter 2, §2.4.

The Fine-tuning algorithm (cf. Chapter 2, §2.3.2) produces functional models at every step of the process. But it is slower. In contrast, the algorithm we are going to propose does not rely on destroy-and-repair, and we designed it with the intent that it is faster than Fine-tuning. This is why we consider it worthy of investigation.

1.3 Work purpose and hypothesis

We define a new unlearning alogirthm. Our hypothesis is that this system works better than the Fine-tuning schema and can be generalized to complex neural network models. If this were the case, given the fact that it is not based on a prior erase phase, it could be of further interest in the realm of deep learning unlearning. The purpose of this work is to contrast our hypothesis.

1.4 Thesis organization

This thesis is organized around four main chapters plus the conclusions. In Chapter 2, we analytically define the *unlearning* concept. Later, we expose some approaches of other studies to assess the fitness of unlearning algorithms. We end this Chapter by reviewing some important unlearning algorithms and explaining in detail the two that we will use to compare our proposal against.

In Chapter 3, we introduce and motivate our unlearning algorithm. This is followed by the definition of the metrics that we will use to evaluate the models during our experiments. Chapter 4 sets the methodology that will define the framework under which the different unlearning models will be evaluated. It thoroughly explains and justifies the decisions that we have taken in this regard.

Finally, Chapter 5 reports the experiments that were conducted. For each of them, it introduces the dataset, the model architecture, the baselines against which each algorithm was assessed and the final evaluation. The main matter ends with Chapter 6, in which we report the final conclusions of this project and introduce some relevant discussions that arose during the development of this thesis.

The thesis is complemented with three appendices. Appendix A details the optimization process that we conducted for each algorithm before the final comparison. Appendix B recovers some additional experiments that were performed to do the optimization process with the due diligence but that did not alter the original parameters choice. Finally, Appendix C informs how to access the source code developed for this work.

Chapter 2

Related work

This chapter formally defines machine unlearning. We introduce different approaches to measuring unlearning that are treated in previous studies. Two basic unlearning mechanisms are explained. This section ends with a detailed explanation of the Fanchuan unlearning algorithm.

2.1 Analytical definition of unlearning

We devote this section to explaining the basic setting of a machine learning problem and the subsequent unlearning process.

Definition 1 *Let $\mathcal{X}$ be a set of possible datasets, and $\mathcal{M}$ a family of models. A machine learning algorithm is a map $\mathcal{L} : \mathcal{X} \to \mathcal{M}$.*

Although technically $\mathcal{M}$ is a family of models and a particular element $M \in \mathcal{M}$ is a model, we often refer to $\mathcal{M}$ as the model and say that once M is determined by $\mathcal{L}(X)$, the model has *learned* $X \in \mathcal{X}$.

The definition of an unlearning process is a little more complicated.

Definition 2 *Let $X \in \mathcal{X}$ be a dataset and $M_X = \mathcal{L}(X)$ be the resulting model. Let $U \subset X$ a subset of X which we will call the* forget set*, and $X \setminus U$ the* retain set*. Finally, let $M_{X\setminus U} = \mathcal{L}(X \setminus U)$ be the model that has learned the retain set.*

An unlearning algorithm is a map $\mathcal{U} : (X, U, M_X) \mapsto M'_{X\setminus U}$ with the intent that $M_{X\setminus U} \sim M'_{X\setminus U}$.

In this work, we will refer to M_X as the original model*, $M_{X\setminus U}$ the* baseline model *and $M'_{X\setminus U}$ the* unlearned model*.*

Note that the unlearning algorithm gets as input the dataset, the forget set and the learned model [1]. This definition relies on a similarity notion between models that can be defined in various ways.

2.2 Approaches to measure unlearning

We will restrict the study of machine unlearning to deep learning models. Then, the learning process is not deterministic. It depends on weights initialization, that tends to be random and tends to rely on optimization algorithms that are variants of stochastic gradient descent. Then, the algorithm defines a probability law over $\mathcal{M}$.

[1] In general terms, the unlearning algorithm is a function on all the data plus the subset that we want to remove. However, each particular algorithm may depend on different subsets of this data. For instance, some models depend only on the retain set, while others also require the forget set.

Conversely, for unlearning algorithms we have similar uncertainty concerns. There are two main sources of uncertainty here: the original model and the optimization algorithm used in the unlearning process. Overall, this defines a new probability law over $\mathcal{M}$.

The fitness of an unlearning algorithm depends on the similarity between the two distributions $P(\mathcal{L}(X \setminus U) \in V)$ and $P(\mathcal{U}(X, U, \mathcal{L}(X)) \in V)$ for any $V \in \mathcal{M}$. Aiming to match the probability distribution $P(\mathcal{U}(X, U, \mathcal{L}(X)))$, known as *exact* machine unlearning (cf. [7]), is very challenging, which leads to the consideration of *approximate* machine unlearning. In approximate machine unlearning, we intend to get a distribution which is *close* to the distribution obtained by training from zero on the retain dataset.

Nonetheless, critical to the study of machine unlearning is that it is very difficult to study these distributions. First, they are prone to have no known analytical expression. Secondly, they are supported on very high-dimensional manifolds. For instance, a simple deep learning architecture to work with very small images may easily contain a number of parameters of the order of 10^6. Studying a distribution over $\mathbb{R}^{100000}$ is highly complex. Thirdly, the direct simulation of the distribution by taking a high number of samples is very computationally expensive since each sample requires training a model. Note that the weights determine the output, but the converse is not true in general.

In the light of the previously exposed complexity, several studies introduce different evaluation frameworks. We list next some approaches.

- **l^2 distance**: Based on measuring the distribution of the l^2 distance between the weights of $\mathcal{U}(X, U, \mathcal{L}(X))$ and $\mathcal{L}(X \setminus U)$ (cf. [13]).

- **KL divergence**: Based on computing empirical approximations of the distributions of $P(\mathcal{L}(X \setminus U))$ and $P(\mathcal{U}(X, U, \mathcal{L}(X)))$ and then using KL-divergence as a metric over the distributions space (cf. [6]).

- **(ε, δ)-unlearning on the weights**: Establishes a uniform affine bound between the two distributions $P(\mathcal{L}(X \setminus U))$ and $P(\mathcal{U}(X, U, \mathcal{L}(X)))$ (cf. [7]). Fixed a dataset X, a forget set U and a learning algorithm $\mathcal{L}$, an unlearning algorithm $\mathcal{U}$ is (ε, δ)-unlearning with respect to the weights if $\forall V \subset \mathcal{M}$

$$P(\mathcal{U}(X, U, \mathcal{L}(X)) \in V) \leq e^{\varepsilon} P(\mathcal{L}(X \setminus U) \in V) + \delta, \qquad \text{and}$$
$$P(\mathcal{L}(X \setminus U) \in V) \leq e^{\varepsilon} P(\mathcal{U}(X, U, \mathcal{L}(X)) \in V) + \delta$$

- **(ε, δ)-unlearning on the outputs**: Once a model is trained, it defines a function from the input space to the output space. Let W be, a measurable subset of functions from the input space to the output space, defined by the values they take, not by its parameters (cf. [8]). A model M belongs to W if it takes the same values as another element of W, ignoring the weights of M and the functional expression of such element. Fixed a dataset X, a forget set U and a learning algorithm $\mathcal{L}$; an unlearning algorithm $\mathcal{U}$ is (ε, δ)-unlearning with respect to the output if $\forall W$

$$P(\mathcal{U}(X, U, \mathcal{L}(X)) \in W) \leq e^{\varepsilon} P(\mathcal{L}(X \setminus U) \in W) + \delta, \qquad \text{and}$$
$$P(\mathcal{L}(X \setminus U) \in W) \leq e^{\varepsilon} P(\mathcal{U}(X, U, \mathcal{L}(X)) \in W) + \delta$$

- **Score based**: Some works construct a score based on the model's outputs. They compare different unlearning algorithms based on the probability distribution for the score that they generate (cf. [10]). This score may incorporate a comparison on the accuracy of the baseline and the unlearned model as long as a forget quality measure. This is the approach that we will take later in this work since its compromise between information and computational complexity matches our resources availability.

2.3 Existent unlearning algorithms

2.3.1 Retraining on the retain set

Retraining the model from scratch on the retain set can be seen as an unlearning model. It is involved in the evaluation of other unlearning algorithms since it serves as a baseline. It is an exact machine unlearning algorithm by definition, but it is naive since the whole information the model had learned is lost and has to retrain from zero. The computational saving of any unlearning algorithm is seen as compared to this option.

2.3.2 Fine-tuning

We refer by *Fine-tuning* to the unlearning algorithm consisting on doing extra epochs with the same loss as used in the training step but only with the retain set. Despite there is no explicit pushing towards changing the model performance on the forget set, the inherent inductive bias of neural networks yields the model to *simplify* its predictions on such subset in a manner that has been experimentally proven to be close to how the model would behave without having seen it. The drawback of this method is that it does not rely on anything else than on this inductive bias and thus may not be fast enough.

2.4 State of the art models

In 2023, Google Research and collaborators organized the first machine unlearning challenge (cf. [10]). This competition was hosted on Kaggle. Participants were given a pretrained ResNet-18 age prediction model based on face images. Their goal was to propose unlearning algorithm to make the model forget a given forget set. The proposals were evaluated based on three key criteria. First, models were assessed according to their utility, which was measured by comparing their accuracy (both on retrain and test set) with respect to that one of the baseline. Secondly, the forget quality was measured by estimating the ε value defined as in §2.2. Finally, efficiency took its role by refusing any submission that exceeded 20% of the time it took to retrain the model on the retain set. For a detailed explanation of how this score was produced, please refer to [10].

We now proceed to explain the unlearning strategy which attained the top position in this unlearning challenge, Fanchuan. For two more examples on state of the art unlearning algorithms that rely on an erase phase followed by a repair one, see Kookmin and Seif (cf. [10]).

2.4.1 Fanchuan

The Fanchuan unlearning algorithm was the winner of the Google competition. For the first epoch, it focuses on minimizing the KL-divergence between its predictions on the forget set U and the uniform distribution. Formally, if the predictions lie on the manifold Y: $\min \mathrm{KL}[M(x_U), U(Y)]$ where x_U can be seen as random variable taking per values batches of U uniformly.

Afterwards, it spends the remaining epochs iteratively applying the following two steps. Firstly, maximizing the contrastive loss between its predictions for batches of the forget set and the retain set: $\max \text{contrastive}(M(x_U), M(x_R))$, $x_R \subset X \setminus U$. Secondly, minimizing cross entropy loss over the retain set between its predictions and the real ones: $\min \mathrm{MSE}(M(x_R), y)$.

Algorithm 1 Fanchuan

Input: Forget dataset U, retain dataset $X \setminus U$, original model weights θ, original model M_θ, temperature coefficient t, KL divergence learning rate α_1, contrastive loss learning rate α_2, cross entropy learning rate α_3, number of epochs N, batch size B.
Output: unlearned model.
for x, y in U (batches of size B) **do** ▷ KL divergence step.
    $y_{\text{pred}} \leftarrow M_\theta(x)$
    $\mathcal{L} \leftarrow \mathrm{KL}(y_{\text{pred}}, \text{uniform})$
    $\theta \leftarrow \theta - \alpha_1 \nabla \mathcal{L}$
end for
for epoch in $\{1, ..., N\}$ **do**
    for x_u, y_u in U and x_r, y_r in $X \setminus U$ (batches of size B) **do**
        $y_{\text{pred-}u} \leftarrow M_\theta(x_u)$ ▷ Contrastive loss step.
        $y_{\text{pred-}r} \leftarrow M_\theta(x_r)$
        $\mathcal{L} \leftarrow \text{mean}(\text{LogSoftmax}(\frac{1}{t} y_{\text{pred-}u} \cdot y_{\text{pred-}r}^T))$
        $\theta \leftarrow \theta + \alpha_2 \nabla \mathcal{L}$
    end for
    for x, y in $X \setminus U$ (batches of size B) **do** ▷ Cross entropy step.
        $y_{\text{pred}} \leftarrow M_\theta(x)$
        $\mathcal{L} \leftarrow \mathrm{CE}(y_{\text{pred}}, y)$
        $\theta \leftarrow \theta - \alpha_3 \nabla \mathcal{L}$
    end for
end for
return M_θ

Chapter 3

Proposed model

In this chapter, we detail the unlearning algorithm we are proposing and visually explaining the intuition behind its construction. Next, we introduce the metrics we have defined and we will use to later assess and compare this model.

3.1 Our proposal: the Regret algorithm for neural networks

Our unlearning algorithm consists on performing several epochs aimed at minimizing the following loss:

$$\mathrm{CE}(M(x), y) + Ce^{-\|\theta_0 - \theta\|^2}$$

over the retain set. We denote by θ_0 the initial weights and by θ the current weights. Both θ and θ_0 can be chosen as the weights of all the layers or only of some of them. The first term of this loss makes the model to keep the parameters that minimize the cross entropy on the retain set, so to keep the parameters that make good predictions on it. However, the second term penalizes the parameters that are close to the original ones. Therefore, our intention is for the parameters to move away from the original ones (the ones that kept information of both X and U) to ones that keep doing good predictions on $X \setminus U$ but that are different to the previous ones.

It is designed for neural networks. It depends on the C parameter and on the choice of layers. Two major interesting parts of this construction are the following. First, it does not need an initial *erase* phase, so one has functional models throughout all the unlearning process. And second, it only needs the retain set, not the forget. And therefore, the forget set can be removed from the database whithout having to wait for the model to have successfully unlearned. See algorithm 2 for the pseudocode explanation of the algorithm.

Algorithm 2 Regret

Input: Retain dataset $X \setminus U$, original model's weights θ, original model M_θ, parameter C, learning rate α, number of epochs N, batch size B.
Output: unlearned model.
$\theta_0 \leftarrow \theta$
for epoch in $\{1, ..., N\}$ **do**
    for x, y in $X \setminus U$ (batches of size B) **do**
        $l_1 \leftarrow \text{CE}(M_\theta(x), y)$
        $l_2 \leftarrow C \cdot e^{\|\theta_0 - \theta\|_2}$
        $\mathcal{L} \leftarrow l_1 + l_2$
        $\theta \leftarrow \theta - \alpha \nabla \mathcal{L}$
    end for
end for
return M_θ

To get a visual intuition on the idea behind this construction, let's observe the following case. We present a very simple dataset of points in $\mathbb{R}^2$ belonging to two different classes. We try to classify it by fitting a line $x_2 = ax_1 + b$, so that the model only depends on the two parameters a and b and can be visualized. The dataset is designed to be separated by a line which we refer to by *old line*. Then, a few samples are removed to construct the retain dataset so that it can also be separated by another line denoted by *new line* and by any line in between these two.

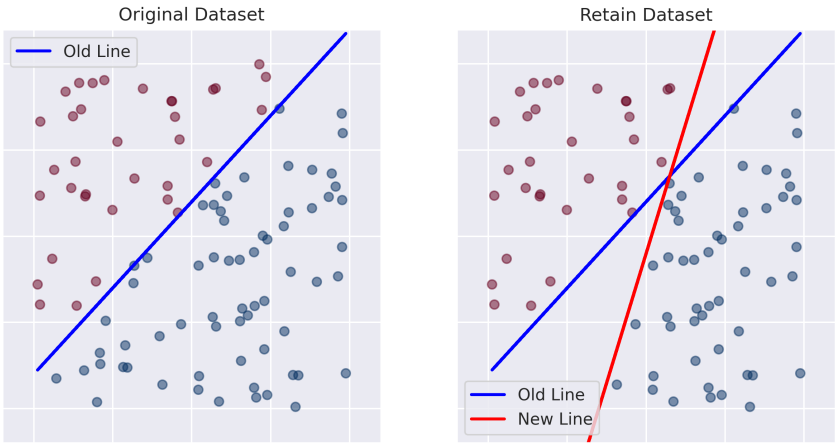


FIGURE 3.1: The original and retain dataset. From left to right: the original dataset and the retain dataset. Points are colored according to the class they belong to. A blue line shows the original separation. A red line shows a new valid separation that has been used to remove the forget set.

We suppose that our model gets trained on the original dataset, so in the a, b-plane it is represented by the blue dot. Then, we want it to forget the points we removed as if it had only seen the retain set. In Figure 3.2 we represent the logistic loss given at each point (a, b) without any regularization term. We can see in it that the original model is no longer the optimal one (the optimal is the point towards all the arrows face), but since it is not committing any explicit error, the magnitude of both the loss

and the gradient is very small. If we continued training on the retain dataset from the blue point, it would evolve very slowly and possible attain convergence before it arrives to the line one would consider natural given only the retain set.

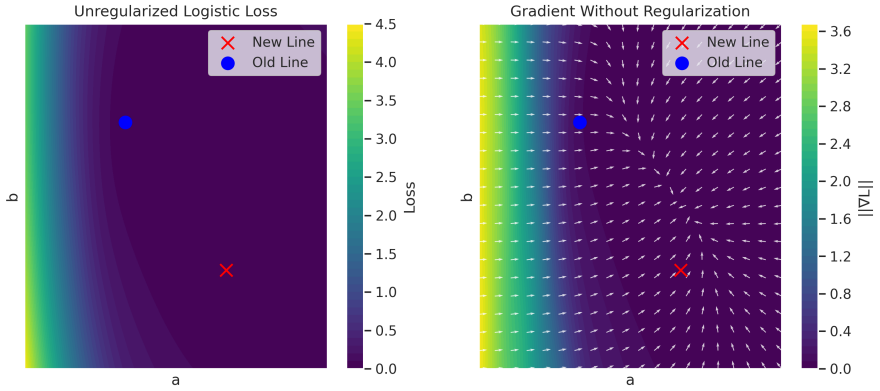


FIGURE 3.2: The unregularized loss. Values are shown on the a, b-plane. A point (a, b) corresponds to a model $x_2 = ax_1 + b$. a is plotted on the horizontal axis and b on the vertical one. The left figure shows the value of the loss. The right figure shows the magnitude of the gradient of the loss with respect to (a, b) together with the gradients of the loss with respect to (a, b) all normalized to the same length to show the direction that a learning curve would follow.

In figure 3.3 we represent the logistic loss plus the regularization loss at each point (a, b). The regularization loss adds a penalization around the original model, and forces the model to quickly distantiate from it while still evolving towards minimizing the logistic loss.

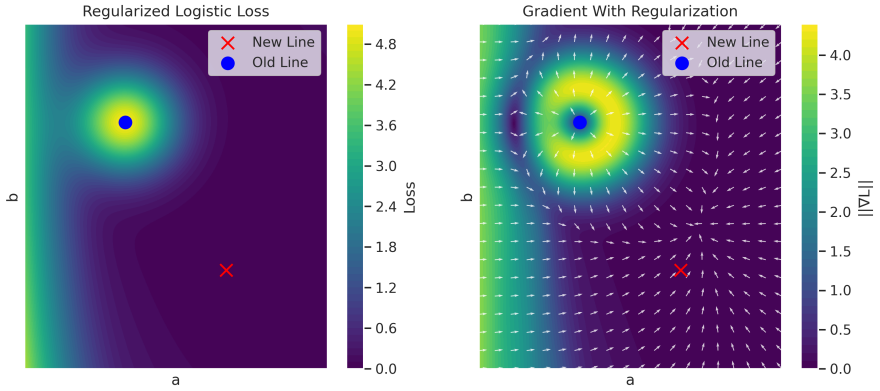


FIGURE 3.3: The regularized loss. Values are shown on the a, b-plane. A point (a, b) corresponds to a model $x_2 = ax_1 + b$. a is plotted on the horizontal axis and b on the vertical one. The left figure shows the value of the loss. The right figure shows the magnitude of the gradient of the loss with respect to (a, b) together with the gradients of the loss with respect to (a, b) all normalized to the same length to show the direction that a learning curve would follow.

As we just said, the layers to which we compute the regularization are a parameter of this algorithm. In this work, we opt to apply it to all the layers to not impose any extra limitation that could impact the model performance. The interest of studying the impact of choosing some subset of layers is discussed in Chapter 6, §6.2.5.

3.2 Defining our metrics

As seen in §2.2, there is no standard way to measure the fitness of an unlearning algorithm. Therefore, together with our proposed algorithm, we introduce the metrics that we will use in our experiments.

Our metrics are defined in order to capture our two concerns. The first one being that the unlearned model keeps making good predictions. And the second one, that the model has unlearned the forget set. We consider that the model has not learned a subset of the data if it predicts on it similarly as the baseline does. These two concerns will be respectively captured by the two following definitions. First, let X be the train set, U be the forget set (so $X \setminus U$ is the retain set), $Y_{X \setminus U}$ the real labels of $X \setminus U$, M_X the original model, $M_{X \setminus U}$ the baseline model and $M'_{X \setminus U}$ the unlearned model.

Definition 3 *We define the* utility *of the unlearned model as*

$$u = \frac{CE(M'_{X \setminus U}(X \setminus U), Y_{X \setminus U})}{CE(M_{X \setminus U}(X \setminus U), Y_{X \setminus U})}$$

There are several things to notice about this definition. Firstly, it is only defined if $\mathrm{CE}(M_{X \setminus U}(X \setminus U), Y_{X \setminus U}) \neq 0$, but it is a reasonably easy to satisfy assumption. Secondly, it is defined in this way because we would expect the unlearned model's accuracy to diminish once we force it through an unlearning process, so that this fraction would be between 0 and 1. And thirdly, that the previous might not always be the case, and so the utility can be greater than 1.

Definition 4 *We define the* forgetting *of the unlearned model as*

$$f = 1 - \frac{1}{\sqrt{2}|U|} \sum_{x_i \in U} \|M'_{X \setminus U}(x_i) - M_{X \setminus U}(x_i)\|_2$$

The reasoning behind this definition is to create a number between 0 and 1 that measures how close are the predictions of the unlearned model to the ones of the baseline model over the forget set. We want them to be as similar as possible, and the forgetting value will be higher the more close the two predictions are over the whole forget set.

The overall metric should result of a compromise between the previous two. This guides the next definition.

Definition 5 *We define the* metric *of the unlearned model as*

$$m = u \cdot f$$

where u and f are its corresponding utility and forget, respectively.

Finally, please note that in these definitions, we are only mentioning the train, retain and forget dataset, while, in practice, there are validation and test datasets as well that add a bit of complexity. In our experiments, we will detail over which specific set was every metric computed.

Chapter 4

Methodology

This chapter is devoted to define a clear methodology that we will follow throughout our experiments.

The purpose of the comparison experiments is defined next. A good way to start to understand an unlearning algorithm is to see how it behaves in comparison to other ones. That is: is it worse than the most basic one? Does it achieve a similar performance than the state-of-the-art ones? Does it fall in-between? Thus, the idea of setting a testing framework to compare them appears naturally. The simplest unlearning algorithm is naturally defined: it is the Fine-tuning algorithm. Any deep learning model can use it since it means to continue applying its learning algorithm just on a smaller subset. As a consequence, it can be set as the lower bound of the acceptable performances of unlearning algorithms: any unlearning algorithm that performs worse than the Fine-tuning will hardly be worthy of further investigation.

However, when it comes to compare to more sophisticated algorithms, things get more complicated. As an example, let's consider the Fanchuan model (defined in Chapter 2 §2.4.1). Suppose we set a framework, we test our model against it and it turns out to outperform Fanchuan. We could not extract from this experiment the conclusion that our model beat a state-of-the-art model. This is because Fanchuan, as long as other existent unlearning algorithms, were designed for a specific context (e.g., dataset, architecture and evaluation metrics). It may be the case that these models are generalizable to other situations and can work in different datasets, as it may not. Moreover, as explained in §4.6, we will not try to optimize Fanchuan over all its hyperparameter space, only on a subset of it. Therefore, a consistent way to address this issue is to think the following: *we are trying to find a model that outperforms ours.* If we manage to do so, we will have set an upper bound for it. If we do not, that will simply mean that we could not find a model that outperformed ours, not that it does not exist. It is by following this idea that we will compare the Regret model to the Fine-tuning and the Fanchuan ones. If our model turns to be better than our implementation and optimization of the Fanchuan algorithm, we will not extract the conclusion that our model is better than it.

