Per-Sample Amnesiac Unlearning

July 15, 2024

Davide Marincione

Abstract

Typical machine unlearning techniques do not scale well with modern deep learning models, as these either require models to abide to strict theoretical guarantees (lowering their effectiveness) or be retrained (unfeasible for big