4.1 Datasets construction

In this section we explain how given an arbitrary dataset, we split it into different subsets that will serve different purposes as training, validation and test. The rules explained here are followed across all the considered datasets and are therefore independent to each particular instance of them.

For each dataset, 5 subsets will be constructed in the following way. Let $D = (X, y)$ be an arbitrary dataset. Let F denote the *forget rule*. F will be constructed in a semantical way. This is, it won't be a particular set of points of the dataset. Rather, it will denote a semantical region. In the case of a dataset whose X part lies in $\mathbb{R}^2$, F

could mean to forget the X region $\{(x_1, x_2) \in \mathbb{R}^2 : x_1^2 + x_2^2 < 2\}$ and its corresponding y part. Another example could be a whole class y_0 and would therefore mean the region $\{(X, y) : y = y_0\}$. With this approach, since F is not defined with a particular subset of points, but with a general rule, it can extend to different sets to create the retain-train set and the retain-validation set. Given any $S \subset D$, let $F(S)$ represent the dataset resulting of applying the forget rule F to the subdataset S.

1. $D_{\text{test}} \subset D$ is selected randomly with $|D_{\text{test}}| = 0.2|D|$.
2. We randomly split $D \setminus D_{\text{test}} = D_{\text{train}} \sqcup D_{\text{val}}$, with $|D_{\text{train}}| = 0.7|D \setminus D_{\text{test}}|$ and $|D_{\text{val}}| = 0.3|D \setminus D_{\text{test}}|$.
3. The retain-train dataset is $D_{\text{retain-train}} = D_{\text{train}} \setminus F(D_{\text{train}})$, the retain-validation dataset is $D_{\text{retain-val}} = D_{\text{val}} \setminus F(D_{\text{val}})$.
4. We still need a subdataset over which compute the forget metric, and this will be $D_{\text{forget}} = F(D_{\text{train}})$.

For each dataset, we will choose an architecture, which will be shared through all the unlearning algorithms we will consider. An original model will be trained on D_{train}, and validated on D_{val}. The validation dataset will be used for early stopping. A baseline model will be trained on $D_{\text{retain-train}}$, and validated on $D_{\text{retain-val}}$. During all our experiments, $D_{\text{retain-val}}$ is only used to prevent the baseline model from overfitting. These original and baseline models will be shared across all algorithms for the same dataset.

4.2 Setting the original learning rate

By original learning rate we refer to the learning rate used by the original and baseline models. This learning rate will have a strong influence on the original number of training epochs: a higher learning rate will tend to require less training epochs and vice-versa. Therefore, the setting of the original learning rate has an impact on the maximum performance that the unlearning algorithms can attain. The good news are that such impact is homogeneous over all of them, so it does not affect their comparison. As a consequence, the initial learning rate is not trivially determined but neither is it analyzed with the same level of detail as the learning rates of the unlearning models.

It is set following the next methodology for each architecture and dataset. We plot the validation loss curves of the original model across different training epochs, starting on a low learning rate that still yields to convergence, until a high learning rate that starts to create uneven jumps indicating instability. These curves often follow a monotonous pattern of being steeper as the learning rate increases up until they start to show irregularity. The final learning rate is selected as the one from the set $\{10^{-n}\}_{n\in\mathbb{N}} \cup \{5 \cdot 10^{-n}\}_{n\in\mathbb{N}}$ that is the steepest while still not showing irregular jumps.

4.3 Setting the number of epochs

The original model will be limited to train for a certain number of epochs for each dataset. In the end, the original model will train for such number of epochs if it did not converge or less if it did and hence the training was ended due to early stopping. This final number of epochs will determine for how long can be the unlearning model

trained. The number of epochs through which the unlearning algorithms will be allowed to train will be set by a fraction of the original epochs. Both the number of original epochs and the fraction of unlearn epochs will be specified when each experiment is introduced.

The early stopping mechanism is defined as follows. At each epoch during the learning process, the validation loss is computed over the validation set (respectively, the retain-validation set). If this validation loss does not improve for 50 epochs in a row, the learning process is stopped.

Note that for each epoch, the Fanchuan algorithm iterates through the train dataset and then through the forget dataset. Still, we can consider this to be just one epoch comparable to the Fine-tuning and Regret ones since the forget size is presumed to be minimal with respect to the train size. And it is actually the case for our experiments. Therefore, all algorithms are allowed to run for the exact same number of epochs.

4.4 Managing uncertainty

The learning process is not deterministic. It depends on weights initialization, which determine to which local minimum can the model converge. This source of uncertainty has been mitigated by always initializing the weights' matrices to orthogonal matrices and the weights' bias terms following a $\mathcal{N}(0, 1)$ distribution.

Also, most deep learning algorithms, and certainly the ones we use, rely on stochastic gradient descent, which obviously has a random component. As a consequence, $(X, \mathcal{L}, \mathcal{M})$ do not determine M_X but rather a probability distribution over $\mathcal{M}$. In the problem of machine unlearning, there are two added main sources of uncertainty: the baseline and the unlearned model.

Let's analyze them one by one. First, let us explain why we think that the original model, at least for these experiments, can be considered as something fix. For the datasets and architectures that we are considering, it is easy and relatively fast to achieve convergence. So almost all original models will converge to making the same predictions.

Now, let's focus on the baseline source of uncertainty. The baselines have not seen any sample in the forget region, so they may classify them differently as they have no reason to converged. Baselines behave qualitatively differently depending on whether the given dataset contains two classes or more. We next discuss each case. In the case of a dataset containing only two classes, when samples of a class are removed on a region, the model tends to shift its predictions on such region towards the other class. Since there is only one direction away from the original prediction, all baselines follow a similar pattern. Therefore, in these cases, a single baseline was considered. In contrast, for datasets containing more than two classes, there are several directions in which predictions can evolve once a model does not see a certain class. This was observed by experimental data, where different baselines showed a completely different distribution of predictions over the class they had not seen during training. As a consequence, several baselines were selected to compute the metrics on. They were selected randomly without any filter to try to be the most faithful to the true underlying distribution of baseline models given the retain and forget data. In the case that several baselines are considered, the metric of a model will be assessed over all the baselines and the metric evaluated at each pair of baseline and unlearned model will be considered as an observation of the metric distribution.

Finally, unlearn models show a high variability. In the previous cases, what was limiting variability was *convergence*. However, when running unlearn models, there may not be convergence on the forget set, as we explained in the preceding section. Moreover, there might also not be convergence on the retain set since we are limiting the number of epochs through which they can train, oftenly before convergence. We have consistently executed every unlearn algorithm for several times, to try to capture the distribution of outcomes.

4.5 Hyperparameter optimization

The unlearning algorithms that we will consider depend on different parameters. Before comparing the probability distributions that each of them produce, we aim to determine the optimal parameters for each. We are trying to optimize a *distribution*, and this requires to define an order in the corresponding space of distributions. This is what this section is for.

Following notation of definition 2 in Chapter 2 §2.1, let $\mathcal{U}$ be an unlearning algorithm that depends on a parameter (or a tuple of parameters) $\omega \in \Omega$. Denote by $\mathcal{U}_\omega$ the algorithm whose parameter is set to ω. Let m denote the metric, so that $m(M_\omega)$ is the metric value that an unlearned model M_ω attains in an experiment and consider it as a random variable. Let $\mathbb{E}$ be the expectation operator, so that $\mathbb{E}[m(M_\omega)]$ is the expected value of the metrics of that model.

We set the following order in the space of metrics random variables. We consider that the distribution of metrics achieved by an unlearning algorithm set by parameter ω_1 is lower than the distribution given by the same unlearning algorithm with parameter ω_2 if

$$\mathbb{E}[m(\mathcal{U}_{\omega_1}(X, U, M_X))] < \mathbb{E}[m(\mathcal{U}_{\omega_2}(X, U, M_X))]$$

In plain words: for each unlearning algorithm on each dataset, we will choose the parameters that make it achieve the greatest expected value for the metric. For this optimization step, the utility will be computed on the validation set.

For what we explained in §4.2, it is evident that the learning rate plays a key role on any deep learning problem. The unlearning algorithms we will consider depend on different losses and different optimization processes, so their dependence on the learning rate is entirely different. It would not be fair to fix a learning rate and analyze all the models on it, nor would it be to choose a different one for each of them without justification. Therefore, before analyzing any model on a dataset, we start by analyzing its dependence on the learning rate. Then, it is for its best learning rate for which it is compared against the other ones.

We now list the parameters which we will optimize for for each algorithm. For the Fine-tuning model, just the learning rate. For the Regret model, the learning rate and the C parameter. For the Fanchuan model, see the next section.

4.6 Fanchuan optimization

The Fanchuan algorithm has an intrincate unlearning process which depends on several parameters. Optimizing through all of them is far beyond what our resources allow us. Here, we describe the rules we have followed for this optimization process.

The Fanchuan algorithm depends on four parameters:

- The learning rate for the KL-divergence step (α_{KL}).

- The learning rate for the cross-entropy minimization steps (α_{CE}).
- The learning rate for the contrastive loss maximization steps (α_{CL}).
- The temperature coefficient to control the contrastive loss maximization steps (t).

The learning rates dependencies have been simplified by the following law:

$$\begin{cases} \alpha_{\mathrm{KL}} = 1.25\,\alpha_{\mathrm{CE}} \\ \alpha_{\mathrm{CL}} = 0.075\,\alpha_{\mathrm{KL}}. \end{cases}$$

Which means to keep the same proportions between them as in the original Fanchuan implementation (cf. [4]). We fix $t = 1.15$, as in the original implementation (cf. [4]) for the Google challenge. This leads to only optimize across α.

This means that for Fanchuan, following the previous notation, we are not optimizing over Ω but over a subset of it. So, if we can make the Fanchuan model better than ours even if t is fixed and the learning rates are simplified, we will have found an upper bound. If not, it may or may not be attained by optimizing across Ω.

4.7 Final evaluation

Once the algorithms' hyperparameters have been optimized over the validation set, they are tested according to the metric whose accuracy part is measured over the test set. The number of samples that are generated will be specified in the corresponding *Final results* section of each dataset.

Chapter 5

Experiments

This section is devoted to detailing the experimental results. We conduct experiments to compare the performance of Fine-tuning, Regret and Fanchuan over four datasets using two architectures. For each dataset, each experiment report begins by explaining the dataset and showing the five subsets considered. Then, we describe the architecture used. We create the original and baseline models and report the parameters used for their training along with the deliberation process for choosing them. The optimization process under which each unlearning algorithm undergoes is detailed in Appendix A. In this chapter we list the optimal parameters configuration found for each case. Finally, we report the results on test data by which we assess the performance of the models.

5.1 $\mathbb{R}^2$ datasets

5.1.1 Considered datasets

In this section we shall investigate how does our unlearning algorithm behave for simple datasets consisting of points on $\mathbb{R}^2$ with their associated binary label. We choose these datasets for the initial experiment for their high interpretability since they allow the model's predictions evolution to be visualized. They are simple, visualizable and interpretable, and we expect them to provide us with some intuition to generalize in more complex and non visualizable datasets. Also, they require less computational effort, we will be able to generate more samples of the studied distributions. Each dataset is comprised of 2000 points.

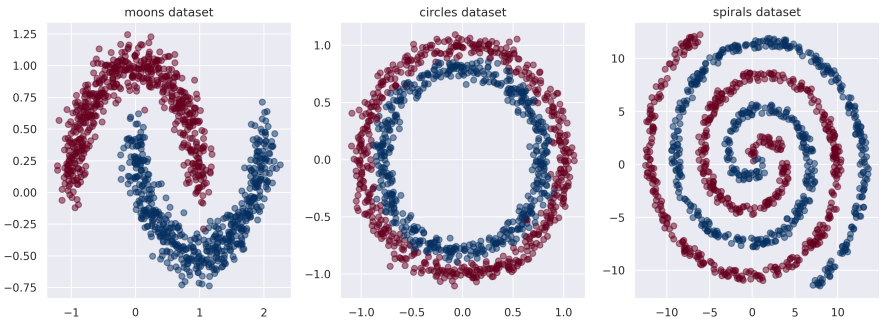


FIGURE 5.1: The three datasets we considered. From left two right, they are referred to by the respective names *moons*, *circles* and *spirals*. Each point is colored according to the class it belongs to.

5.1.2 Chosen architecture

For these datasets we choose an architecture consting on four fully connected layers since it is simple and still is able to learn the data's underlying pattern. Between each layer, the ReLU activation function is used. The final layer outputs a two dimensional logits function which is passed through a softmax function to become the predicted probability for each class. We append next the summary table of the model, which shows precisely the number of parameters of each layer, their shape and the order in which they are called, together with some additional information about the model.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                      [-1, 5]              15
              ReLU-2                      [-1, 5]               0
            Linear-3                     [-1, 10]              60
              ReLU-4                     [-1, 10]               0
            Linear-5                     [-1, 15]             165
              ReLU-6                     [-1, 15]               0
            Linear-7                      [-1, 2]              32
           Softmax-8                      [-1, 2]               0
================================================================
Total params: 272
Trainable params: 272
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.00
Estimated Total Size (MB): 0.00
----------------------------------------------------------------
```

5.1.3 Initial experiment on the moons dataset

For our initial experiment, we choose the moons dataset. We proceed to visualize its corresponding subsets. Figure 5.2 shows, in this order, the train set, the validation set, the retain-train together with the forget set, the retain-validation set and the test set.

The retain-train and forget set are plotted together, circles represent the retain points and lighter crosses the forget ones.

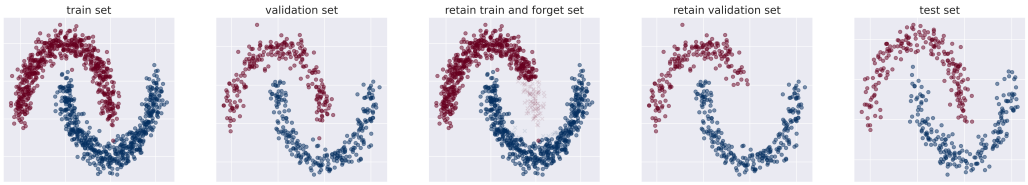


FIGURE 5.2: The complete moons dataset. From left to right. The train set. The validation set. The retain-train set with the forget set marked by paler crosses. The validation-retain set. The test set.

For the moons dataset, the original epochs are limited to 100. The unlearn epochs will be given by 20% of the final original epochs.

Original and baseline models

The epochs during which the original model was trained will constrain the upcomming experiments, and since they directly depend on the learning rate, the latter cannot be set arbitrarily. Figure 5.3 shows the evolution along the number of epochs of the validation loss of the original model, for different learning rates (`lr`). As expected, there is a trade-off between speed and stability. According to these results, a learning rate of 0.005 is considered adequate.

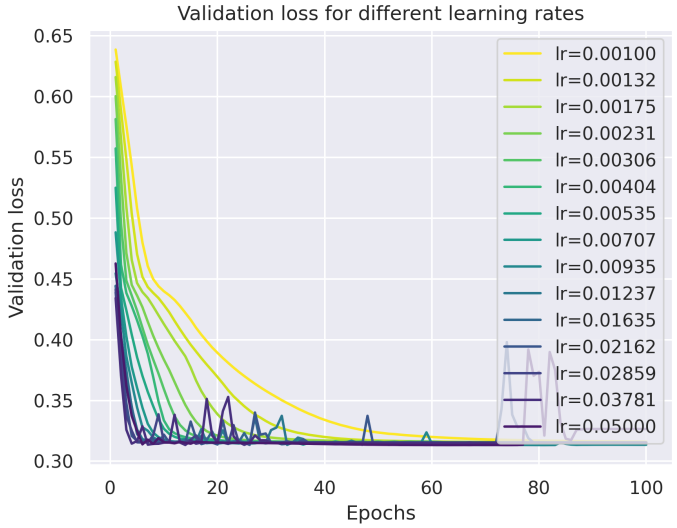


FIGURE 5.3: Validation loss dependency on the learning rate for the original model on the moons dataset. On the horizontal axis: the epochs count. On the vertical axis, the validation loss. A different line plots the validation loss evolution across epochs, its color represent the learning rate that is used, shown in the legend.

A training for a maximum of 100 epochs and a learning rate of 0.005 yielded the original and baseline models given in figure 5.4. None of them early-stopped. This set the unlearn epochs to 20.

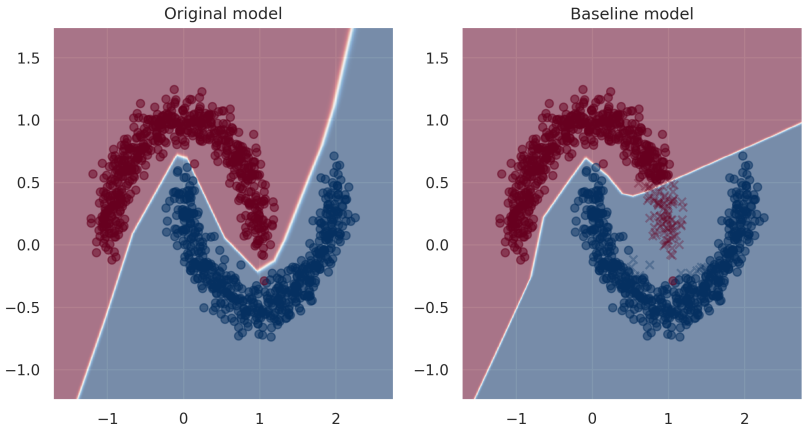


FIGURE 5.4: The original and baseline models for the moons dataset. From left to right: the original and the baseline model. The model's decision boundary is plotted overlapped with the train set for the original model and the retain and forget set for the baseline one. Each region is colored according to the model's predictions on it, following the same colormap as the data points.

Models optimization results

The unlearning algorithms Fine-tuning, Regret and Fanchuan have been optimized according to the methodology detailed in Chapter 4, §4.5 for the moons dataset given the original and baseline models generated in §5.1.3. Each corresponding optimization process is reported in Appendix A, §A.1. The optimization process was performed using only validation data, not the test set. The optimal parameters found for each model are listed in Table 5.1.

Algorithm	**Parameter name**	**Optimal value**
Fine-tuning	learning rate	10^{-1}
Regret	learning rate	10^{-3}
	C	1
Fanchuan	learning rate	5×10^{-2}

TABLE 5.1: Optimal parameters for Fine-tuning, Regret and Fanchuan algorithms on the moons dataset. They have been measured on validation data.

Final results on the moons dataset

Using the optimal configurations detailed in §5.1.3, we now show the results attained on the test dataset. For testing on the moons dataset, a hundred models were generated for each algorithm.

algorithm	**metric**	**forget**	**utility**
Fine-tuning	0.72 ± 0.24	0.77 ± 0.24	0.93 ± 0.15
regret	0.83 ± 0.12	0.81 ± 0.13	1.02 ± 0.02
fanchuan	0.87 ± 0.02	0.88 ± 0.03	0.99 ± 0.01

TABLE 5.2: Mean and standard deviation of metric, forget, utility, grouped by algorithm for the moons dataset. Each value is given by its mean ± the observed standard deviation.

For this dataset, the Fanchuan model outperformed the Regret one, while the latter outperformed Fine-tuning. We can see on the distributions shown in Figure 5.5 that for each metric, the distributions of the Fanchuan algorithm show the lower spread, followed by the Regret' ones. The Fine-tuning algorithm shows the largest spread, indicating a high variability in the output, probably due to its requirement of too high learning rates since it is slower than the other two algorithms. The spread in the Fine-tuning algorithm makes it achieve a relatively low average value. The greater spread on the left tail of the Regret algorithm distribution compared to Fanchuan also makes its average value to be lower. These spreads indicate slowliness because they are given by the algorithms requiring learning rates too high due to not being able to reach good metrics within the unlearn epochs limit. The best overall distribution is obtained by the Fanchuan algorithm.

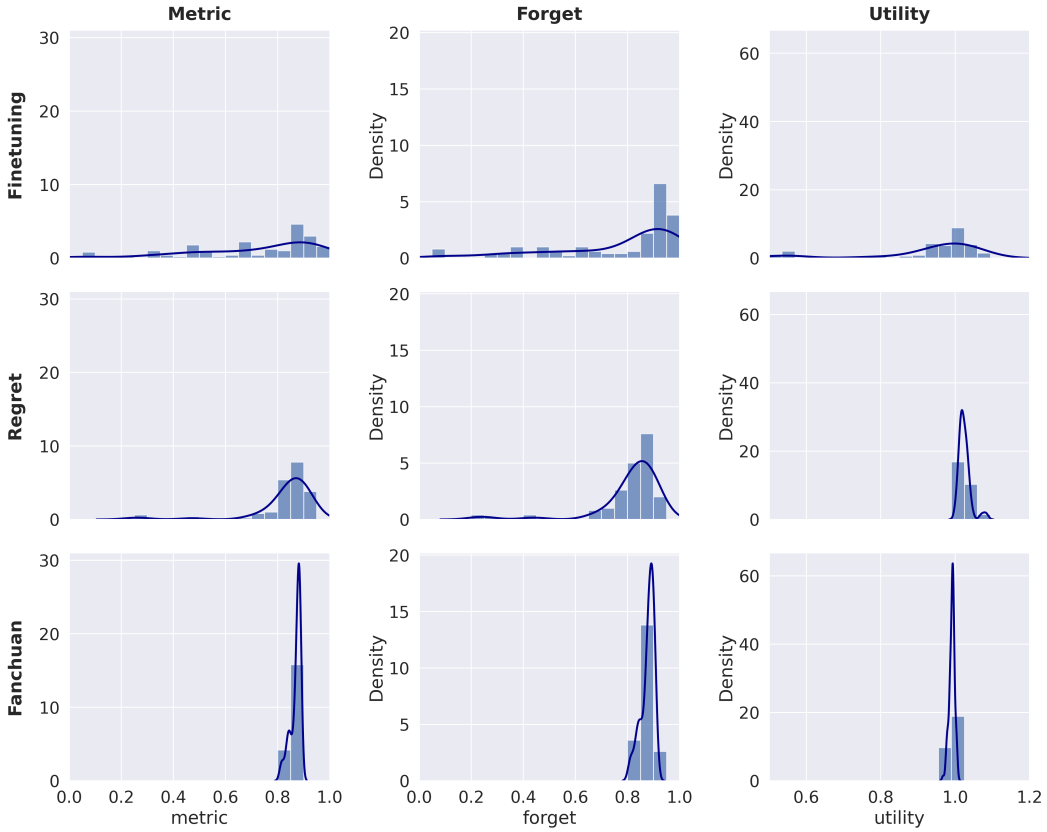


FIGURE 5.5: The metrics distribution for each model on the moons dataset for test subset. Columns represent, from left to right, the metric, forget and utility. Rows represent, from top to bottom, the Fine-tuning, the Regret and the Fanchuan models. Each graph represents the density of the histogram of the corresponding results attained on the test, together with a line showing the approximate density curve obtained by Kernel Density Estimation.

5.1.4 Experiments on the circles and spirals datasets

We now proceed to extend the experiments on the other two toy datasets. Our intention is to see if the patterns seen in the previous section are mantained in different (though similar) datasets. For these two datasets, the original learning epochs will be limited to 300. The number of unlearn epochs will be 20% of the final original epochs.

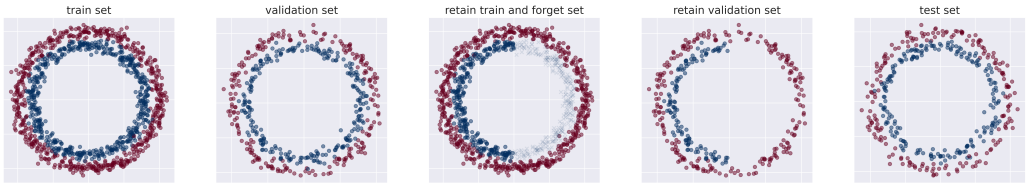


FIGURE 5.6: The complete circles dataset. From left to right. The train set. The validation set. The retain-train set with the forget set marked by paler crosses. The validation-retain set. The test set.

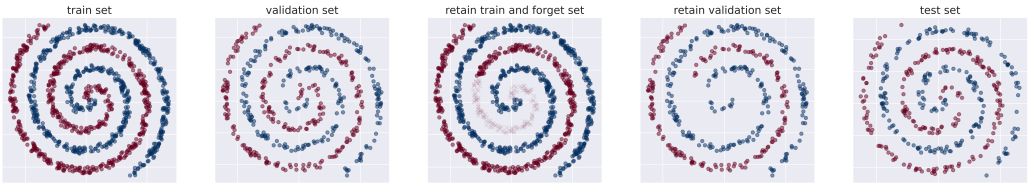
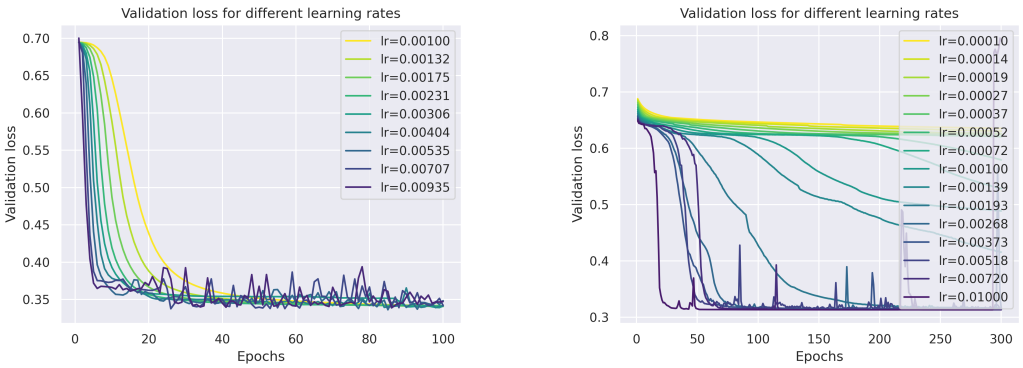


FIGURE 5.7: The complete spirals dataset. From left to right. The train set. The validation set. The retain-train set with the forget set marked by paler crosses. The validation-retain set. The test set.

Original and baseline models

To start, we compute the validation loss by epochs for different learning rates to select an appropriate one for each dataset.

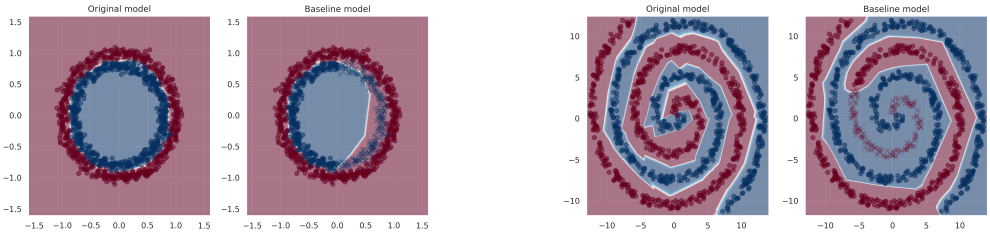


(A) On the circles dataset.

(B) On the spirals dataset.

FIGURE 5.8: Validation loss dependency on the learning rate for the original model on the circles and spirals dataset. On the horizontal axis: the epochs count. On the vertical axis, the validation loss. A different line plots the validation loss evolution across epochs, its color represent the learning rate that was used, shown in the legend.

Based on results shown in Figure 5.8a, a learning rate of 0.002 was selected for the baseline and original model. They were then trained for a maximum of 300 epochs and early stopped after 156 epochs. This set the unlearn epochs limit to 30. Conversely, based on figure 5.8b, the selected learning rate was 0.002. The model was allowed to train for 300 epochs and did not early stopped. This set the unlearn epochs bound to 60.



(A) On the circles dataset. (B) On the spirals dataset.

FIGURE 5.9: The original and baseline models for the circles dataset. From left to right: the original and the baseline model. The model's decision boundary is plotted overlapped with the train set for the original model and the retain and forget set for the baseline one. Each region is colored according to the model's predictions on it, following the same colormap as the data points.

We show in Figure 5.9 the original and baseline models of the circles and spirals datasets. We observe that the circle baseline, due to not seeing the right half of the inner blue points, contracts a little bit the right half of the decision boundary towards the left. The spirals baseline ends the red spiral right where the forget set starts.

Models optimization results

The unlearning algorithms Fine-tuning, Regret and Fanchuan have been optimized according to the methodology detailed in Chapter 4, §4.5 for the circles and spirals datasets given the original and baseline models generated in §5.1.4. Each corresponding optimization process is reported in Appendix A, §A.2 and §A.3 for the circles and spirals datasets, respectively. The optimization process was performed using only validation data, not the test set. The optimal parameters found for each model are listed in Table 5.3.

Dataset	Algorithm	Parameter name	Optimal value
Circles	Fine-tuning	learning rate	10^{-2}
	Regret	learning rate	10^{-4}
		C	1
	Fanchuan	learning rate	10^{-2}
Spirals	Fine-tuning	learning rate	2×10^{-2}
	Regret	learning rate	2×10^{-2}
		C	1
	Fanchuan	learning rate	5×10^{-3}

TABLE 5.3: Optimal parameters for Fine-tuning, Regret and Fanchuan algorithms on the circles and spirals datasets. They have been measured on validation data.

5.1.5 Final results

Using the parameters that revealed optimal for the validation dataset, we proceed to compute the results on the test dataset. For testing on both the circles and spirals datasets, a hundred samples were generated for each algorithm.

TABLE 5.4: Mean and standard deviation of metric, forget, utility, grouped by algorithm for the circles and spirals datasets. Each value is given by its mean ± the observed standard deviation.

Dataset	Algorithm	Metric	Forget	Utiliy
Circles	Fine-tuning	0.84 ± 0.06	0.82 ± 0.07	1.02 ± 0.04
	Regret	0.91 ± 0.02	0.86 ± 0.02	1.06 ± 0.01
	Fanchuan	0.86 ± 0.04	0.85 ± 0.03	1.01 ± 0.04
Spirals	Fine-tuning	0.84 ± 0.15	0.87 ± 0.14	0.97 ± 0.08
	Regret	0.84 ± 0.13	0.94 ± 0.07	0.90 ± 0.12
	Fanchuan	0.95 ± 0.01	0.95 ± 0.01	1.00 ± 0.00

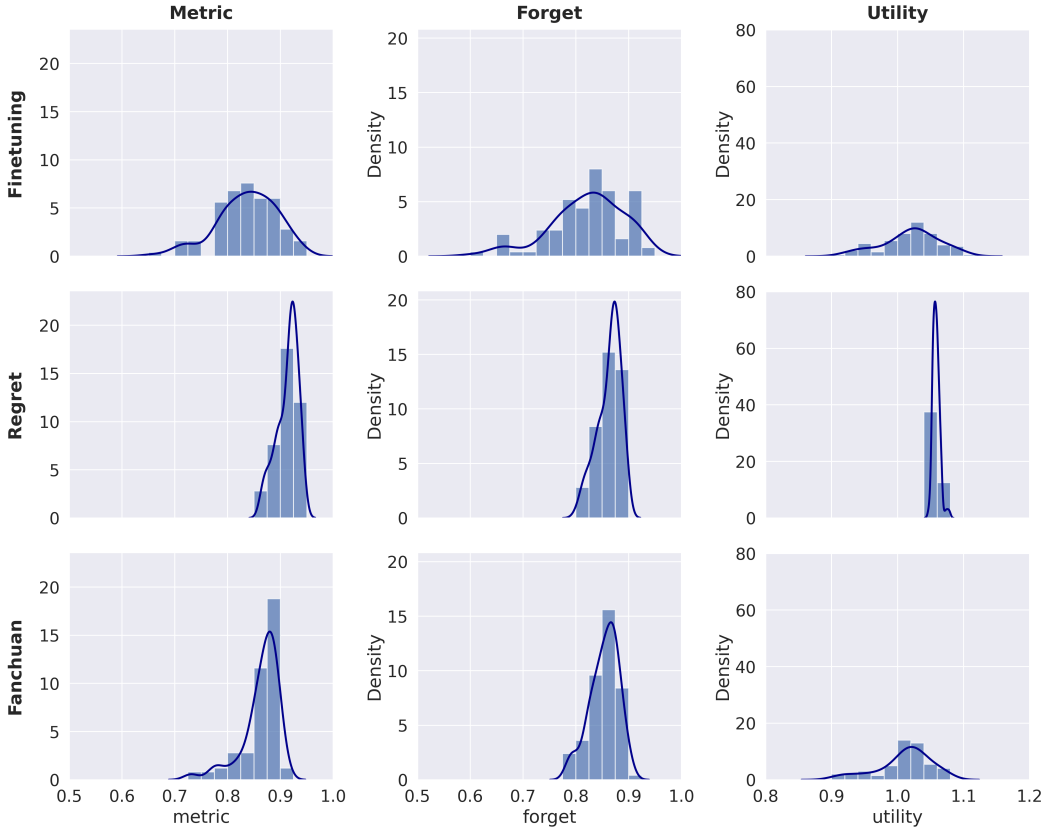


FIGURE 5.10: The metrics distribution for each model on the circles dataset for test subset. Columns represent, from left to right, the metric, forget and utility. Rows represent, from top to bottom, the Fine-tuning, the Regret and the Fanchuan models. Each graph represents the density of the histogram of the corresponding results attained on the test, together with a line showing the approximate density curve obtained by Kernel Density Estimation.

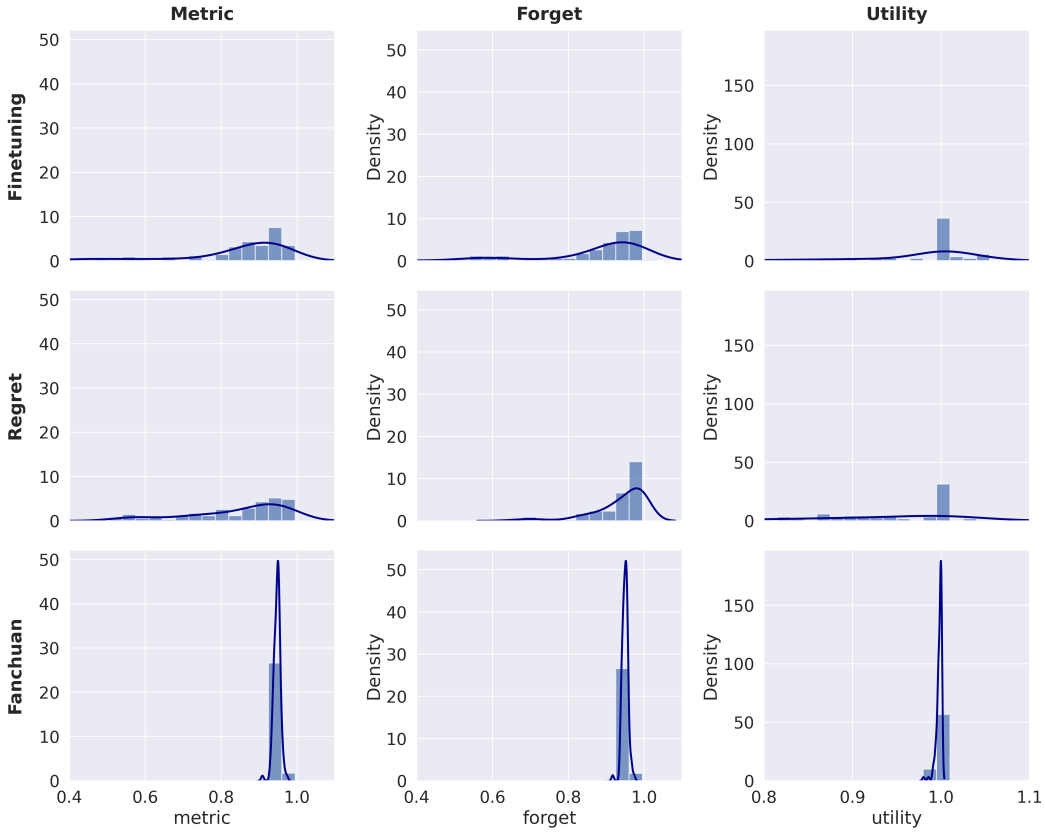


FIGURE 5.11: The metrics distribution for each model on the spirals dataset for test subset. Columns represent, from left to right, the metric, forget and utility. Rows represent, from top to bottom, the Fine-tuning, the Regret and the Fanchuan models. Each graph represents the density of the histogram of the corresponding results attained on the test, together with a line showing the approximate density curve obtained by Kernel Density Estimation.

For the circles dataset, the Regret algorithm attains the highest average metric at a value of 0.91. As Figure 5.16 shows, the distribution of the metric and forget are relatively similar between the Regret and Fanchuan models. The utility, however, is more similar between the Fanchuan and Fine-tuning models, showing a wide spread. The Regret distribution of the three metrics shows a narrow concentration, indicating that it could achieve a fine unlearning within the epochs limit and therefore was able to use a adequate learning rate yielding low uncertainty.

When it comes to the spirals dataset, we observe a different pattern. In it, the Fine-tuning and Regret algorithms presented a similar behaviour, with very high distribution spreads indicating high variability. In contrast, the Fanchuan model presents a very centralized distribution for the three metrics. This indicates that the latter algorithm can achieve correct unlearnings within the unlearn epochs limits while using adequate small learning rates, while the two former ones need to rely on too high learning rates yielding high instability.

In conclusion, the Regret model outperforms the Fine-tuning in both datasets. For the Circles dataset, the Regret model outperforms Fanchuan. And for Spirals, Fanchuan outperforms the Regret model.

5.2 MNIST dataset

5.2.1 The MNIST dataset

The MNIST dataset consists of 70000 pictures of hand-written digits between 0 and 9, together with the true digit they represent. For the reader to gain more insight on this dataset, Figure 5.12 illustrates five samples of this dataset, showing the handwritten digit image and its corresponding label. In Table 5.5 we show the digits distribution across each set constructed as indicated in the methodology given by Chapter 4, §4.1.

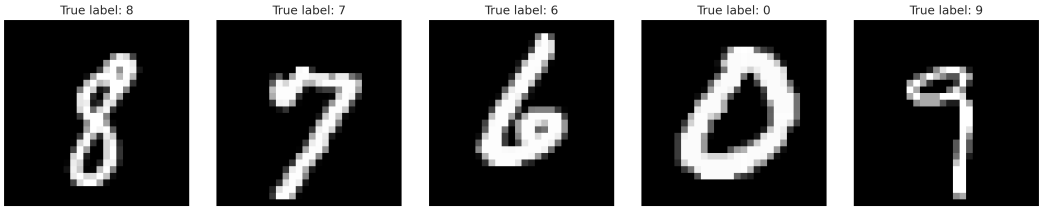


FIGURE 5.12: Some samples from the MNIST dataset. Each image is shown with its true label on top.

For the reader to gain more insight on this dataset, table 5.5 shows the digits distribution across each set.

Set	0	1	2	3	4	5	6	7	8	9	Total
Train	10%	11%	10%	10%	10%	9%	10%	10%	10%	10%	60000
Val	11%	12%	11%	10%	10%	9%	10%	10%	9%	10%	5000
Retain Train	11%	12%	11%	11%	11%	10%	0%	12%	11%	11%	54082
Retain Val	12%	13%	12%	11%	11%	10%	0%	11%	10%	11%	4502
Forget	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	5918
Test	9%	11%	10%	11%	10%	9%	9%	11%	10%	10%	5000

TABLE 5.5: Labels distribution across the complete MNIST dataset. Each column represents a possible label: the digits between 0 and 9. Top right-end column gives the total number of samples. Each row, from top to bottom, the train, validation, retain train set, retain-validation set, forget set and test set. The relative frequencies of each label on each dataset are shown in % values.

Training the architecture presented in §5.2.2 on an image dataset with so many samples as MNIST is very computationally expensive. Since the number of epochs through which the original model was trained on will determine the number of epochs for the unlearning algorithms, we were forced to keep this number low enough so we could run our experiments with enough samples to capture the distribution. For this reason, the original learning epochs were limited to 100 for this dataset. This number of epochs was experimentally checked to be enough for the model to learn the MNIST dataset (see Figure 5.14). The fraction of unlearn epochs was given by 10% of the original epochs. This fraction was lower than with the $\mathbb{R}^2$ datasets for computational limitations.

5.2.2 Chosen architecture

We now detail the architecture used for the MNIST dataset. It comprises convolutional layers, max pooling layers, flattening layers and fully connected layers. All convolutional layers have padding 1, stride 1 and kernel size 3×3. The ReLU activation function is used again. The final layer outputs a ten dimensional logits function which is passed through a softmax function to become the predicted probability for each class. As indicated in Chapter 3, §3.1, all unlearning algorithms —including Regret— will be applied to all the layers with trainable parameters, both the fully connected and the convolutional layers.

We append next the summary table of the model which shows precisely the number of parameters of each layer, their shape and the order in which they are called, together with some additional information about the model.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 32, 28, 28]             320
              ReLU-2          [-1, 32, 28, 28]               0
            Conv2d-3          [-1, 64, 28, 28]          18,496
              ReLU-4          [-1, 64, 28, 28]               0
         MaxPool2d-5          [-1, 64, 14, 14]               0
            Conv2d-6          [-1, 64, 14, 14]          36,928
              ReLU-7          [-1, 64, 14, 14]               0
            Conv2d-8          [-1, 64, 14, 14]          36,928
              ReLU-9          [-1, 64, 14, 14]               0
        MaxPool2d-10            [-1, 64, 7, 7]               0
           Linear-11                 [-1, 10]          31,370
             ReLU-12                 [-1, 10]               0
           Linear-13                 [-1, 10]             110
             ReLU-14                 [-1, 10]               0
           Linear-15                 [-1, 10]             110
          Softmax-16                 [-1, 10]               0
================================================================
Total params: 124,262
Trainable params: 124,262
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 1.65
Params size (MB): 0.47
Estimated Total Size (MB): 2.13
----------------------------------------------------------------
```

5.2.3 Original and baseline models

We start running several models for 100 epochs and different learning rates. We plot the error on the validation sets in figure 5.13. According to these results, a learning rate of 10^{-4} was selected. The original model was trained for 100 epochs and did not early-stop. Its confusion matrix is shown in figure 5.14.

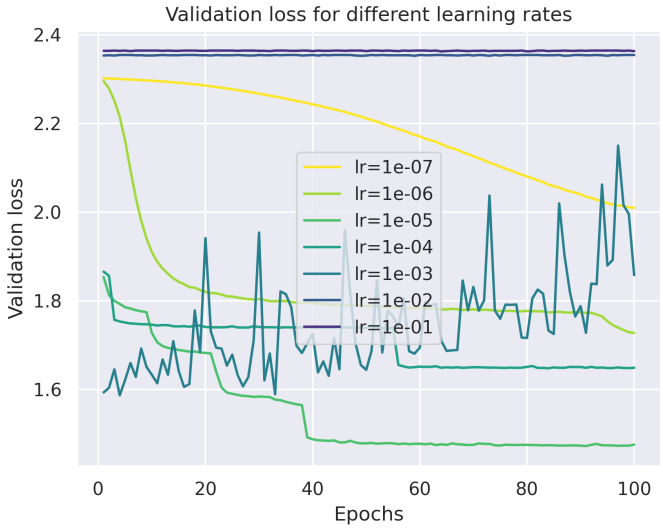


FIGURE 5.13: Validation loss dependency on the learning rate for the original model on the MNIST dataset. On the horizontal axis: the epochs count. On the vertical axis, the validation loss. A different line plots the validation loss evolution across epochs, its color represents the learning rate that was used, shown in the legend.

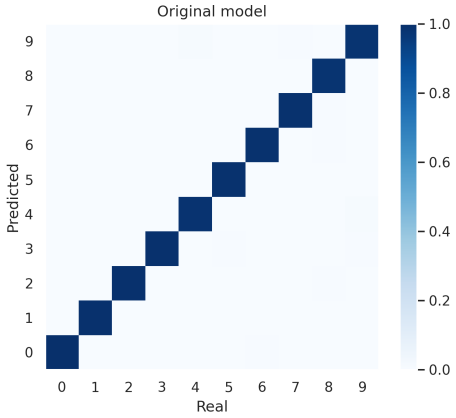


FIGURE 5.14: The original model for the MNIST dataset. We plot the confusion matrix of the model, with the real labels on the horizontal axis and the predicted ones on the vertical one.

For the MNIST dataset, there are several directions in which predictions can evolve once a model does not see a certain class. This was observed by experimental data, where different baselines showed a completely different distribution of predictions over the class they had not seen during training. As a consequence, 6 baselines were selected to compute the metrics on. They were selected randomly without any filter to try to be the most faithful to the true underlying distribution of baseline models given the retain and forget data. Their confusion matrices are shown in figure 5.15.

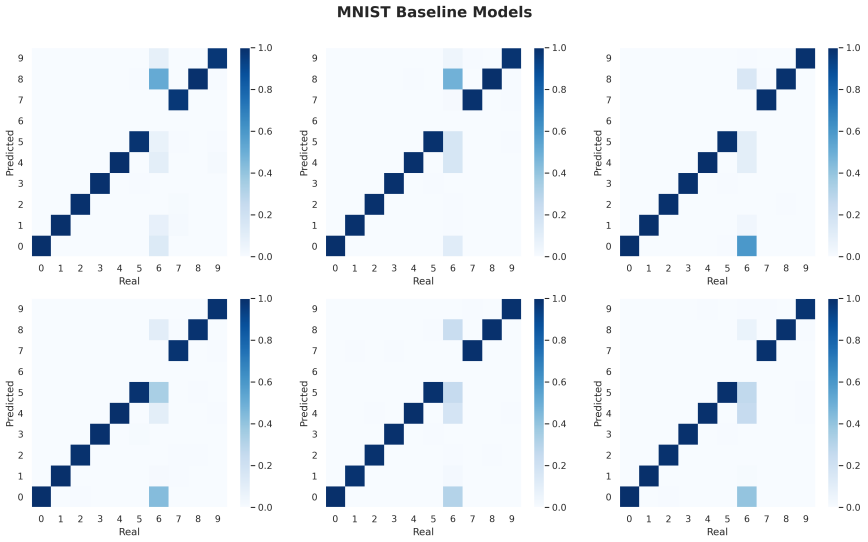


FIGURE 5.15: The baseline models for the MNIST dataset. For each one of the six baselines, we plot the confusion matrix with the same format as in Figure 5.14.

Models optimization results

The unlearning algorithms Fine-tuning, Regret and Fanchuan have been optimized according to the methodology detailed in Chapter 4, §4.5 for the MNIST dataset given the original and baseline models generated in §5.2.3. Each corresponding optimization process is reported in Appendix A, §A.4. The optimization process was performed using only validation data, not the test set. The optimal parameters found for each model are listed in Table 5.6.

Algorithm	Parameter name	Optimal value
Fine-tuning	learning rate	10^{-3}
Regret	learning rate	10^{-3}
	C	1
Fanchuan	learning rate	10^{-2}

TABLE 5.6: Optimal parameters for Fine-tuning, Regret and Fanchuan algorithms on the MNIST dataset. They have been measured on validation data.

5.2.4 Final results

Using the optimal hyperparameters found in the preceding sections, we proceed to give the final results achieved over the test set for the MNIST dataset. To test on the MNIST dataset, 30 samples were generated for each algorithm. We create less unlearn models than for the $\mathbb{R}^2$ datasets for computational limitations.

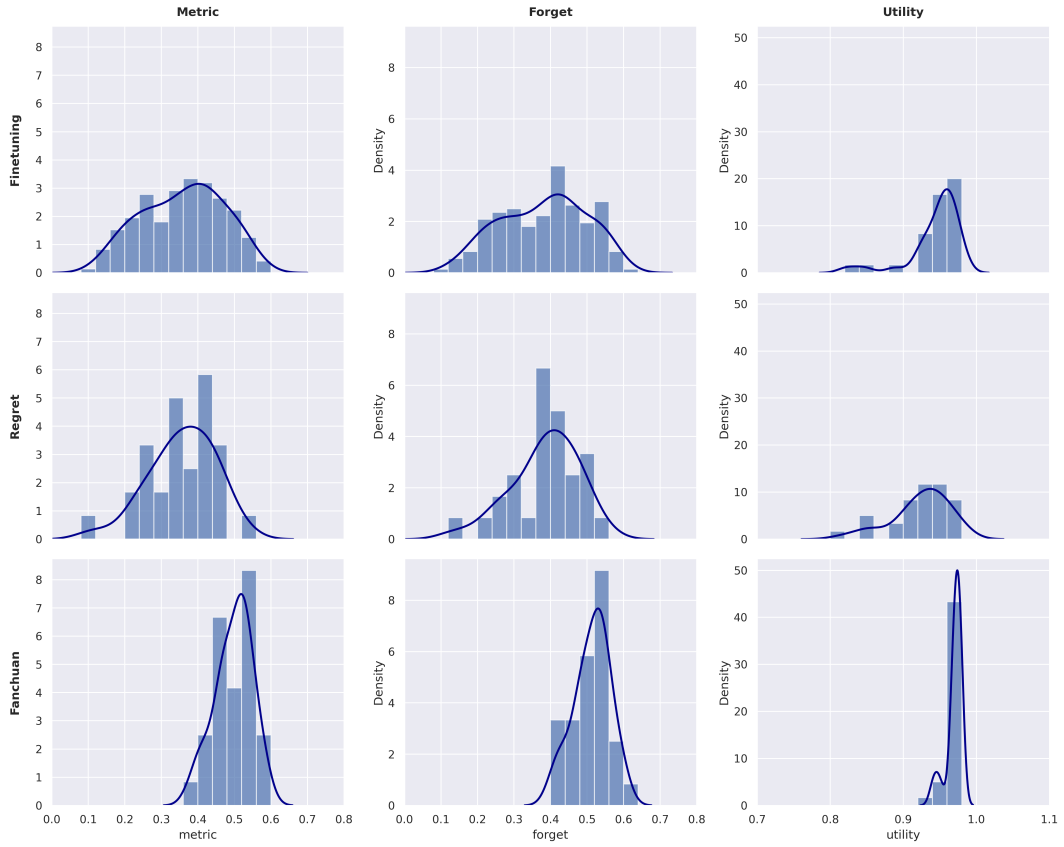


FIGURE 5.16: The metrics distribution for each model on the MNIST dataset for test subset. Columns represent, from left to right, the metric, forget and utility. Rows represent, from top to bottom, the Fine-tuning, the Regret and the Fanchuan models. Each graph represents the density of the histogram of the corresponding results attained on the test, together with a line showing the approximate density curve obtained by Kernel Density Estimation.

TABLE 5.7: Mean and standard deviation of metric, forget, utility, grouped by algorithm for the MNIST dataset on the test set. Each value is given by its mean ± the observed standard deviation.

Algorithm	**Metric**	**Forget**	**Utility**
Fine-tuning	0.36 ± 0.11	0.38 ± 0.12	0.94 ± 0.04
Regret	0.36 ± 0.09	0.38 ± 0.09	0.92 ± 0.04
Fanchuan	0.50 ± 0.05	0.51 ± 0.05	0.97 ± 0.01

This final evaluation shows that the Fanchuan model outperformed the Regret one. The Regret algorithm performed very similarly than Fine-tuning.

The Regret metric and forget distributions are less spreaded than the Fine-tuning ones, despite being centered around the same value. In contrast, the utility distribution is more concentrated and shifted to the right for Fine-tuning than for Regret. Overall, these two algorithms show an even performance on the MNIST dataset. Both the large spread they present plus the worsen of the predictions over classes different than the forget class (6) shown in Figures A.20 and A.22 indicate that these models

were not able to reach correct unlearnings within the limit of unlearn epochs, and hence required too high learning rates that produced high instability. As with the $\mathbb{R}^2$ datasets, what these spreaded distributions indicate is that these two models are slow.

In contrast, Fanchuan achieves much better results, with greater means and substantially lower variances than the other two algorithms. This low deviation together with the observation in Figure A.24 that no substantial worsen of predictive power appears for retain classes, indicates that the model could achieve correct unlearning within the epochs limit and so used adequate low learning rates. This in turn indicates that this model is faster than the other two.

Chapter 6

Conclusion and discussion

This final chapter is dedicated to synthetize this work and to provide a global conclusion that summarizes our understanding of the Regret model. Finally, we conclude this work with some open discussions.

6.1 Conclusions

In this work, we retrieved the problem of machine unlearning as presented in the related literature, gave a formal definition of it and recovered the different assessment methods used by scientists on this field. We then examined different unlearning methods, including several state of the art ones. In them, we observed a pattern: an erase phase followed by a repair one. This led us to propose a new unlearning model that did not rely on this method: the Regret algorithm.

We proposed our metrics and an evaluation framework for unlearning problems. We implemented three unlearning algorithms: Fine-tuning, Regret and Fanchuan. We have thoroughly tested them on three $\mathbb{R}^2$ datasets using an architecture consisting on fully connected layers. Moreover, we also tested them on the MNIST dataset, using a more complex architecture involving convolutional and pooling layers, together with fully connected ones. We have therefore tested them on four different datasets and two architectures.

The results varied within different datasets but overall we can conclude that the Regret algorithm produces a metric distribution that is more shifted towards greater values with still a lower spread than the Fine-tuning method. The gain was greater on the $\mathbb{R}^2$ datasets. It was less clear on the MNIST dataset, but in this dataset, due to its computational complexity, a substantial smaller sample of results was taken, resulting into a less statistically informative conclusion.

Our implementation of the Fanchuan algorithm, although it was not designed for dealing with these datasets or be evaluated with our metrics, nor optimized through all its parameters, greatly outperformed the Regret algorithm (and therefore Fine-tuning). This implies that although Regret is better than Fine-tuning, it does not compare with the state of the art models.

6.2 Discussion

6.2.1 Limitations of this work

We faced important computational limitations for carrying out these experiments. They were conducted on a computer whose key specifications are listed next. CPU: Intel i7-14700KF, RAM: 16 GB, GPU: Nvidia GeForce RTX 4060.

This made the experiments for the $\mathbb{R}^2$ datasets extremely time consuming. Despite training a single model on such datasets is relatively fast, the random nature

of these experiments required the training of not one but several (e.g., 81,920 models were needed to be trained to conduct the experiment reported in Annex B, §B.1). The code was optimized to use GPU acceleration and parallel multiprocessing, which substantially decreased the runtime. Still, these experiments took even days to finish.

For the MNIST experiments, things got more complicated. Parallel computing was unfeasible since the model and the data were very heavy and consumed all GPU's resources. So only one model could be trained at a time. Therefore, these experiments had to be more limited reducing both the domain points in which they could be assessed and also the number of samples that were generated to approximate the probability distribution. This notably impacted the quality of the optimization process conducted for the models.

6.2.2 Experimental framework

The experiments were defined by limiting the number of unlearning epochs to 20% or 10% of the training epochs of the original model. This being set, the models were free to optimize their hyperparameters according to the expectancy of their metric. This made the slower algorithms to require greater learning epochs, making the optimization process unstable, yielding a wide variability of results and potentially ending up in catastrophic unlearning. This is not a major concern, since when evaluating them, we have penalized wide spreads in the distribution.

Determining which learning rate is excessive is very complicated to formalize. Even more complicated is to do so when one is dealing with different loss functions, since a shared learning rate would not guarantee that. If, in contrast, one opts for normalizing all gradients to length 1 before multiplying by the learning rate, this would make gradients be of equal length no matter how close they are to a local optimum, slowing the optimization at far points and impeding convergence at close points.

Another way to construct the experiments would be as follows. Instead of fixing the number of epochs and then evaluate which metric is attained, we could have let each model to train until it is early stopped, and then evaluate both the metric attained and the number of epochs required. However, this approach also has its limitations. It may be a little inconsistent with some existing literature, since in these unlearning experiments, unlearning algorithms that require too many epochs tend to be discarded (cf. [10], where unlearning time is cut-off at 20% of the retain-from-scratch time). Also, if the models are penalized for the number of epochs they require, they might also opt for high learning rates, since this would trigger the early-stopping mechanism before.

6.2.3 Metrics definition

During the experiments, it has been observed that it is optimal for some algorithms to choose high learning rates for their positive impact on the forget despite their negative impact on the utility. However, this does not align with the user preferences, which are only to sacrifice the minimal utility necessary to predict on the forget set as if the model was not trained with it.

There may be different interesting approaches to mitigate this issue. On one hand, the final metric could be reformulated to give more importance to utility than to forget. On the other hand, the very utility measure could be formualted at the light of the following. It is not the same the loss of accuracy on the forget set than on the retain set. A loss of accuracy on the forget set is natural and may indicate a

good unlearning of it. In contrast, a loss of accuracy on the retain set can indicate a potential model destruction, which should be avoided at all costs.

6.2.4 Unlearning features

Unlearning convolutional neural networks as the architecture used for the MNIST dataset is conceptually more complicated than dealing with simple fully connected neural networks for points datasets. It is so because these architectures, at least with simple constructions as our architecture, can be split into two phases: a feature construction phase plus a classification phase. It is unclear if the unlearning procedure should be applied to all the layers or just to the classification layers.

The first approach may indeed produce a model whose weights are closer than the baseline's ones, since it would also be able to change the convolutional weights. The second option, in contrast, may be of interest if we accept the idea that the construction of features depends on the dataset and not of the particular samples, so that if a model has to forget a subset, it should only forget how to classify it, still obtaining the same features of it. In this work, we have opted for the first approach since we thought is was simpler and with less potential impact on the fairness of the comparison experiments. With this impact we mean that it could be the case that an unlearning algorithm affects more some kind of layers while another one alters a different type of layers; then, fixing a subset of layers might benefit some algorithms in detriment of anothers.

6.2.5 Future work

As future work, we highlight two interesting areas to study regarding the Regret algorithm. Firstly, it is the study of the impact of selecting a certain subset of layers instead of all of them. Lastly, it could be of interest to study if the Regret's regularization loss can be reformulated in a manner which is model and dataset-agnostic.

Appendix A

Models optimization

This chapter details the optimization process performed to all three unlearning algorithms for each considered dataset. Fine-tuning and Fanchuan are optimized across the learning rate. Regret is optimized across both the learning rate and the C parameter. The optimization across C for the Regret model is reported in Annex B. For the optimal parameters configuration, the best and worst unlearned models of each algorithm are displayed, showing their decision boundary in case of $\mathbb{R}^2$ datasets or their confusion matrix for MNIST, together with their metric, forget and utility curves across the unlearn epochs.

A.1 Moons dataset

A.1.1 Optimizing Fine-tuning

The different unlearning algorithms are very sensitive to the chosen learning rate. Since they rely on different loss functions and optimization processes, it might be unfair to compare them using a given learning rate. Instead, we will compute their performance for different learning rates, and compare the best setting of each one.

The basic Fine-tuning model, as figure A.1 shows, has its best performance with learning rates quite high (of order 10^{-1}). Thus, for the range of learning rates $(0, 10^{-1})$, there is a trade-off between performance and stability. As pointed in Chapter 4, §4.5, during this optimization stage, the utility part of the metric is computed over the validation set.

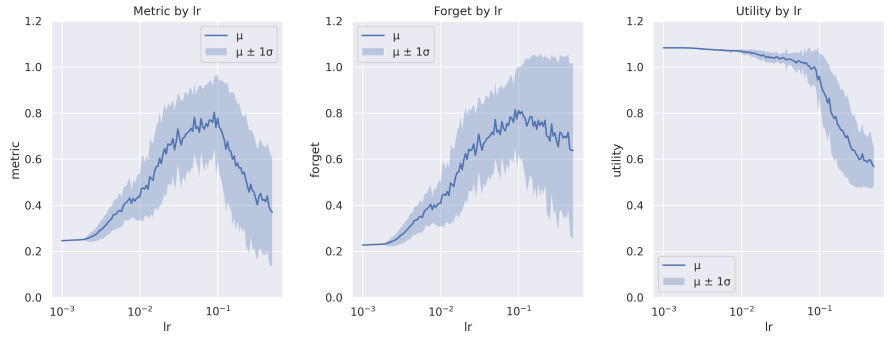


FIGURE A.1: The Fine-tuning model performance by learning rate for the moons dataset. From left to right, the values corresponding to metric, forget and utility. Horizontal axis represents the learning rate being used. The vertical axis represents the respective value. The bold line represents the average observed value. The pale band around it shows the spread of the observations. For each learning rate, the vertical distance between the blue line and each end of the band around it is the standard deviation observed. Forget is computed over the forget set and utility over the validation set.

In table A.1 we list the metrics values with its uncertainty (± one standard deviation) for a few learning rates. The Fine-tuning model achieves its best metric for a learning rate of 10^{-1} at a value of 0.75 ± 0.20.

Learning Rate	**Metric**	**Forget**	**Utility**
0.001	0.25 ± 0.00	0.23 ± 0.00	1.08 ± 0.00
0.005	0.37 ± 0.07	0.34 ± 0.06	1.07 ± 0.00
0.010	0.44 ± 0.10	0.42 ± 0.10	1.07 ± 0.01
0.050	0.72 ± 0.20	0.70 ± 0.20	1.02 ± 0.06
0.100	0.75 ± 0.20	0.81 ± 0.19	0.94 ± 0.13
0.500	0.39 ± 0.23	0.66 ± 0.36	0.59 ± 0.11

TABLE A.1: Performance metrics by learning rate for the Fine-tuning model and moons dataset. For each metric, the values are given by their observed mean ± their observed standard deviation. Forget is computed over the forget set and utility over the validation set.

For the learning rate that attained the best average result (10^{-1}), we show in figure A.2 the plots to compare the best and the worst models produced by the Fine-tuning algorithm. We plot them together with the original model and the baseline to facilitate comparison. On the right hand, we plot the metrics evolution per epochs.

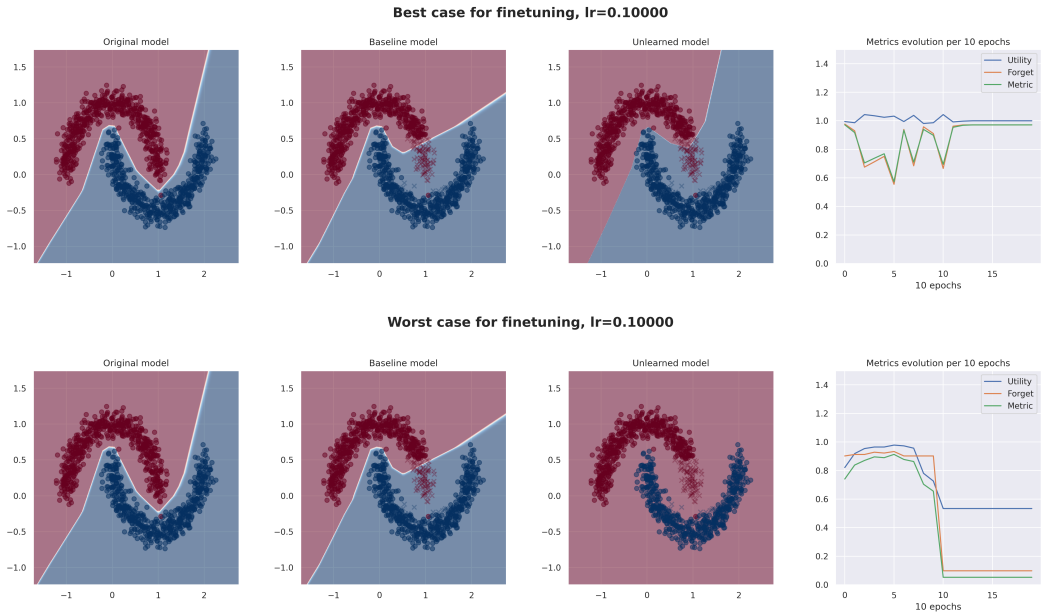


FIGURE A.2: The Fine-tuning best and worst models for a learning rate of 0.1 on the moons dataset. Above, the best performance observed at this learning rate. Below, the worst. For each case, from left to right we show: the original model, the baseline model, the unlearned model and the corresponding metrics evolution's curves per epoch.

As we can see, for such high learning rate, the Fine-tuning model can achieve a good result, but is very volatile and can also destroy the model. This can be attributed to the fact that this algorithm is not fast enough and cannot reach the optimal weights for the unlearn problems with a reasonable learning rate within the unlearn epochs limit. It then opts for higher learning rates to compensate this slowliness, but at the expense of higher uncertainty.

A.1.2 Optimizing Regret

The regret model depends on both the learning rate and the C parameter. To reduce complexity, we will start by fixing $C = 1$ and analyzing its learning rate-dependency.

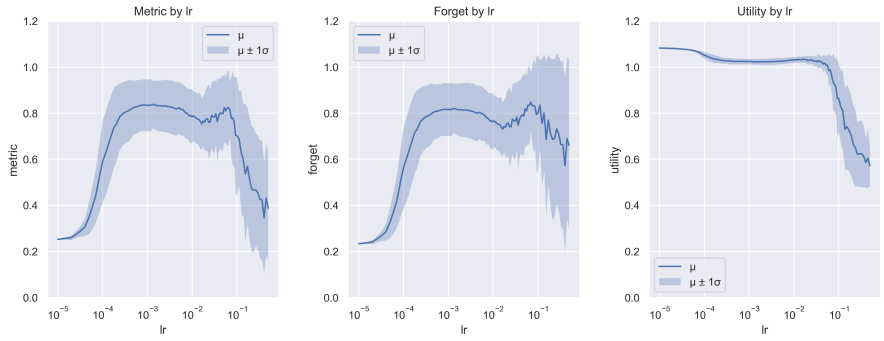


FIGURE A.3: The regret model performance by learning rate for the moons dataset. From left to right, the values corresponding to metric, forget and utility. Horizontal axis represents the learning rate being used. The vertical axis represents the respective value. The bold line represents the average observed value. The pale band around it shows the spread of the observations. For each learning rate, the vertical distance between the blue line and each end of the band around it is the standard deviation observed. Forget is computed over the forget set and utility over the validation set.

Table A.2 shows the impact of several learning rates on the metrics. For this setting, the regret algorithm achieves its maximum metric for a learning rate of 10^{-3} with a value of 0.83 ± 0.09. Note that compared with the Fine-tuning model, this metric is 11% higher while its variability is 55% lower.

Learning Rate	**Metric**	**Forget**	**Utility**
0.00001	0.25 ± 0.00	0.23 ± 0.00	1.08 ± 0.00
0.00005	0.36 ± 0.07	0.33 ± 0.07	1.07 ± 0.00
0.00010	0.60 ± 0.17	0.57 ± 0.17	1.05 ± 0.02
0.00050	0.83 ± 0.10	0.81 ± 0.10	1.03 ± 0.01
0.00100	0.83 ± 0.09	0.81 ± 0.10	1.03 ± 0.01
0.00500	0.81 ± 0.09	0.79 ± 0.10	1.03 ± 0.01
0.01000	0.77 ± 0.10	0.75 ± 0.10	1.03 ± 0.01
0.05000	0.81 ± 0.15	0.80 ± 0.16	1.01 ± 0.05
0.10000	0.72 ± 0.25	0.81 ± 0.24	0.87 ± 0.15
0.50000	0.35 ± 0.25	0.59 ± 0.39	0.58 ± 0.10

TABLE A.2: Performance metrics by learning rate for the Regret model and moons dataset. For each metric, the values are given by their observed mean ± their observed standard deviation. Forget is computed over the forget set and utility over the validation set.

Figure A.4 shows the best and worst regret models for $C = 1$ and a learning rate of 10^{-3}. As we can clearly see, the regret model can attain good results with a lower relative learning rate, compared to the Fine-tuning model. Therefore, with this learning rate, the algorithm is more stable and does not menace with destroying the model and still achieves better results.

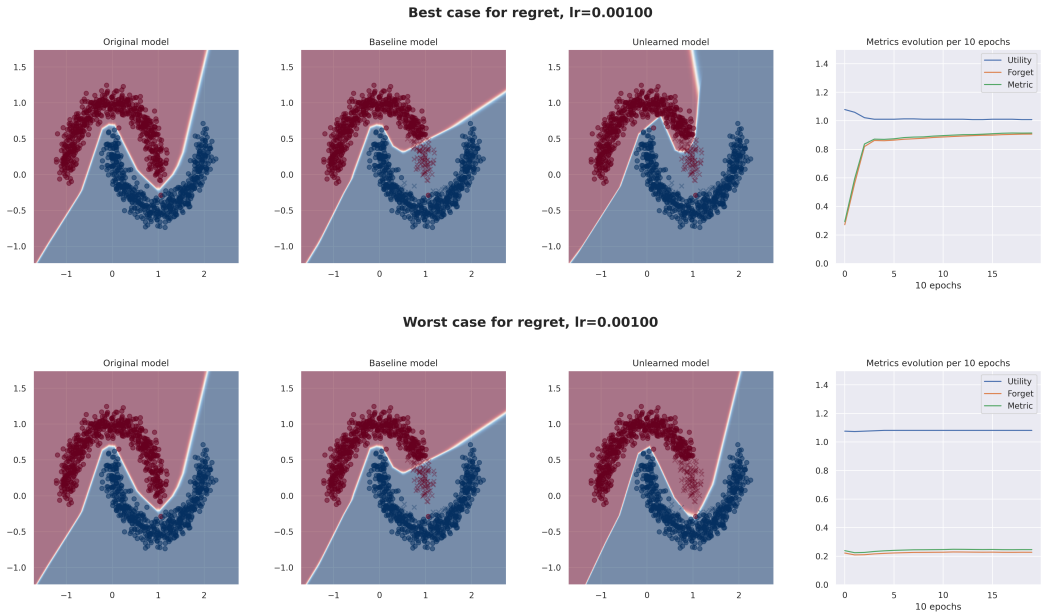


FIGURE A.4: The Regret ($C = 1$) best and worst models for a learning rate of 0.001 on the moons dataset. Above, the best performance observed at this learning rate. Below, the worst. For each case, from left to right we show: the original model, the baseline model, the unlearned model and the corresponding metrics evolution's curves per epoch.

The C dependency has been calculated in §B.1. The conclusion of this experiment is to keep $C = 1$ and the previously found learning rate.

A.1.3 Optimizing Fanchuan

According to the methodology defined in Chapter 4, §4.6, Fanchuan is optimized over the learning rate. We proceed to show its dependence on this parameter.

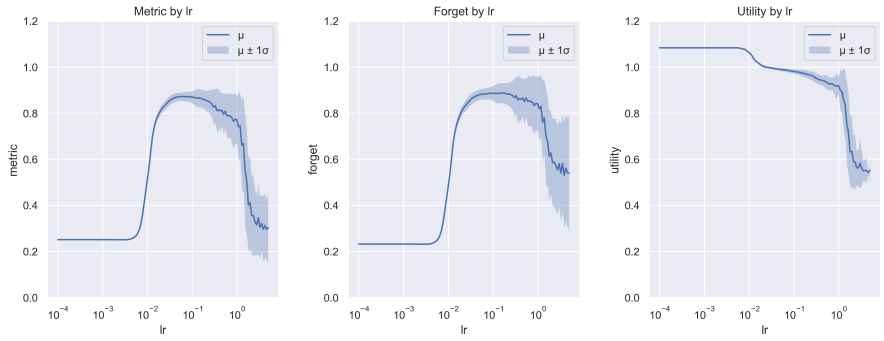


FIGURE A.5: The Fanchuan model performance by learning rate for the moons dataset. From left to right, the values corresponding to metric, forget and utility. Horizontal axis represents the learning rate being used. The vertical axis represents the respective value. The bold line represents the average observed value. The pale band around it shows the spread of the observations. For each learning rate, the vertical distance between the blue line and each end of the band around it is the standard deviation observed. Forget is computed over the forget set and utility over the validation set.

The metrics curves over the learning rate are smoother than the ones of Fine-tuning and Regret and present a lower variance.

lr	**metric**
1.0×10^{-5}	0.25 ± 0.00
5.0×10^{-5}	0.25 ± 0.00
1.0×10^{-4}	0.25 ± 0.00
5.0×10^{-4}	0.25 ± 0.00
1.0×10^{-3}	0.25 ± 0.00
5.0×10^{-3}	0.26 ± 0.00
1.0×10^{-2}	0.52 ± 0.01
5.0×10^{-2}	0.87 ± 0.02
1.0×10^{-1}	0.86 ± 0.03
5.0×10^{-1}	0.81 ± 0.08
1.0×10^{0}	0.77 ± 0.12

TABLE A.3: Performance metrics by learning rate for the Fanchuan model and moons dataset. For each metric, the values are given by their observed mean ± their observed standard deviation. Forget is computed over the forget set and utility over the validation set.

Based on these results, the best learning rate seems to be 0.05. We proceed to show in figure A.6 the best and worst Fanchuan models for this configuration.

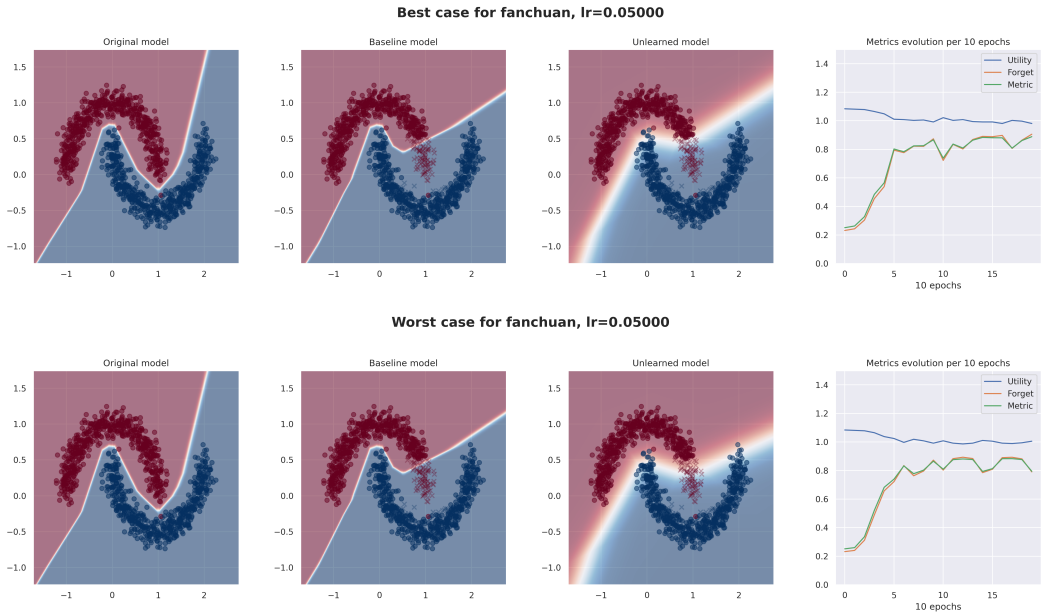


FIGURE A.6: The Fanchuan best and worst models for a learning rate of 0.05. Above, the best performance observed at this learning rate. Below, the worst. For each case, from left to right we show: the original model, the baseline model, the unlearned model and the corresponding metrics evolution's curves per epoch.

We can appreciate that the difference between the best and the worst case for the optimal learning rate differ mostly on their performance on the forget dataset, but the worst case still yielded a correct model. This is aligned with the lower uncertainty that this model generates at the optimal parameters. Also, it is worth noticing that the decision boundary after the application of the Fanchuan algorithm reflects a higher entropy, probably being the consequence of the initial step aimed at replicating a uniform distribution over the forget set.

A.2 Circles dataset

A.2.1 Optimizing Fine-tuning

We proceed to compute the Fine-tuning model's dependency on the learning rate for the circles dataset.

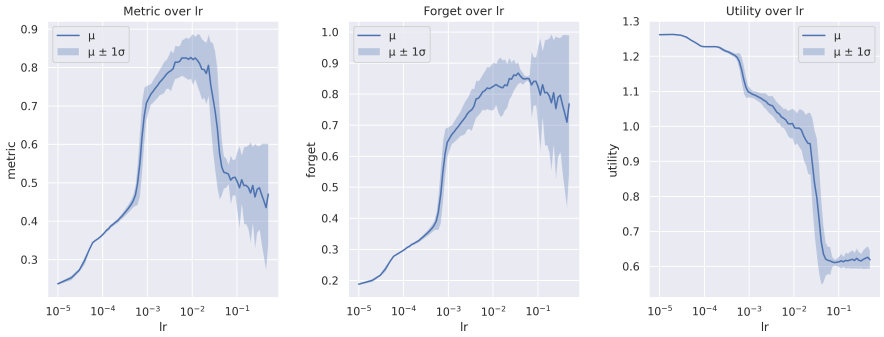


FIGURE A.7: The Fine-tuning model performance by learning rate for the circles dataset. From left to right, the values corresponding to metric, forget and utility. Horizontal axis represents the learning rate being used. The vertical axis represents the respective value. The bold line represents the average observed value. The pale band around it shows the spread of the observations. For each learning rate, the vertical distance between the blue line and each end of the band around it is the standard deviation observed. Forget is computed over the forget set and utility over the validation set.

lr	**metric**	**forget**	**utility**
1.0×10^{-4}	0.37 ± 0.00	0.30 ± 0.00	1.23 ± 0.00
5.0×10^{-4}	0.45 ± 0.02	0.38 ± 0.02	1.20 ± 0.01
1.0×10^{-3}	0.72 ± 0.03	0.66 ± 0.04	1.09 ± 0.01
5.0×10^{-3}	0.82 ± 0.05	0.80 ± 0.06	1.03 ± 0.03
1.0×10^{-2}	0.83 ± 0.06	0.82 ± 0.08	1.01 ± 0.04
5.0×10^{-2}	0.53 ± 0.04	0.85 ± 0.01	0.62 ± 0.04
1.0×10^{-1}	0.51 ± 0.06	0.84 ± 0.10	0.61 ± 0.01
5.0×10^{-1}	0.45 ± 0.16	0.73 ± 0.26	0.62 ± 0.03

TABLE A.4: Performance metrics by learning rate for the Fine-tuning model and circles dataset. For each metric, the values are given by their observed mean ± their observed standard deviation. Forget is computed over the forget set and utility over the validation set.

Based on the results shown in figure A.7 and table A.4, the peak performance is attained for a learning rate of 0.01 at a value of 0.83 ± 0.06 on the circles dataset.

Figure A.8 shows the best and worst performances of the Fine-tuning model with this learning rate on the circles dataset.

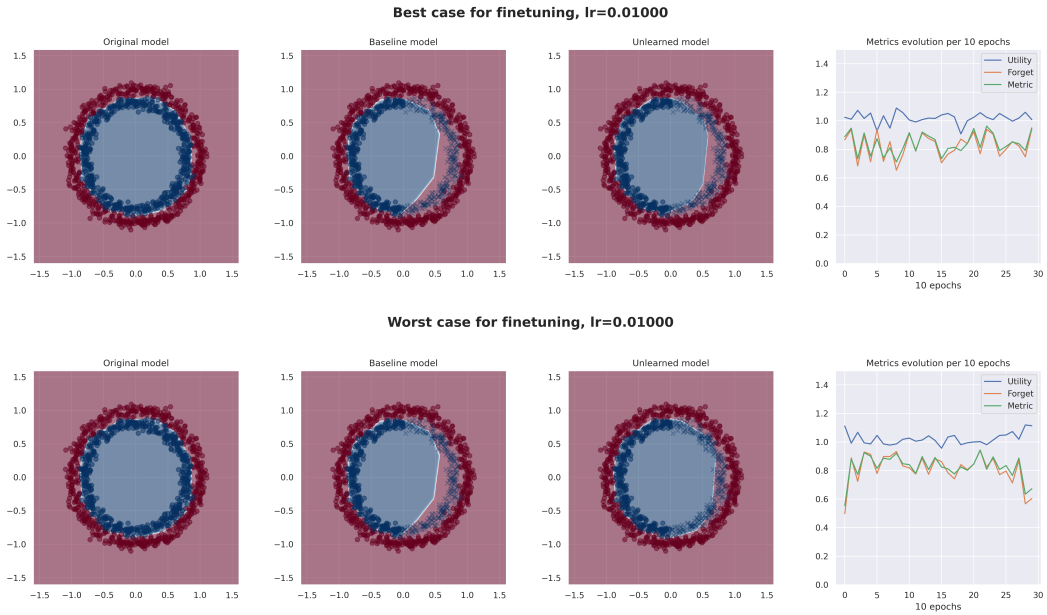


FIGURE A.8: The Fine-tuning best and worst models for a learning rate of 0.01 on the circles dataset. Above, the best performance observed at this learning rate. Below, the worst. For each case, from left to right we show: the original model, the baseline model, the unlearned model and the corresponding metrics evolution's curves per epoch.

For this dataset, Figure A.8 shows that the worst case for the optimal parameter of the Fine-tuning algorithm did not yield any model destruction. The worst executions simply show an incomplete unlearning of the forget set.

A.2.2 Optimizing Regret

Starting by a fixed $C = 1$, Figure A.9 and Table A.5 show its dependency on the learning rate for the circles dataset.

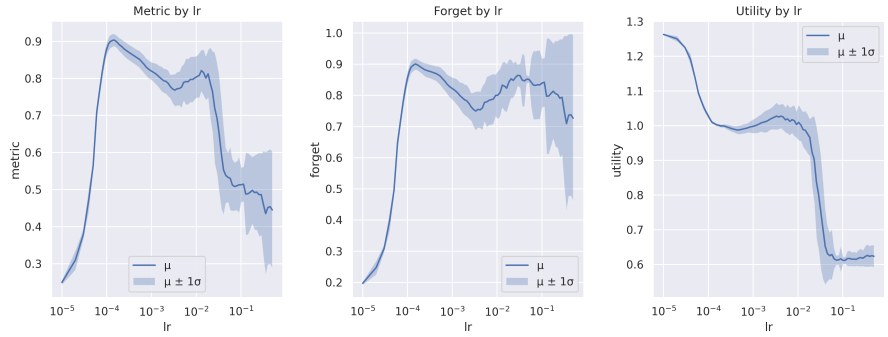


FIGURE A.9: The regret model performance by learning rate for the circles dataset. From left to right, the values corresponding to metric, forget and utility. Horizontal axis represents the learning rate being used. The vertical axis represents the respective value. The bold line represents the average observed value. The pale band around it shows the spread of the observations. For each learning rate, the vertical distance between the blue line and each end of the band around it is the standard deviation observed. Forget is computed over the forget set and utility over the validation set.

lr	**metric**	**forget**	**utility**
1.0×10^{-5}	0.24 ± 0.00	0.19 ± 0.00	1.26 ± 0.00
5.0×10^{-5}	0.57 ± 0.02	0.50 ± 0.02	1.14 ± 0.01
1.0×10^{-4}	0.88 ± 0.02	0.86 ± 0.02	1.03 ± 0.01
5.0×10^{-4}	0.86 ± 0.02	0.87 ± 0.02	0.99 ± 0.01
1.0×10^{-3}	0.82 ± 0.02	0.83 ± 0.03	1.00 ± 0.02
5.0×10^{-3}	0.78 ± 0.06	0.76 ± 0.07	1.02 ± 0.04
1.0×10^{-2}	0.80 ± 0.08	0.79 ± 0.10	1.02 ± 0.05
5.0×10^{-2}	0.52 ± 0.06	0.84 ± 0.07	0.62 ± 0.04
1.0×10^{-1}	0.51 ± 0.06	0.84 ± 0.10	0.61 ± 0.01
5.0×10^{-1}	0.44 ± 0.16	0.71 ± 0.27	0.63 ± 0.03

TABLE A.5: Performance metrics by learning rate for the Regret model and circles dataset. For each metric, the values are given by their observed mean ± their observed standard deviation. Forget is computed over the forget set and utility over the validation set.

Figure A.9 shows that the performance of the Regret model on the learning rate seems to present two local optimal points for the circles dataset. However, the one attained at a lower learning rate is preferable because it has a higher expected value and substantially less uncertainty. It is for learning rates higher than this one that the variance starts to progressively increase, indicating too high learning rates. Its best performance on the circles dataset is hereby attained at a learning rate of 0.0001 at a metric of 0.88 ± 0.02.

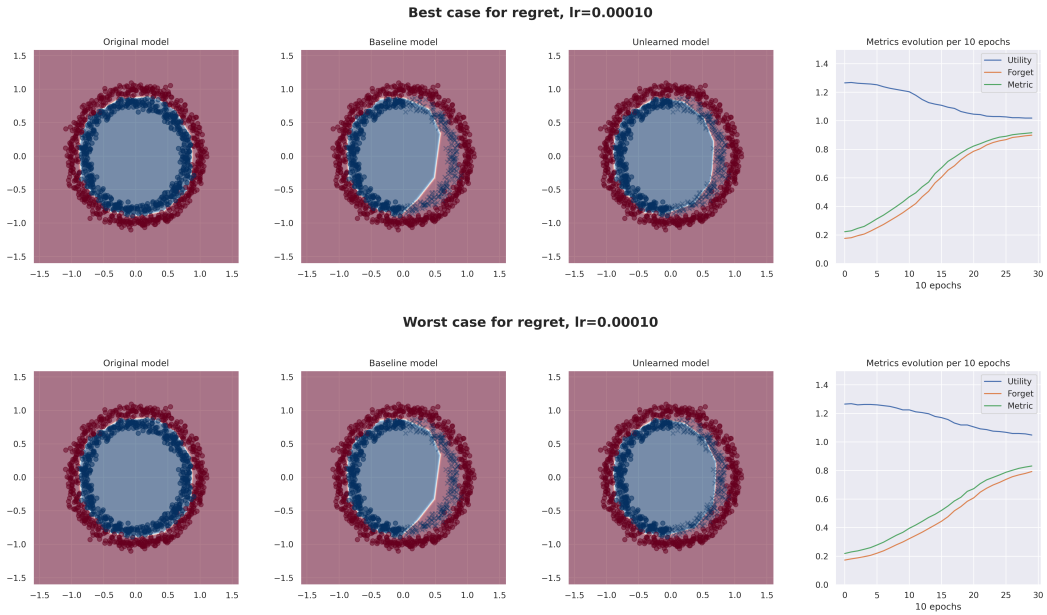


FIGURE A.10: The Regret ($C = 1$) best and worst models for a learning rate of 0.0001 on the circles dataset. Above, the best performance observed at this learning rate. Below, the worst. For each case, from left to right we show: the original model, the baseline model, the unlearned model and the corresponding metrics evolution's curves per epoch.

The best and worst unlearned models for the circles dataset shown in Figure A.10 show little difference, which is consistent with the low variance observed in the curves presented in Figure A.9.

An experiment has been conducted to determine the Regret algorithm's dependency on the C parameter for the circles dataset. The results are reported in §B.2. In this case, the conclusion was to keep $C = 1$.

A.2.3 Optimizing Fanchuan

We start by computing the Fanchuan's dependency on the learning rate.

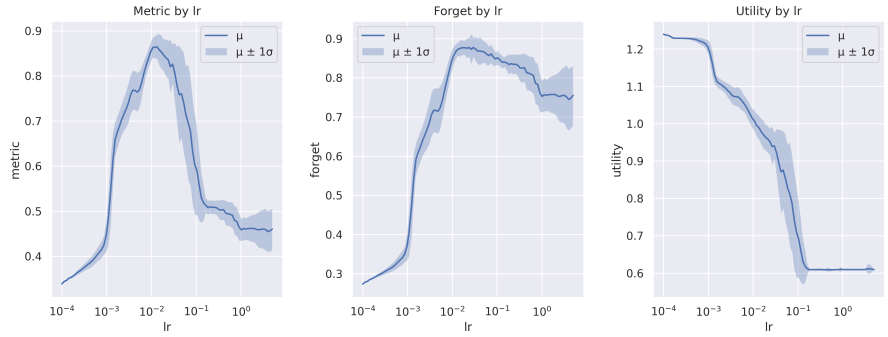


FIGURE A.11: The Fanchuan model performance by learning rate for the circles dataset. From left to right, the values corresponding to metric, forget and utility. Horizontal axis represents the learning rate being used. The vertical axis represents the respective value. The bold line represents the average observed value. The pale band around it shows the spread of the observations. For each learning rate, the vertical distance between the blue line and each end of the band around it is the standard deviation observed. Forget is computed over the forget set and utility over the validation set.

lr	**metric**	**forget**	**utility**
1.0×10^{-5}	0.24 ± 0.00	0.19 ± 0.00	1.26 ± 0.00
5.0×10^{-5}	0.28 ± 0.00	0.22 ± 0.00	1.26 ± 0.00
1.0×10^{-4}	0.34 ± 0.00	0.27 ± 0.00	1.24 ± 0.00
5.0×10^{-4}	0.39 ± 0.01	0.32 ± 0.01	1.23 ± 0.01
1.0×10^{-3}	0.46 ± 0.05	0.38 ± 0.05	1.20 ± 0.02
5.0×10^{-3}	0.77 ± 0.05	0.73 ± 0.06	1.06 ± 0.03
1.0×10^{-2}	0.86 ± 0.02	0.85 ± 0.03	1.01 ± 0.03
5.0×10^{-2}	0.76 ± 0.10	0.86 ± 0.03	0.88 ± 0.10
1.0×10^{-1}	0.59 ± 0.10	0.85 ± 0.02	0.69 ± 0.11
5.0×10^{-1}	0.50 ± 0.02	0.82 ± 0.04	0.61 ± 0.01
1.0×10^{0}	0.46 ± 0.03	0.75 ± 0.05	0.61 ± 0.01

TABLE A.6: Performance metrics by learning rate for the Fanchuan model and circles dataset. For each metric, the values are given by their observed mean ± their observed standard deviation. Forget is computed over the forget set and utility over the validation set.

Figure A.11 shows a single optimal value which also generates low variance. Learning rates higher than the optimal one produce higher variance, indicating that they are excessively high. The peak performance is achieved for a learning rate of 10^{-2}.

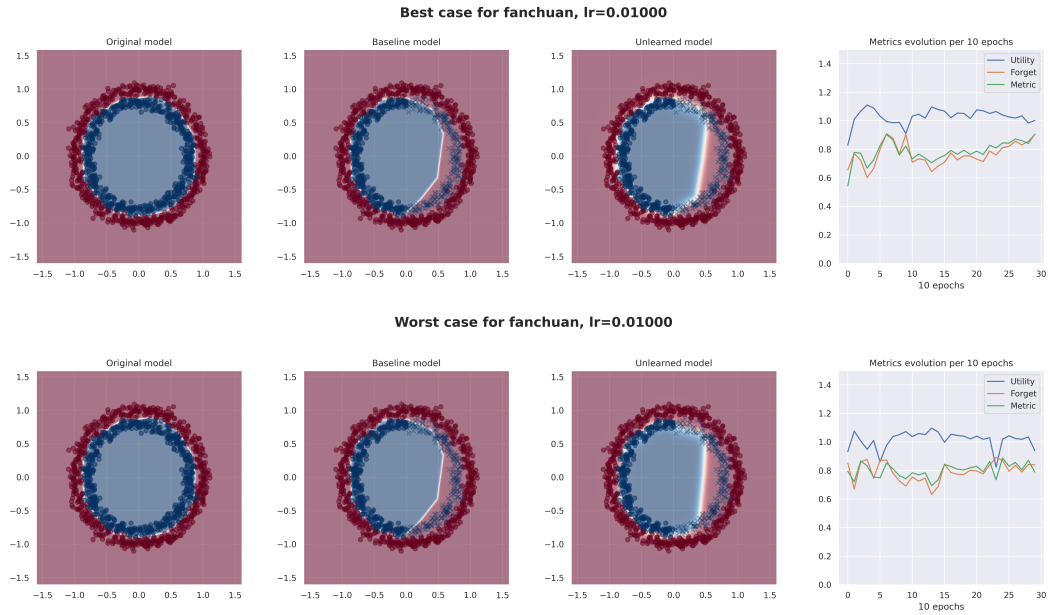


FIGURE A.12: The Fanchuan best and worst models for a learning rate of 0.01 on the circles dataset. Above, the best performance observed at this learning rate. Below, the worst. For each case, from left to right we show: the original model, the baseline model, the unlearned model and the corresponding metrics evolution's curves per epoch.

The low variance for the optimal learning rate shown in Figure A.11 is reflected on the unlearned models shown in Figure A.12, where the worst model does not differ much from the best execution. As observed in §A.1.3, the Fanchuan algorithm also produced a relatively high entropy in the decision boundary for this dataset.

A.3 Spirals dataset

A.3.1 Optimizing Fine-tuning

This section evaluates Fine-tuning model's dependency on the learning rate for the spirals dataset.

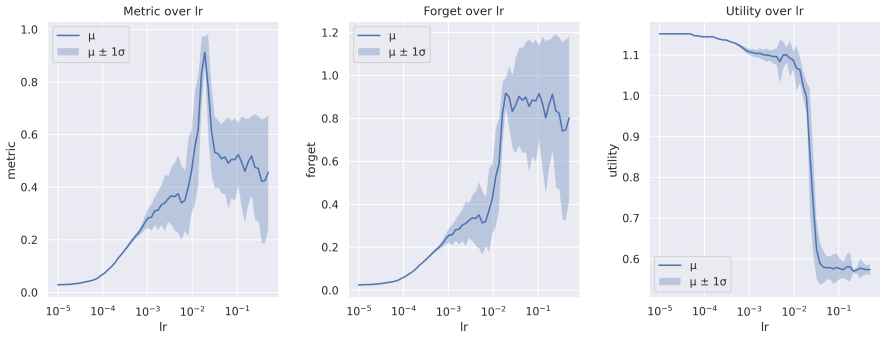


FIGURE A.13: The Fine-tuning model performance by learning rate for the spirals dataset. From left to right, the values corresponding to metric, forget and utility. Horizontal axis represents the learning rate being used. The vertical axis represents the respective value. The bold line represents the average observed value. The pale band around it shows the spread of the observations. For each learning rate, the vertical distance between the blue line and each end of the band around it is the standard deviation observed. Forget is computed over the forget set and utility over the validation set.

lr	**metric**	**forget**	**utility**
1.0×10^{-4}	0.07 ± 0.00	0.06 ± 0.00	1.14 ± 0.00
5.0×10^{-4}	0.21 ± 0.01	0.18 ± 0.01	1.13 ± 0.00
1.0×10^{-3}	0.26 ± 0.04	0.24 ± 0.03	1.11 ± 0.01
5.0×10^{-3}	0.36 ± 0.15	0.33 ± 0.15	1.10 ± 0.04
1.0×10^{-2}	0.50 ± 0.20	0.47 ± 0.21	1.08 ± 0.04
5.0×10^{-2}	0.51 ± 0.13	0.88 ± 0.25	0.58 ± 0.04
1.0×10^{-1}	0.49 ± 0.17	0.86 ± 0.31	0.58 ± 0.03
5.0×10^{-1}	0.41 ± 0.25	0.72 ± 0.43	0.58 ± 0.02

TABLE A.7: Performance metrics by learning rate for the Fine-tuning model and spirals dataset. For each metric, the values are given by their observed mean ± their observed standard deviation. Forget is computed over the forget set and utility over the validation set.

A visual exploration of the metric curve of figure A.13 shows that there is a substantial peak which is so concentrated that it is not reflected in the table. In the light of the above, the data was manually explored and the best argument turned out to be a learning rate of around 0.02. At such value, the metric was 0.91 ± 0.06.

Figure A.14 shows the best and worst performances of the Fine-tuning algorithm with the optimal learning rate on the spirals dataset.

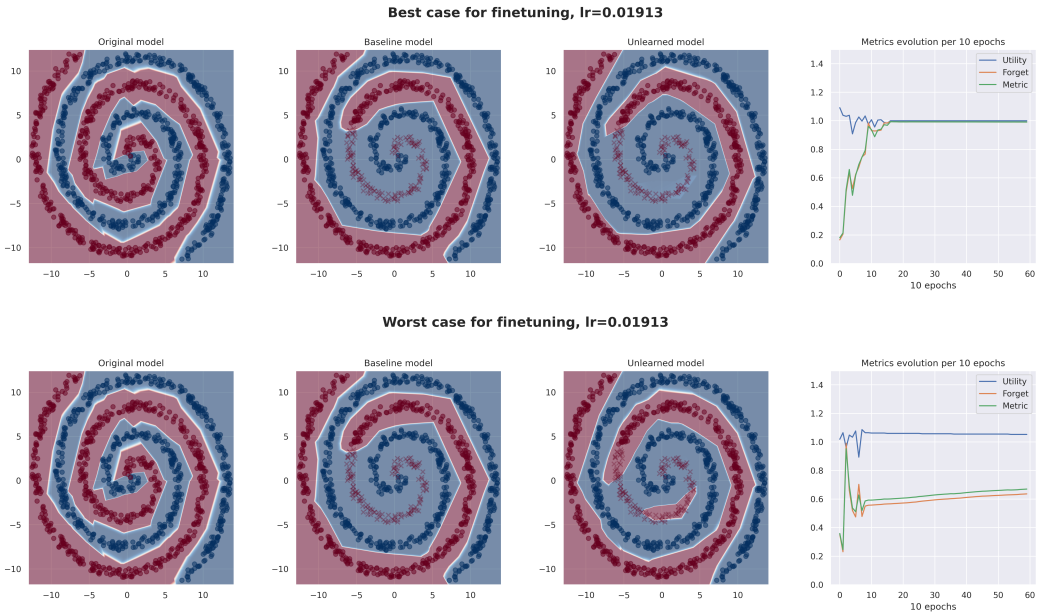


FIGURE A.14: The Fine-tuning best and worst models for a learning rate of 0.02 on the spirals dataset. Above, the best performance observed at this learning rate. Below, the worst. For each case, from left to right we show: the original model, the baseline model, the unlearned model and the corresponding metrics evolution's curves per epoch.

For this dataset, similarly than in §A.2.1, Figure A.14 shows that the worst case for the optimal parameter of the Fine-tuning algorithm did not ended in model destruction but in an incomplete unlearning of the forget set.

A.3.2 Optimizing Regret

We proceed to study the Regret's dependency on the learning rate for the spirals dataset. Figure A.15 and Table A.8 show the results of this experiment.

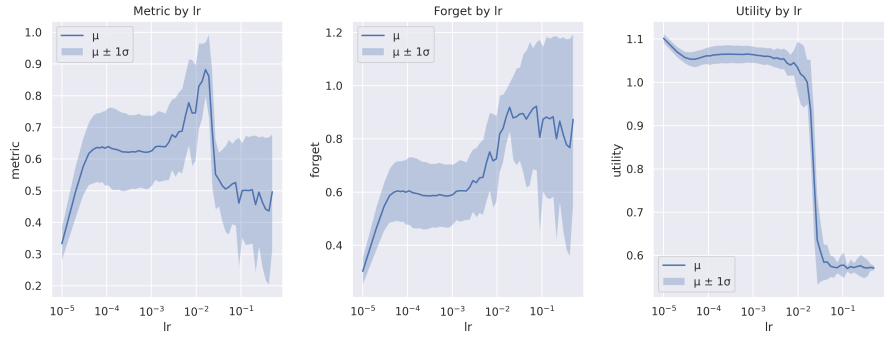


FIGURE A.15: The regret model performance by learning rate for the spirals dataset. From left to right, the values corresponding to metric, forget and utility. Horizontal axis represents the learning rate being used. The vertical axis represents the respective value. The bold line represents the average observed value. The pale band around it shows the spread of the observations. For each learning rate, the vertical distance between the blue line and each end of the band around it is the standard deviation observed. Forget is computed over the forget set and utility over the validation set.

lr	metric	forget	utility
1.0×10^{-5}	0.27 ± 0.03	0.24 ± 0.03	1.11 ± 0.01
5.0×10^{-5}	0.65 ± 0.11	0.62 ± 0.12	1.05 ± 0.02
1.0×10^{-4}	0.67 ± 0.12	0.64 ± 0.12	1.06 ± 0.02
5.0×10^{-4}	0.68 ± 0.11	0.64 ± 0.12	1.06 ± 0.02
1.0×10^{-3}	0.67 ± 0.10	0.63 ± 0.11	1.06 ± 0.02
5.0×10^{-3}	0.72 ± 0.13	0.69 ± 0.14	1.05 ± 0.03
1.0×10^{-2}	0.80 ± 0.12	0.78 ± 0.12	1.03 ± 0.05
5.0×10^{-2}	0.51 ± 0.14	0.88 ± 0.26	0.58 ± 0.04
1.0×10^{-1}	0.49 ± 0.17	0.86 ± 0.31	0.58 ± 0.03
5.0×10^{-1}	0.46 ± 0.21	0.81 ± 0.37	0.57 ± 0.01

TABLE A.8: Performance metrics by learning rate for the Regret model and spirals dataset. For each metric, the values are given by their observed mean ± their observed standard deviation. Forget is computed over the forget set and utility over the validation set.

We see in Figure A.15 a narrow peak after a learning rate of 10−2. A manual exploration of the data file reveals that the best performing learning rate was 0.02.

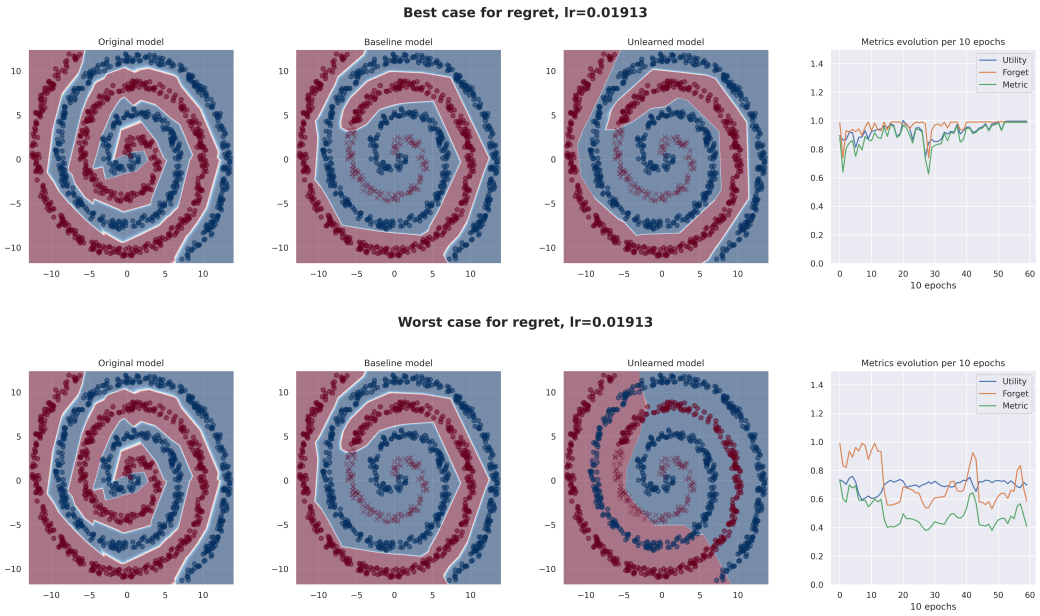


FIGURE A.16: The Regret ($C = 1$) best and worst models for a learning rate of 0.01913 on the spirals dataset. Above, the best performance observed at this learning rate. Below, the worst. For each case, from left to right we show: the original model, the baseline model, the unlearned model and the corresponding metrics evolution's curves per epoch.

The worst model for the spirals dataset, shown in Figure A.16 shows model destruction while the best one really fits the retain dataset. This is consistent with the high spread shown in Figure A.15.

An experiment has been conducted to determine the Regret algorithm's dependency on the C parameter for this dataset. The results are reported in §B.3. In this case, the conclusion was to keep $C = 1$.

A.3.3 Optimizing Fanchuan

This section details the optimization process for the Fanchuan algorithm with respect to the learning rate for the spirals dataset.

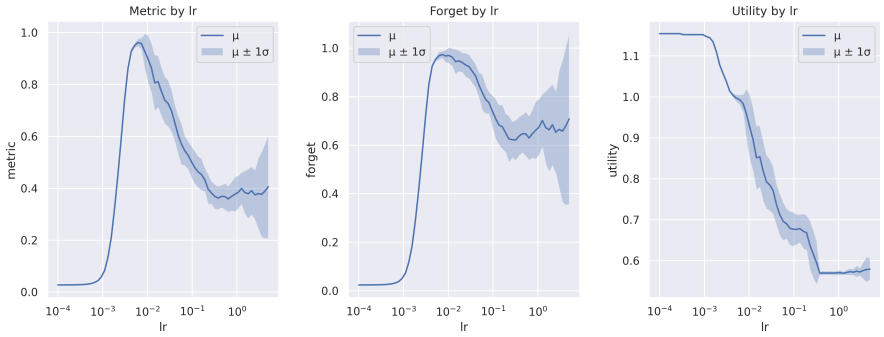


FIGURE A.17: The Fanchuan model performance by learning rate for the spirals dataset. From left to right, the values corresponding to metric, forget and utility. Horizontal axis represents the learning rate being used. The vertical axis represents the respective value. The bold line represents the average observed value. The pale band around it shows the spread of the observations. For each learning rate, the vertical distance between the blue line and each end of the band around it is the standard deviation observed. Forget is computed over the forget set and utility over the validation set.

lr	**metric**	**forget**	**utility**
1.0×10^{-5}	0.03 ± 0.00	0.02 ± 0.00	1.15 ± 0.00
5.0×10^{-5}	0.03 ± 0.00	0.02 ± 0.00	1.15 ± 0.00
1.0×10^{-4}	0.03 ± 0.00	0.02 ± 0.00	1.15 ± 0.00
5.0×10^{-4}	0.03 ± 0.00	0.03 ± 0.00	1.15 ± 0.00
1.0×10^{-3}	0.07 ± 0.00	0.06 ± 0.00	1.15 ± 0.00
5.0×10^{-3}	0.94 ± 0.02	0.95 ± 0.01	1.00 ± 0.02
1.0×10^{-2}	0.92 ± 0.07	0.98 ± 0.03	0.94 ± 0.06
5.0×10^{-2}	0.60 ± 0.08	0.84 ± 0.07	0.71 ± 0.05
1.0×10^{-1}	0.50 ± 0.07	0.73 ± 0.08	0.68 ± 0.04
5.0×10^{-1}	0.37 ± 0.05	0.64 ± 0.09	0.57 ± 0.01
1.0×10^{0}	0.37 ± 0.06	0.64 ± 0.11	0.57 ± 0.01

TABLE A.9: Performance metrics by learning rate for the Fanchuan model and spirals dataset. For each metric, the values are given by their observed mean ± their observed standard deviation. Forget is computed over the forget set and utility over the validation set.

Similarly to the results seen in §A.2.3, the Fanchuan algorithm yields smoother learning rate-dependencies and substantially lower variances compared to Fine-tuning and Regret. The optimal learning rate generates low variance and learning rates higher than this one produce higher deviations. The best performance is attained for a learning rate of 5×10^{-3}.

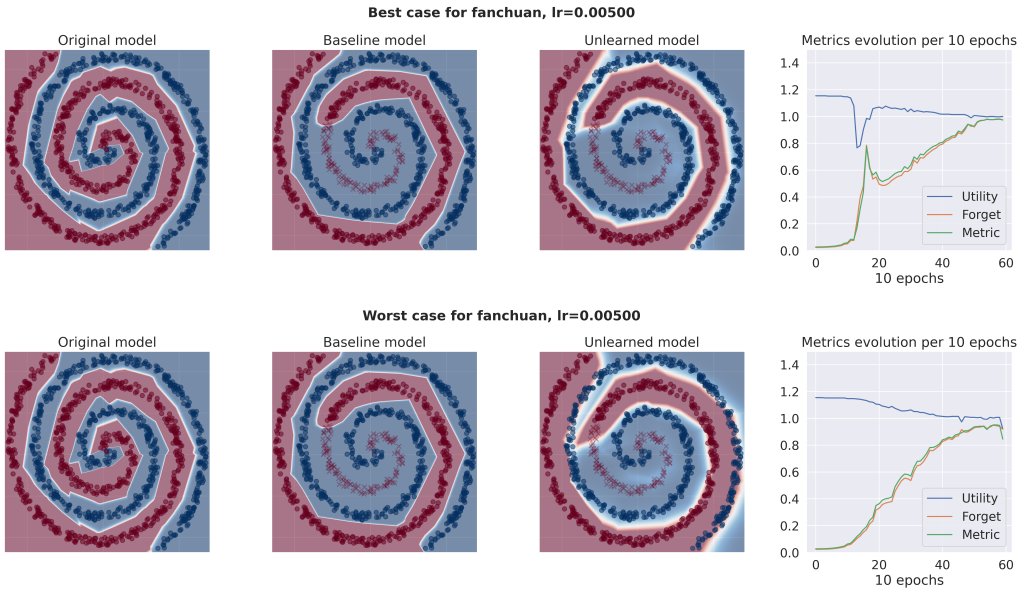


FIGURE A.18: The Fanchuan best and worst models for a learning rate of 0.005 on the spirals dataset. Above, the best performance observed at this learning rate. Below, the worst. For each case, from left to right we show: the original model, the baseline model, the unlearned model and the corresponding metrics evolution's curves per epoch.

The best and worst models for the optimal learning rate presented in Figure A.18 show a similar behaviour than the ones reported in §A.2.3, with low difference between them, aligned with the curves seen in Figure A.17. Also, as in the moons and circles executions of the Fanchuan algorithm, the decision boundary presents high entropy, meaning that the model is uncertain about its predictions on a band around this decision frontier.

A.4 MNIST dataset

A.4.1 Optimizing Fine-tuning

We start by studying the dependence on the learning rate by the Fine-tuning model. Since the MNIST optimization is much more costly, we cannot compute a curve as in the toy datasets section. We will plot the mean metrics with their associated confidence ranges for the learning rates in $\{10^{-n}\}_{n=1}^{8}$.

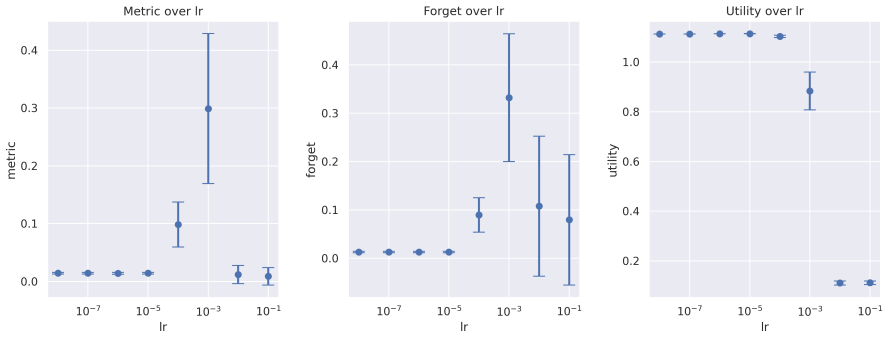


FIGURE A.19: The Fine-tuning model performance by learning rate for the MNIST dataset. From left to right, the values corresponding to metric, forget and utility. Horizontal axis represents the learning rate being used. The vertical axis represents the respective value. For each considered learning rate, each metrics value is represented by a dot on the observed average value together with confidence bars representing one standard deviation above and below. Forget is computed over the forget set and utility over the validation set.

lr	**metric**	**forget**	**utility**
1.0×10^{-8}	0.01 ± 0.00	0.01 ± 0.00	1.11 ± 0.00
1.0×10^{-7}	0.01 ± 0.00	0.01 ± 0.00	1.11 ± 0.00
1.0×10^{-6}	0.01 ± 0.00	0.01 ± 0.00	1.11 ± 0.00
1.0×10^{-5}	0.01 ± 0.00	0.01 ± 0.00	1.11 ± 0.00
1.0×10^{-4}	0.10 ± 0.04	0.09 ± 0.04	1.10 ± 0.00
1.0×10^{-3}	0.30 ± 0.13	0.33 ± 0.13	0.88 ± 0.08
1.0×10^{-2}	0.01 ± 0.02	0.11 ± 0.14	0.11 ± 0.01
1.0×10^{-1}	0.01 ± 0.02	0.08 ± 0.13	0.11 ± 0.01

TABLE A.10: Performance metrics by learning rate for the Fine-tuning model and MNIST dataset. For each metric, the values are given by their observed mean $\pm$ their observed standard deviation. Forget is computed over the forget set and utility over the validation set.

In this limited set of learning rates, the optimal one seems to be 10^{-3}. For this learning rate, we show in figure A.20 the best and worst results obtained for it. Note that we now plot the evolution of forget and metric with confidence bands since they depend on not one but 6 baselines.

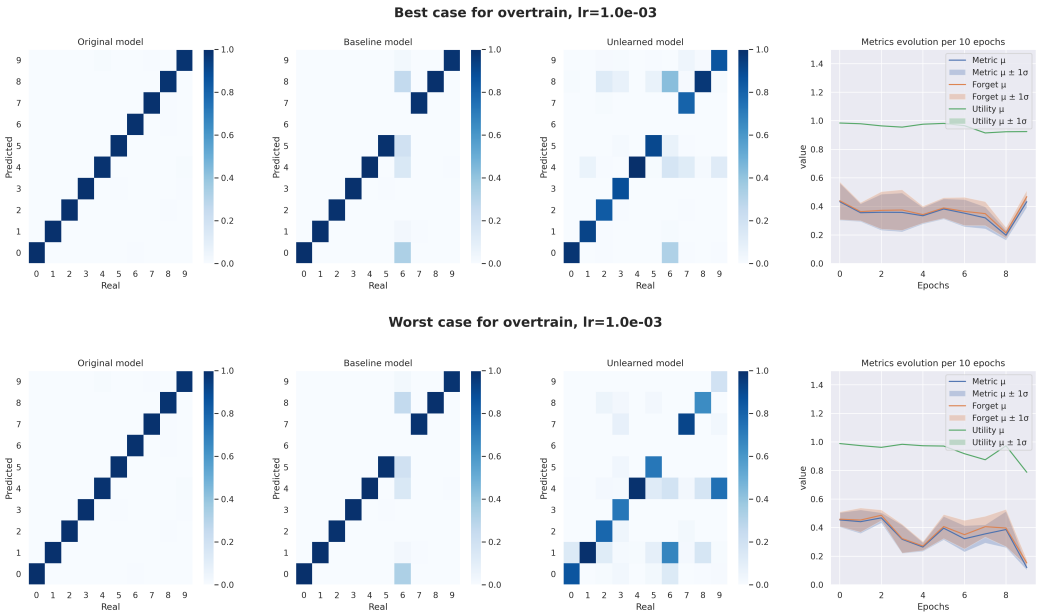


FIGURE A.20: The Fine-tuning best and worst models for a learning rate of 0.001 on the MNIST dataset. Above, the best performance observed at this learning rate. Below, the worst. For each case, from left to right we show: the original model, the baseline model, the unlearned model and the corresponding metrics evolution's curves per epoch.

A.4.2 Optimizing Regret

Setting $C = 1$, we start by computing its dependency on the learning rate.

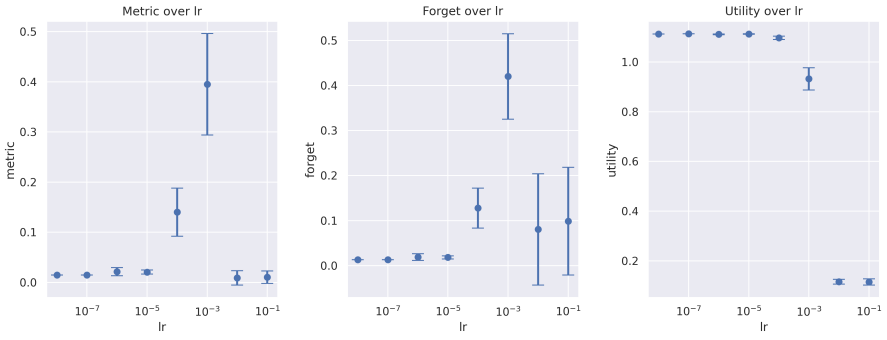


FIGURE A.21: The Regret model ($C = 1$) performance by learning rate for the MNIST dataset. From left to right, the values corresponding to metric, forget and utility. Horizontal axis represents the learning rate being used. The vertical axis represents the respective value. For each considered learning rate, each metrics value is represented by a dot on the observed average value together with confidence bars representing one standard deviation above and below. Forget is computed over the forget set and utility over the validation set.

lr	metric	forget	utility
1.0×10^{-8}	0.01 ± 0.00	0.01 ± 0.00	1.11 ± 0.00
1.0×10^{-7}	0.01 ± 0.00	0.01 ± 0.00	1.11 ± 0.00
1.0×10^{-6}	0.02 ± 0.01	0.02 ± 0.01	1.11 ± 0.00
1.0×10^{-5}	0.02 ± 0.00	0.02 ± 0.00	1.11 ± 0.00
1.0×10^{-4}	0.14 ± 0.05	0.13 ± 0.04	1.10 ± 0.01
1.0×10^{-3}	0.40 ± 0.10	0.42 ± 0.09	0.93 ± 0.04
1.0×10^{-2}	0.01 ± 0.01	0.08 ± 0.12	0.12 ± 0.01
1.0×10^{-1}	0.01 ± 0.01	0.10 ± 0.12	0.11 ± 0.01

TABLE A.11: Performance metrics by learning rate for regret model and MNIST dataset. For each metric, the values are given by their observed mean ± their observed standard deviation. Forget is computed over the forget set and utility over the validation set.

This experiment indicates that the best learning rate may be 10^{-3}. Figure A.22 shows the best and worst performances attained at this learning rate for this experiment.

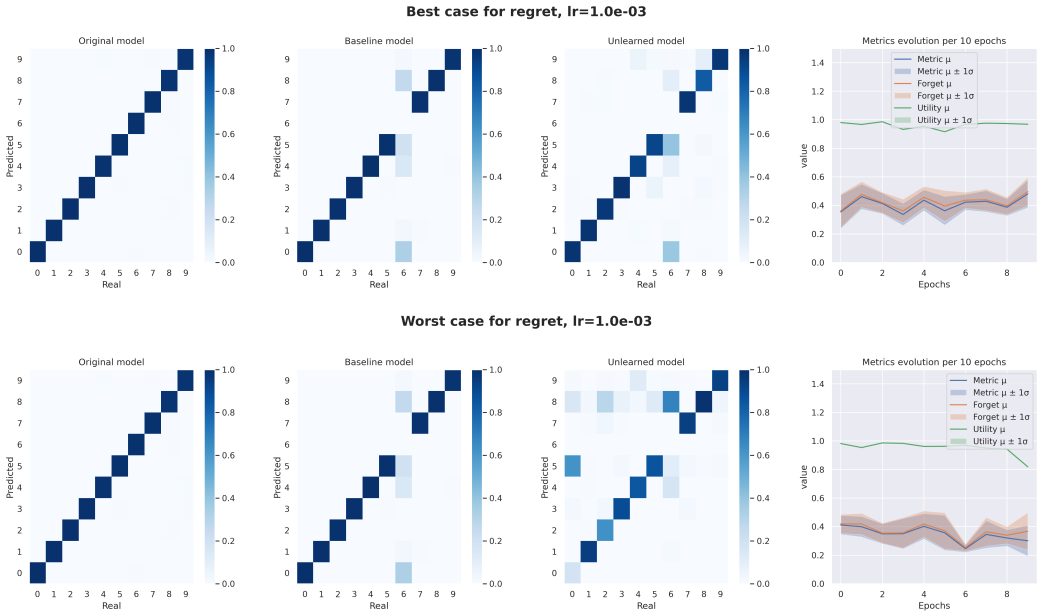


FIGURE A.22: The Regret ($C = 1$) best and worst models for a learning rate of 0.001 on the MNIST dataset. Above, the best performance observed at this learning rate. Below, the worst. For each case, from left to right we show: the original model, the baseline model, the unlearned model and the corresponding metrics evolution's curves per epoch.

An experiment has been conducted to determine the dependency on the C parameter and is reported in §B.4. It did not find substantial evidence sustaining that any C value different than 1 would improve the results, so the original $C = 1$ value was kept.

A.4.3 Optimizing Fanchuan

We now turn our attention to the Fanchuan model. We analyze its dependency on the learning rate and display it in Figure A.23 and Table A.12.

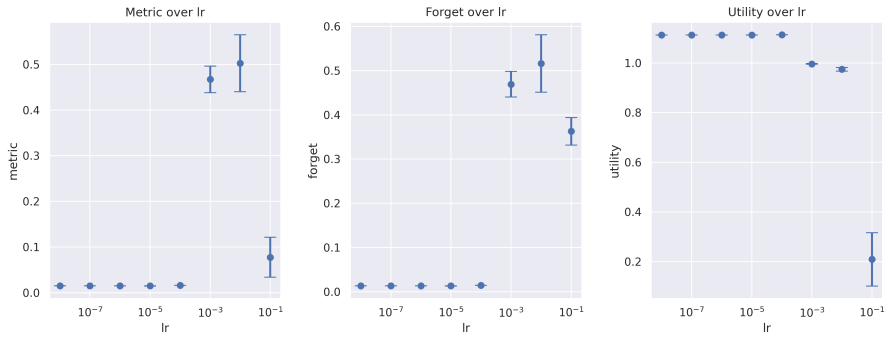


FIGURE A.23: The Fanchuan model performance by learning rate for the MNIST dataset. From left to right, the values corresponding to metric, forget and utility. Horizontal axis represents the learning rate being used. The vertical axis represents the respective value. For each considered learning rate, each metrics value is represented by a dot on the observed average value together with confidence bars representing one standard deviation above and below. Forget is computed over the forget set and utility over the validation set.

lr	metric	forget	utility
1.0×10^{-8}	0.01 ± 0.00	0.01 ± 0.00	1.11 ± 0.00
1.0×10^{-7}	0.01 ± 0.00	0.01 ± 0.00	1.11 ± 0.00
1.0×10^{-6}	0.01 ± 0.00	0.01 ± 0.00	1.11 ± 0.00
1.0×10^{-5}	0.01 ± 0.00	0.01 ± 0.00	1.11 ± 0.00
1.0×10^{-4}	0.02 ± 0.00	0.01 ± 0.00	1.11 ± 0.00
1.0×10^{-3}	0.47 ± 0.03	0.47 ± 0.03	1.00 ± 0.00
1.0×10^{-2}	0.50 ± 0.06	0.52 ± 0.07	0.97 ± 0.01
1.0×10^{-1}	0.08 ± 0.04	0.36 ± 0.03	0.21 ± 0.11

TABLE A.12: Performance metrics by learning rate for the Fanchuan model and MNIST dataset. For each metric, the values are given by their observed mean $\pm$ their observed standard deviation. Forget is computed over the forget set and utility over the validation set.

This indicates that the optimal learning rate for this discretization is 10^{-2}. Figure A.24 shows the best and worst performance attained by the Fanchuan model with this learning rate in our experiment.

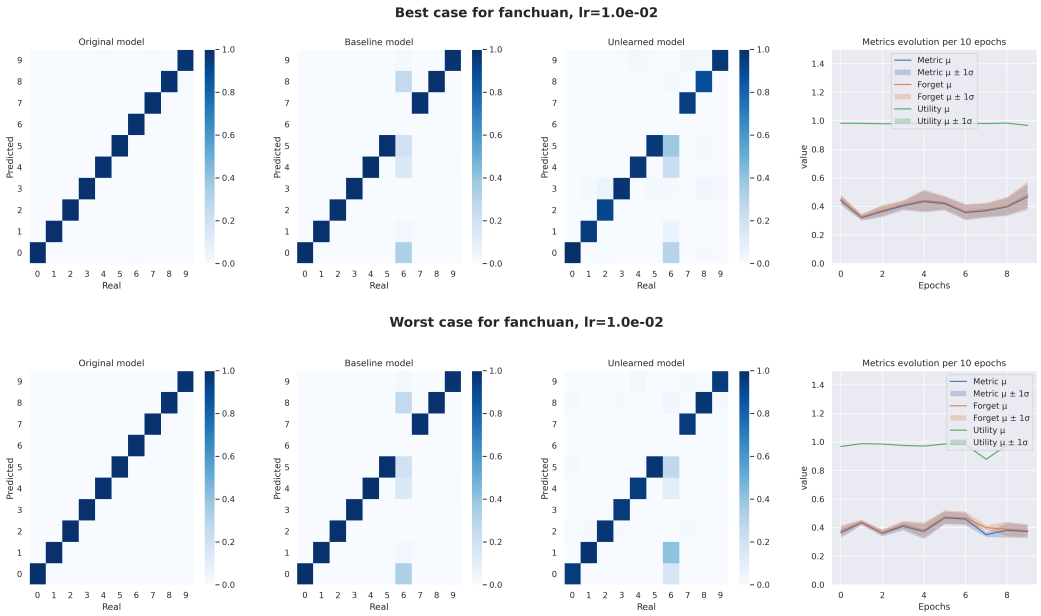


FIGURE A.24: The Fanchuan best and worst models for a learning rate of 0.01 on the MNIST dataset. Above, the best performance observed at this learning rate. Below, the worst. For each case, from left to right we show: the original model, the baseline model, the unlearned model and the corresponding metrics evolution's curves per epoch.

Appendix B

Studying C dependencies

This chapter analyses the dependency of the Regret model on its C parameter for each dataset.

B.1 On the moons dataset

We now proceed to calculate the dependency on both the learning rate and the C parameter. Results will be shown in a heatmap and a table.

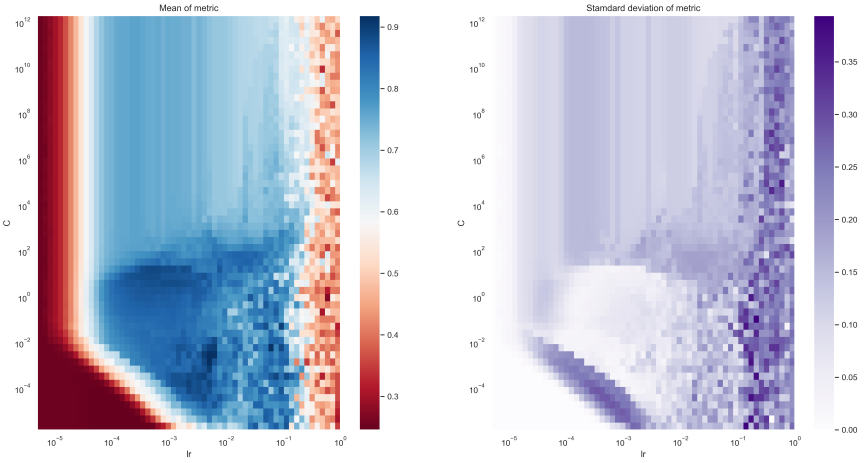


FIGURE B.1: Regret model's metric across C and learning rate on the moons dataset. From left to right: the mean value and the standard deviation. For each plot, the horizontal axis represents the learning rate and the vertical line represents the C parameter. Each point on this heatmap corresponds to the respective observed value for the combination (learning rate, C).

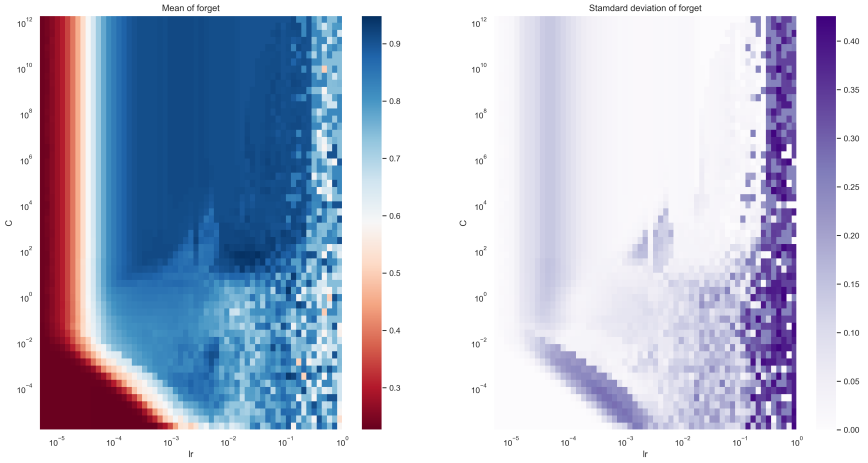


FIGURE B.2: Regret model's forget across C and learning rate on the moons dataset. From left to right: the mean value and the standard deviation. For each plot, the horizontal axis represents the learning rate and the vertical line represents the C parameter. Each point on this heatmap corresponds to the respective observed value for the combination (learning rate, C).

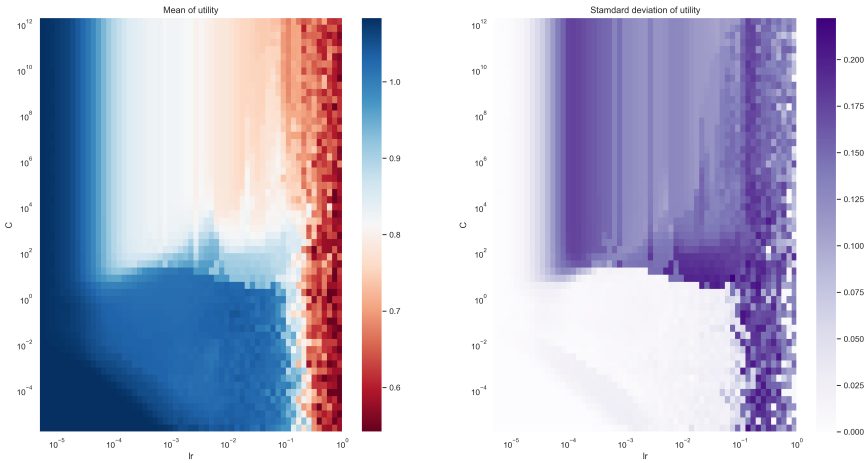


FIGURE B.3: Regret model's utility across C and learning rate on the moons dataset. From left to right: the mean value and the standard deviation. For each plot, the horizontal axis represents the learning rate and the vertical line represents the C parameter. Each point on this heatmap corresponds to the respective observed value for the combination (learning rate, C).

In the three Figures B.1, B.2 and B.3, we observe an assymptotic behaviour as C increases above 10^3. Uncertainty increases as the learning rate surpasses 10^{-1}. The utility heatmap represented in Figure B.3 follows an opposite pattern as the forget heatmap of Figure B.2. As expected, higher C's improve the forget at the expense of

lowering the utility. It is interesting to notice the lower region on which both forget and utility realize good enough values resulting in the best area for the overall metric since it reaches the best compromise between the former two. Next, Table B.1 reports the exact average and standard deviation values at some points.

lr	C	metric
1.0×10^{-5}	0	0.25 ± 0.00
	1.0×10^{-2}	0.25 ± 0.00
	1.0×10^{0}	0.25 ± 0.00
	5.0×10^{1}	0.25 ± 0.00
	1.0×10^{2}	0.25 ± 0.00
	5.0×10^{2}	0.25 ± 0.00
	1.0×10^{3}	0.25 ± 0.00
	1.0×10^{4}	0.25 ± 0.00
	1.0×10^{6}	0.25 ± 0.00
5.0×10^{-5}	0	0.25 ± 0.00
	1.0×10^{-2}	0.26 ± 0.02
	1.0×10^{0}	0.35 ± 0.09
	5.0×10^{1}	0.35 ± 0.09
	1.0×10^{2}	0.35 ± 0.09
	5.0×10^{2}	0.35 ± 0.09
	1.0×10^{3}	0.35 ± 0.09
	1.0×10^{4}	0.35 ± 0.09
	1.0×10^{6}	0.35 ± 0.09
1.0×10^{-4}	0	0.25 ± 0.00
	1.0×10^{-2}	0.44 ± 0.14
	1.0×10^{0}	0.56 ± 0.21
	5.0×10^{1}	0.55 ± 0.21
	1.0×10^{2}	0.55 ± 0.21
	5.0×10^{2}	0.55 ± 0.21
	1.0×10^{3}	0.56 ± 0.21
	1.0×10^{4}	0.56 ± 0.21
	1.0×10^{6}	0.56 ± 0.21
5.0×10^{-4}	0	0.25 ± 0.00
	1.0×10^{-2}	0.68 ± 0.23
	1.0×10^{0}	0.76 ± 0.19
	5.0×10^{1}	0.71 ± 0.20
	1.0×10^{2}	0.68 ± 0.20
	5.0×10^{2}	0.67 ± 0.19
	1.0×10^{3}	0.67 ± 0.19
	1.0×10^{4}	0.67 ± 0.19
	1.0×10^{6}	0.67 ± 0.19
1.0×10^{-3}	0	0.25 ± 0.00
	1.0×10^{-2}	0.71 ± 0.24
	1.0×10^{0}	0.76 ± 0.19
	5.0×10^{1}	0.78 ± 0.21
	1.0×10^{2}	0.74 ± 0.19
	5.0×10^{2}	0.70 ± 0.19
	1.0×10^{3}	0.69 ± 0.19
	1.0×10^{4}	0.69 ± 0.19
	1.0×10^{6}	0.69 ± 0.19

TABLE B.1: Mean and standard deviation of metric grouped by lr, C

lr	C	metric
5.0×10^{-3}	0	0.36 ± 0.08
	1.0×10^{-2}	0.76 ± 0.21
	1.0×10^{0}	0.77 ± 0.16
	5.0×10^{1}	0.80 ± 0.20
	1.0×10^{2}	0.78 ± 0.21
	5.0×10^{2}	0.76 ± 0.22
	1.0×10^{3}	0.74 ± 0.21
	1.0×10^{4}	0.71 ± 0.19
	1.0×10^{6}	0.70 ± 0.19
1.0×10^{-2}	0	0.47 ± 0.15
	1.0×10^{-2}	0.78 ± 0.14
	1.0×10^{0}	0.75 ± 0.11
	5.0×10^{1}	0.82 ± 0.22
	1.0×10^{2}	0.82 ± 0.22
	5.0×10^{2}	0.77 ± 0.21
	1.0×10^{3}	0.76 ± 0.22
	1.0×10^{4}	0.70 ± 0.19
	1.0×10^{6}	0.69 ± 0.19
5.0×10^{-2}	0	0.74 ± 0.20
	1.0×10^{-2}	0.83 ± 0.12
	1.0×10^{0}	0.85 ± 0.10
	5.0×10^{1}	0.79 ± 0.25
	1.0×10^{2}	0.81 ± 0.24
	5.0×10^{2}	0.80 ± 0.23
	1.0×10^{3}	0.79 ± 0.23
	1.0×10^{4}	0.76 ± 0.22
	1.0×10^{6}	0.73 ± 0.22
1.0×10^{-1}	0	0.76 ± 0.22
	1.0×10^{-2}	0.81 ± 0.15
	1.0×10^{0}	0.75 ± 0.21
	5.0×10^{1}	0.77 ± 0.24
	1.0×10^{2}	0.75 ± 0.23
	5.0×10^{2}	0.76 ± 0.23
	1.0×10^{3}	0.75 ± 0.23
	1.0×10^{4}	0.73 ± 0.23
	1.0×10^{6}	0.73 ± 0.23
5.0×10^{-1}	0	0.39 ± 0.24
	1.0×10^{-2}	0.35 ± 0.30
	1.0×10^{0}	0.41 ± 0.22
	5.0×10^{1}	0.40 ± 0.21
	1.0×10^{2}	0.39 ± 0.24
	5.0×10^{2}	0.41 ± 0.23
	1.0×10^{3}	0.45 ± 0.24
	1.0×10^{4}	0.37 ± 0.29
	1.0×10^{6}	0.48 ± 0.21

TABLE B.2: (continues) for the Regret model on the moons dataset.

These results suggest that a configuration close to the optimum in the studied region is given by a learning rate of 0.05 and $C = 1$ (see Table B.1, second column, row corresponding to 5.0×10^{-2}). Since the optimal C turns to be 1, this is the selected C but the learning rate is chosen according to the results given in table A.2 given in Chapter 5, §A.1.2. We do so because in this grid-search experiment we generate 10 samples for each configuration. In contrast, the experiment reported in Chapter 5, §A.1.2 trains 100 models for each configuration. Therefore, we are more confident of these later results. This leads to a final learning rate of 0.001 and thus to the model configuration shown in figure A.4.

B.2 On the circles dataset

We now present the heatmaps showing the relation between the metrics and both the learning rate and the C parameter for the circles dataset.

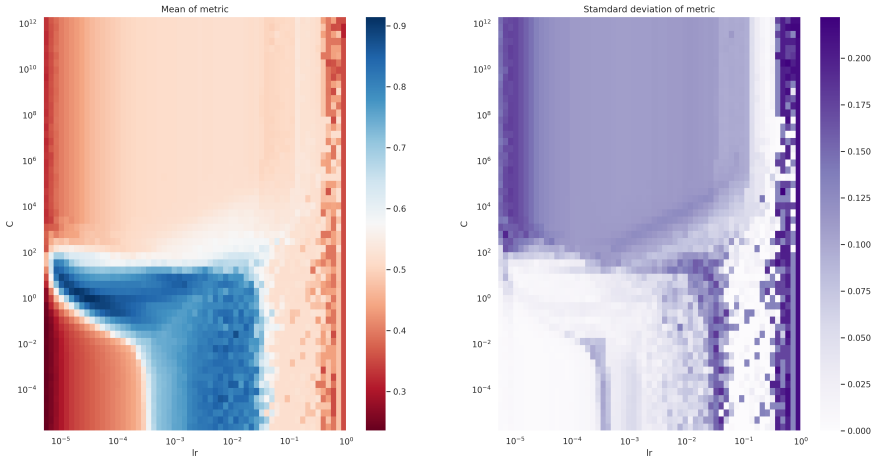


FIGURE B.4: Regret model's metric across C and learning rate on the circles dataset. From left to right: the mean value and the standard deviation. For each plot, the horizontal axis represents the learning rate and the vertical line represents the C parameter. Each point on this heatmap corresponds to the respective observed value for the combination (learning rate, C).

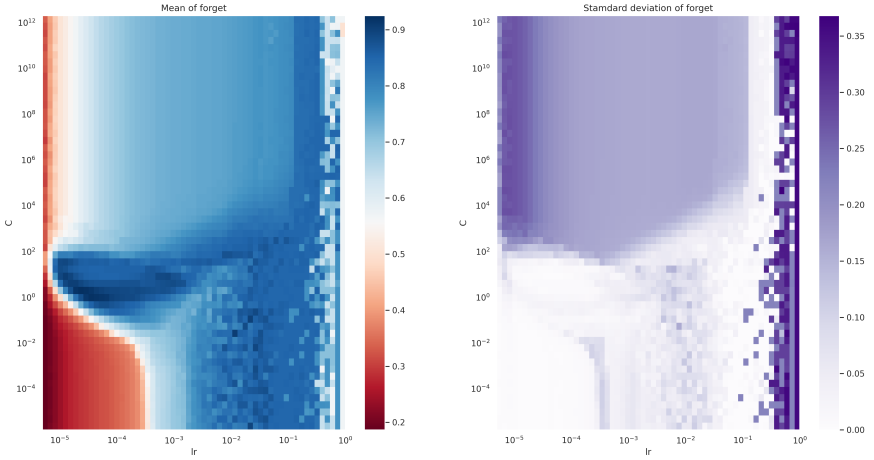


FIGURE B.5: Regret model's forget across C and learning rate on the circles dataset. From left to right: the mean value and the standard deviation. For each plot, the horizontal axis represents the learning rate and the vertical line represents the C parameter. Each point on this heatmap corresponds to the respective observed value for the combination (learning rate, C).

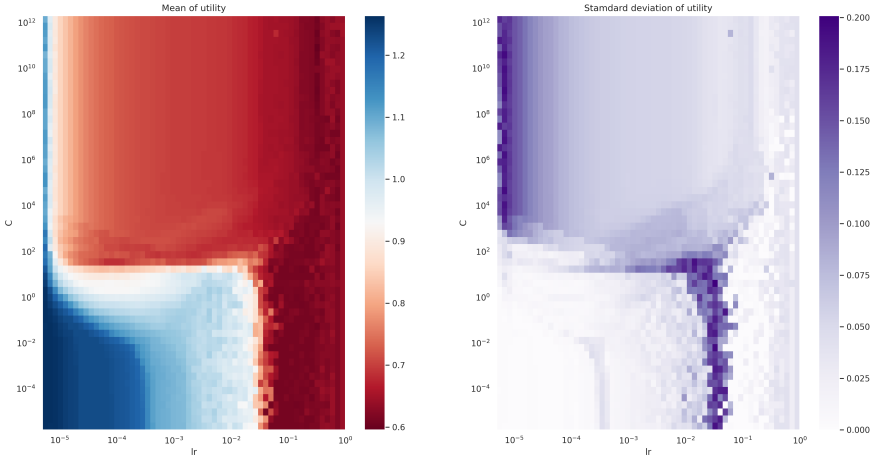


FIGURE B.6: Regret model's utility across C and learning rate on the circles dataset. From left to right: the mean value and the standard deviation. For each plot, the horizontal axis represents the learning rate and the vertical line represents the C parameter. Each point on this heatmap corresponds to the respective observed value for the combination (learning rate, C).

These plots show a similar pattern as the ones given in §B.1 despite it is not evident at first sight due to the different scale used by the colormap. For the metric, we observe again an assymptotic behaviour for C values bigger than 10^3. It also presents the same reverse relation between utility and forget and also a region comprised in the bottom which shows a good compromise between forget and utility and where the metric attains its best values.

lr	C	metric
1.0×10^{-5}	0	0.23 ± 0.00
	1.0×10^{-2}	0.23 ± 0.00
	1.0×10^{0}	0.24 ± 0.00
	5.0×10^{1}	0.39 ± 0.02
	1.0×10^{2}	0.40 ± 0.02
	5.0×10^{2}	0.36 ± 0.09
	1.0×10^{3}	0.38 ± 0.13
	1.0×10^{4}	0.38 ± 0.16
	1.0×10^{6}	0.37 ± 0.15
5.0×10^{-5}	0	0.33 ± 0.00
	1.0×10^{-2}	0.33 ± 0.00
	1.0×10^{0}	0.57 ± 0.01
	5.0×10^{1}	0.73 ± 0.03
	1.0×10^{2}	0.68 ± 0.05
	5.0×10^{2}	0.52 ± 0.17
	1.0×10^{3}	0.48 ± 0.19
	1.0×10^{4}	0.46 ± 0.18
	1.0×10^{6}	0.46 ± 0.18
1.0×10^{-4}	0	0.37 ± 0.00
	1.0×10^{-2}	0.37 ± 0.00
	1.0×10^{0}	0.88 ± 0.02
	5.0×10^{1}	0.63 ± 0.04
	1.0×10^{2}	0.61 ± 0.05
	5.0×10^{2}	0.49 ± 0.15
	1.0×10^{3}	0.48 ± 0.15
	1.0×10^{4}	0.47 ± 0.16
	1.0×10^{6}	0.47 ± 0.16
5.0×10^{-4}	0	0.46 ± 0.02
	1.0×10^{-2}	0.52 ± 0.08
	1.0×10^{0}	0.86 ± 0.03
	5.0×10^{1}	0.61 ± 0.06
	1.0×10^{2}	0.54 ± 0.09
	5.0×10^{2}	0.49 ± 0.13
	1.0×10^{3}	0.49 ± 0.13
	1.0×10^{4}	0.49 ± 0.13
	1.0×10^{6}	0.49 ± 0.13
1.0×10^{-3}	0	0.72 ± 0.04
	1.0×10^{-2}	0.73 ± 0.03
	1.0×10^{0}	0.82 ± 0.03
	5.0×10^{1}	0.61 ± 0.09
	1.0×10^{2}	0.55 ± 0.09
	5.0×10^{2}	0.50 ± 0.12
	1.0×10^{3}	0.49 ± 0.12
	1.0×10^{4}	0.49 ± 0.13
	1.0×10^{6}	0.49 ± 0.13

TABLE B.3: Mean and standard deviation of metric grouped by lr, C

lr	C	metric
5.0×10^{-3}	0	0.82 ± 0.05
	1.0×10^{-2}	0.82 ± 0.04
	1.0×10^{0}	0.77 ± 0.06
	5.0×10^{1}	0.63 ± 0.11
	1.0×10^{2}	0.55 ± 0.07
	5.0×10^{2}	0.52 ± 0.09
	1.0×10^{3}	0.52 ± 0.10
	1.0×10^{4}	0.50 ± 0.12
	1.0×10^{6}	0.49 ± 0.12
1.0×10^{-2}	0	0.83 ± 0.06
	1.0×10^{-2}	0.83 ± 0.08
	1.0×10^{0}	0.80 ± 0.07
	5.0×10^{1}	0.63 ± 0.13
	1.0×10^{2}	0.58 ± 0.09
	5.0×10^{2}	0.53 ± 0.07
	1.0×10^{3}	0.53 ± 0.09
	1.0×10^{4}	0.50 ± 0.11
	1.0×10^{6}	0.49 ± 0.12
5.0×10^{-2}	0	0.54 ± 0.08
	1.0×10^{-2}	0.53 ± 0.06
	1.0×10^{0}	0.53 ± 0.07
	5.0×10^{1}	0.52 ± 0.05
	1.0×10^{2}	0.52 ± 0.05
	5.0×10^{2}	0.53 ± 0.07
	1.0×10^{3}	0.53 ± 0.08
	1.0×10^{4}	0.51 ± 0.08
	1.0×10^{6}	0.50 ± 0.08
1.0×10^{-1}	0	0.51 ± 0.04
	1.0×10^{-2}	0.51 ± 0.07
	1.0×10^{0}	0.51 ± 0.04
	5.0×10^{1}	0.51 ± 0.04
	1.0×10^{2}	0.51 ± 0.04
	5.0×10^{2}	0.52 ± 0.04
	1.0×10^{3}	0.52 ± 0.06
	1.0×10^{4}	0.51 ± 0.08
	1.0×10^{6}	0.50 ± 0.07
5.0×10^{-1}	0	0.44 ± 0.16
	1.0×10^{-2}	0.44 ± 0.16
	1.0×10^{0}	0.44 ± 0.16
	5.0×10^{1}	0.44 ± 0.16
	1.0×10^{2}	0.44 ± 0.16
	5.0×10^{2}	0.44 ± 0.16
	1.0×10^{3}	0.44 ± 0.16
	1.0×10^{4}	0.44 ± 0.16
	1.0×10^{6}	0.44 ± 0.16

TABLE B.4: (continues) for the Regret model on the circles dataset.

On the light of the results shown in figure B.4 and table B.3 (first column, row corresponding to $lr = 1.0 \times 10^{-4}$), the best combination of learning rate and C seems 10^{-4} and 1, respectively. Since the optimal C parameter given in this grid search experiment is $C = 1$, we will keep this parameter and the learning rate as decided in the corresponding section. This decision regarding the learning rate is motivated by the reasoning exposed in §B.1.

B.3 On the spirals dataset

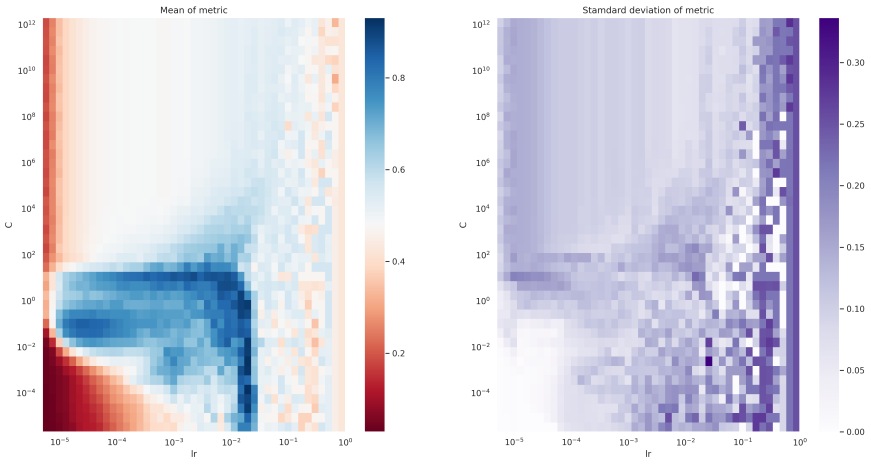


FIGURE B.7: Regret model's metric across C and learning rate on the spirals dataset. From left to right: the mean value and the standard deviation. For each plot, the horizontal axis represents the learning rate and the vertical line represents the C parameter. Each point on this heatmap corresponds to the respective observed value for the combination (learning rate, C).

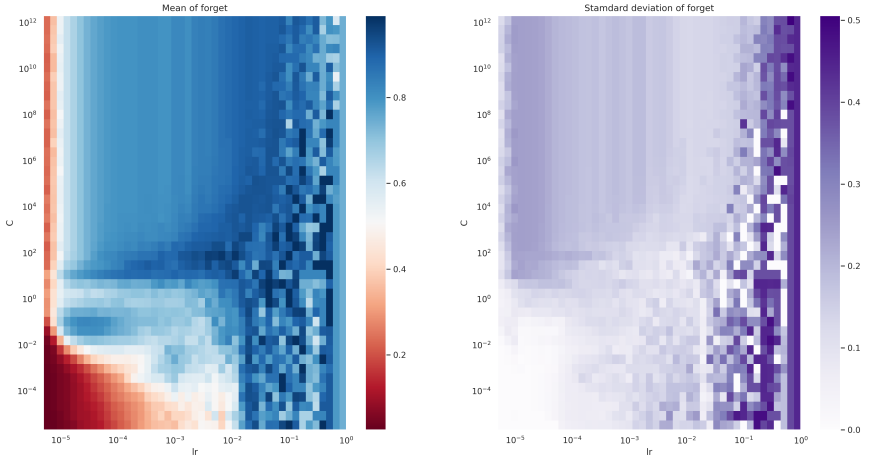


FIGURE B.8: Regret model's forget across C and learning rate on the spirals dataset. From left to right: the mean value and the standard deviation. For each plot, the horizontal axis represents the learning rate and the vertical line represents the C parameter. Each point on this heatmap corresponds to the respective observed value for the combination (learning rate, C).

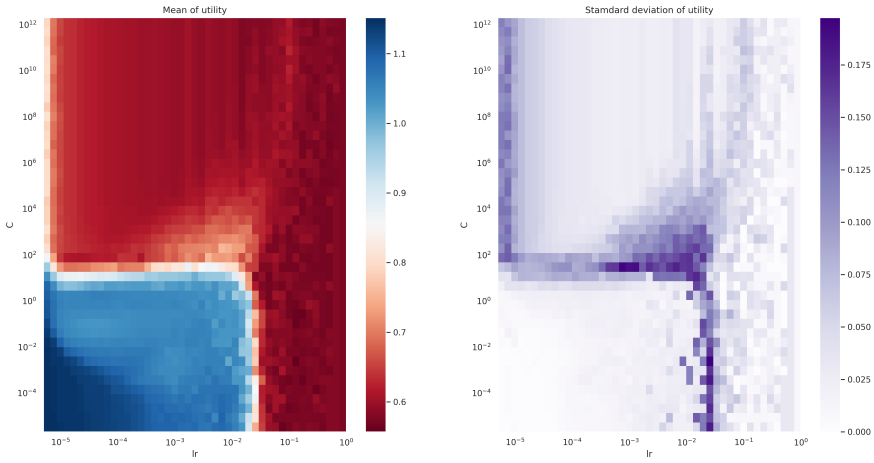


FIGURE B.9: Regret model's utility across C and learning rate on the spirals dataset. From left to right: the mean value and the standard deviation. For each plot, the horizontal axis represents the learning rate and the vertical line represents the C parameter. Each point on this heatmap corresponds to the respective observed value for the combination (learning rate, C).

A visual exploration on Figures B.7, B.8 and B.9 reveals the same patterns discussed in §B.1 and B.2.

lr	C	metric
1.0×10^{-5}	0	0.03 ± 0.00
	1.0×10^{-2}	0.04 ± 0.00
	1.0×10^{0}	0.27 ± 0.03
	5.0×10^{1}	0.20 ± 0.09
	1.0×10^{2}	0.19 ± 0.09
	5.0×10^{2}	0.20 ± 0.10
	1.0×10^{3}	0.20 ± 0.10
	1.0×10^{4}	0.20 ± 0.10
	1.0×10^{6}	0.20 ± 0.10
5.0×10^{-5}	0	0.04 ± 0.00
	1.0×10^{-2}	0.51 ± 0.01
	1.0×10^{0}	0.65 ± 0.12
	5.0×10^{1}	0.49 ± 0.14
	1.0×10^{2}	0.48 ± 0.14
	5.0×10^{2}	0.49 ± 0.13
	1.0×10^{3}	0.48 ± 0.13
	1.0×10^{4}	0.48 ± 0.13
	1.0×10^{6}	0.48 ± 0.13
1.0×10^{-4}	0	0.07 ± 0.00
	1.0×10^{-2}	0.65 ± 0.02
	1.0×10^{0}	0.66 ± 0.14
	5.0×10^{1}	0.52 ± 0.12
	1.0×10^{2}	0.51 ± 0.12
	5.0×10^{2}	0.50 ± 0.10
	1.0×10^{3}	0.50 ± 0.11
	1.0×10^{4}	0.50 ± 0.11
	1.0×10^{6}	0.50 ± 0.11
5.0×10^{-4}	0	0.21 ± 0.01
	1.0×10^{-2}	0.65 ± 0.07
	1.0×10^{0}	0.67 ± 0.11
	5.0×10^{1}	0.59 ± 0.10
	1.0×10^{2}	0.57 ± 0.07
	5.0×10^{2}	0.54 ± 0.08
	1.0×10^{3}	0.53 ± 0.08
	1.0×10^{4}	0.52 ± 0.08
	1.0×10^{6}	0.52 ± 0.08
1.0×10^{-3}	0	0.28 ± 0.03
	1.0×10^{-2}	0.66 ± 0.11
	1.0×10^{0}	0.67 ± 0.11
	5.0×10^{1}	0.60 ± 0.10
	1.0×10^{2}	0.57 ± 0.11
	5.0×10^{2}	0.54 ± 0.08
	1.0×10^{3}	0.53 ± 0.08
	1.0×10^{4}	0.52 ± 0.08
	1.0×10^{6}	0.52 ± 0.08

Table B.5: Mean and standard deviation of metric grouped by lr, C

lr	C	metric
5.0×10^{-3}	0	0.38 ± 0.20
	1.0×10^{-2}	0.67 ± 0.10
	1.0×10^{0}	0.71 ± 0.13
	5.0×10^{1}	0.75 ± 0.18
	1.0×10^{2}	0.67 ± 0.14
	5.0×10^{2}	0.59 ± 0.06
	1.0×10^{3}	0.56 ± 0.07
	1.0×10^{4}	0.54 ± 0.06
	1.0×10^{6}	0.53 ± 0.08
1.0×10^{-2}	0	0.49 ± 0.14
	1.0×10^{-2}	0.76 ± 0.16
	1.0×10^{0}	0.80 ± 0.11
	5.0×10^{1}	0.68 ± 0.16
	1.0×10^{2}	0.65 ± 0.15
	5.0×10^{2}	0.57 ± 0.08
	1.0×10^{3}	0.58 ± 0.06
	1.0×10^{4}	0.55 ± 0.08
	1.0×10^{6}	0.53 ± 0.08
5.0×10^{-2}	0	0.45 ± 0.16
	1.0×10^{-2}	0.42 ± 0.20
	1.0×10^{0}	0.54 ± 0.06
	5.0×10^{1}	0.54 ± 0.07
	1.0×10^{2}	0.56 ± 0.00
	5.0×10^{2}	0.55 ± 0.04
	1.0×10^{3}	0.54 ± 0.05
	1.0×10^{4}	0.53 ± 0.12
	1.0×10^{6}	0.57 ± 0.01
1.0×10^{-1}	0	0.56 ± 0.00
	1.0×10^{-2}	0.52 ± 0.14
	1.0×10^{0}	0.52 ± 0.15
	5.0×10^{1}	0.56 ± 0.00
	1.0×10^{2}	0.56 ± 0.00
	5.0×10^{2}	0.55 ± 0.05
	1.0×10^{3}	0.56 ± 0.00
	1.0×10^{4}	0.54 ± 0.08
	1.0×10^{6}	0.57 ± 0.01
5.0×10^{-1}	0	0.50 ± 0.16
	1.0×10^{-2}	0.50 ± 0.16
	1.0×10^{0}	0.50 ± 0.16
	5.0×10^{1}	0.50 ± 0.16
	1.0×10^{2}	0.50 ± 0.16
	5.0×10^{2}	0.50 ± 0.16
	1.0×10^{3}	0.50 ± 0.16
	1.0×10^{4}	0.50 ± 0.16
	1.0×10^{6}	0.50 ± 0.16

Table B.6: (continues) for the Regret model on the spirals dataset.

In Figure B.7 and Table B.5 (second column, row corresponding to $lr = 1.0 \times 10^{-2}$) we observe that the best combination of learning rate and C appears to be 10^{-2} and 1, respectively. Since the optimal C parameter given in this grid search experiment is $C = 1$, we will keep this parameter and the learning rate as decided in the corresponding section, following the reasoning expressed in §B.1.

B.4 On the MNIST dataset

We compute in this section the dependency on both the learning rate and the C parameter for the Regret model on the MNIST dataset.

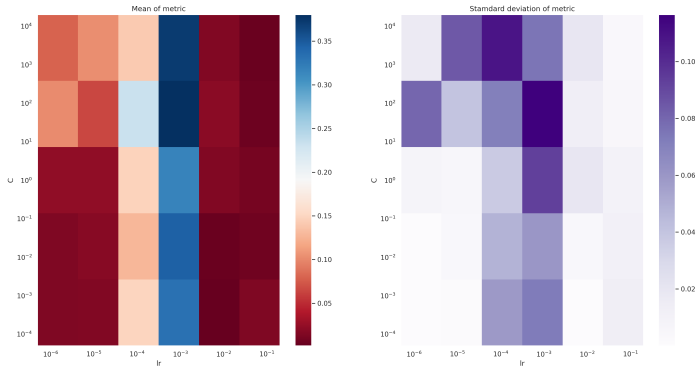


FIGURE B.10: Regret model's metric across C and learning rate on the MNIST dataset. From left to right: the mean value and the standard deviation. For each plot, the horizontal axis represents the learning rate and the vertical line represents the C parameter. Each point on this heatmap corresponds to the respective observed value for the combination (learning rate, C).

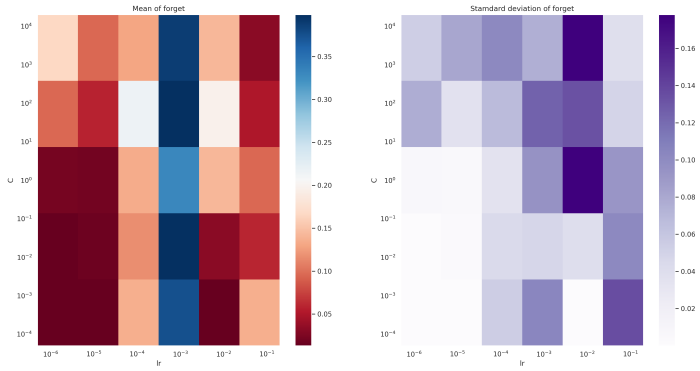


FIGURE B.11: Regret model's forget across C and learning rate on the MNIST dataset. From left to right: the mean value and the standard deviation. For each plot, the horizontal axis represents the learning rate and the vertical line represents the C parameter. Each point on this heatmap corresponds to the respective observed value for the combination (learning rate, C).

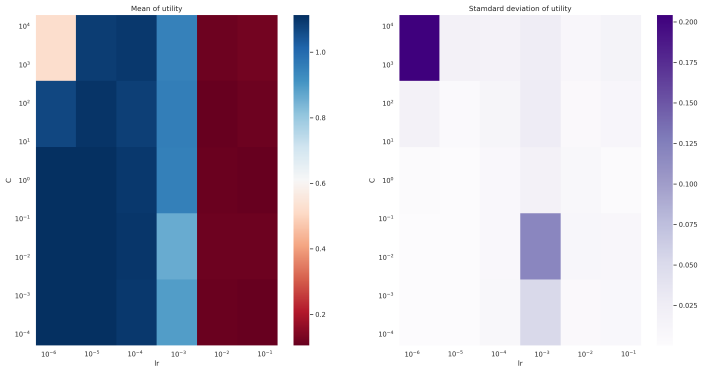


FIGURE B.12: Regret model's utility across C and learning rate on the MNIST dataset. From left to right: the mean value and the standard deviation. For each plot, the horizontal axis represents the learning rate and the vertical line represents the C parameter. Each point on this heatmap corresponds to the respective observed value for the combination (learning rate, C).

Possibly due to the substantially lower frequency at which we sample both the learning rate and the C parameter due to the computational expense of training the MNIST models, we do not observe the previous patterns very clearly. What we can see is that, for the studied values, the learning rate here has a critical impact on the metrics, much more than the C parameter. Low learning rates yield better utilities than high learning rates. A learning rate of 10^{-3} results in the best compromise between utility and forget.

lr	C	metric
1.0×10^{-6}	1.0×10^{-4}	0.01 ± 0.00
	1.0×10^{-2}	0.01 ± 0.00
	1.0×10^{0}	0.02 ± 0.01
	1.0×10^{2}	0.10 ± 0.08
	1.0×10^{4}	0.08 ± 0.02
1.0×10^{-5}	1.0×10^{-4}	0.02 ± 0.00
	1.0×10^{-2}	0.02 ± 0.00
	1.0×10^{0}	0.02 ± 0.01
	1.0×10^{2}	0.06 ± 0.04
	1.0×10^{4}	0.10 ± 0.08
1.0×10^{-4}	1.0×10^{-4}	0.15 ± 0.06
	1.0×10^{-2}	0.13 ± 0.05
	1.0×10^{0}	0.15 ± 0.04
	1.0×10^{2}	0.23 ± 0.07
	1.0×10^{4}	0.14 ± 0.11

TABLE B.7: Mean and standard deviation of metric grouped by lr, C

lr	C	metric
1.0×10^{-3}	1.0×10^{-4}	0.33 ± 0.07
	1.0×10^{-2}	0.34 ± 0.06
	1.0×10^{0}	0.32 ± 0.09
	1.0×10^{2}	0.38 ± 0.12
	1.0×10^{4}	0.37 ± 0.08
1.0×10^{-2}	1.0×10^{-4}	0.00 ± 0.00
	1.0×10^{-2}	0.00 ± 0.00
	1.0×10^{0}	0.02 ± 0.02
	1.0×10^{2}	0.02 ± 0.01
	1.0×10^{4}	0.02 ± 0.02
1.0×10^{-1}	1.0×10^{-4}	0.01 ± 0.01
	1.0×10^{-2}	0.01 ± 0.01
	1.0×10^{0}	0.01 ± 0.01
	1.0×10^{2}	0.01 ± 0.01
	1.0×10^{4}	0.00 ± 0.00

TABLE B.8: (continues) for the Regret model on the MNIST dataset.

This experiment reveals that all learning rates different than 10^{-3} perform worse for any C. Therefore, the dependence of C reduces to studying it with the learning

rate fixed to 10^{-3}. The consequent results are shown in figure B.13 and table B.7.

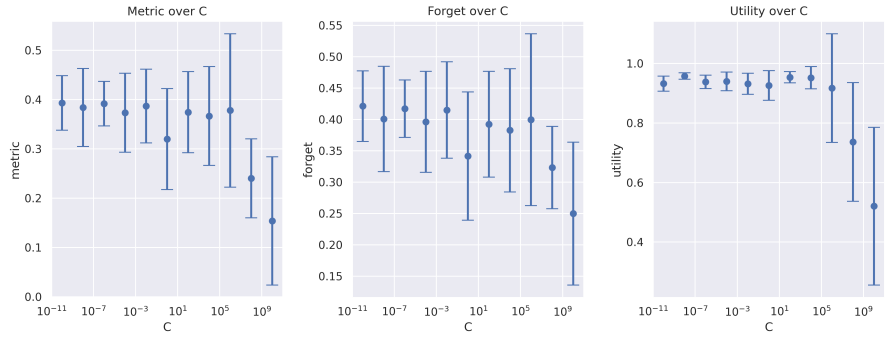


FIGURE B.13: Regret model's performance across C for a learning rate of 10^{-3} on the MNIST dataset. From left to right, the values corresponding to metric, forget and utility. Horizontal axis represents the learning rate being used. The vertical axis represents the respective value. For each considered C value, each metrics value is represented by a dot on the observed average value together with confidence bars representing one standard deviation above and below.

C	**metric**	**forget**	**utility**
1.0×10^{-10}	0.39 ± 0.06	0.42 ± 0.06	0.93 ± 0.03
1.0×10^{-8}	0.38 ± 0.08	0.40 ± 0.08	0.96 ± 0.01
1.0×10^{-6}	0.39 ± 0.05	0.42 ± 0.05	0.94 ± 0.02
1.0×10^{-4}	0.37 ± 0.08	0.40 ± 0.08	0.94 ± 0.03
1.0×10^{-2}	0.39 ± 0.07	0.42 ± 0.08	0.93 ± 0.04
1.0×10^{0}	0.32 ± 0.10	0.34 ± 0.10	0.93 ± 0.05
1.0×10^{2}	0.37 ± 0.08	0.39 ± 0.08	0.95 ± 0.02
1.0×10^{4}	0.37 ± 0.10	0.38 ± 0.10	0.95 ± 0.04
1.0×10^{6}	0.38 ± 0.16	0.40 ± 0.14	0.92 ± 0.18
1.0×10^{8}	0.24 ± 0.08	0.32 ± 0.07	0.74 ± 0.20
1.0×10^{10}	0.15 ± 0.13	0.25 ± 0.11	0.52 ± 0.26

TABLE B.9: Performance metrics by C for the Regret model and MNIST dataset. For each metric, the values are given by their observed mean $\pm$ their observed standard deviation.

From this experiment we can extract that the performance worsens for extremely big learning rates. However, the same is not observed for extremely small learning rates. Based on the previous results for the toy datasets and the fact that for $C = 0$ we recover the Fine-tuning model, which had a lower average metric, we conclude that this is likely due to numerical instability. In terms of optimizing the model, there does not seem to be any clear optimal C value for this discretization, so we keep the original $C = 1$ value.

Appendix C

Source code repository URL

All the code used to generate this thesis can be found at the GitHub repository with the following URL: https://github.com/arnauJutglar/MFPDS_TFM_UNLEARNING.

Bibliography

[1] Lucas Bourtoule et al. “Machine Unlearning”. In: *arXiv preprint arXiv:1912.03817* (2020).

[2] Tom B. Brown et al. “Language Models are Few-Shot Learners”. In: *arXiv preprint arXiv:2005.14165* (2020).

[3] Katharina Buchholz. *ChatGPT training cost estimation*. https://www.forbes.com/sites/katharinabuchholz/2024/08/23/the-extreme-cost-of-training-ai-models/. Accessed: 2025-02-13. 2024.

[4] *Fanchuan implementation for Google's Unlearning Challenge*. https://www.kaggle.com/code/fanchuan/2nd-place-machine-unlearning-solution. Accessed: 2025-02-10.

[5] Antonio A. Ginart et al. “Making AI Forget You: Data Deletion in Machine Learning”. In: *arXiv preprint arXiv:1907.05012* (2019).

[6] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. “Eternal Sunshine of the Spotless Net: Selective Forgetting in Deep Networks”. In: *arXiv preprint arXiv:1911.04933* (2020).

[7] Thanh Tam Nguyen et al. “A Survey of Machine Unlearning”. In: *arXiv preprint arXiv:2209.02299* (2024).

[8] Subhodip panda2023 and Prathosh AP. “FAST: Feature Aware Similarity Thresholding for Weak Unlearning in Black-Box Generative Models”. In: *arXiv preprint arXiv:2312.14895v1* (2023).

[9] Anvith Thudi et al. “Unrolling SGD: Understanding Factors Influencing Machine Unlearning”. In: *arXiv preprint arXiv:2109.13398* (2022).

[10] Eleni Triantafillou et al. “Are we making progress in unlearning? Findings from the first NeurIPS unlearning competition”. In: *arXiv preprint arXiv:2406.09073* (2024).

[11] Pablo Noriega Vázquez. *Machine unlearning: el arte de olvidar en la era de la Inteligencia Artificial*. https://hdl.handle.net/2445/216709. Accessed: 2025-01-16.

[12] Weiqi Wang et al. “Machine Unlearning: A Comprehensive Survey”. In: *arXiv preprint arXiv:2405.07406* (2024).

[13] Yinjun Wu, Edgar Dobriban, and Susan B. Davidson. “DeltaGrad: Rapid retraining of machine learning models”. In: *arXiv preprint arXiv:2006.14755* (2020).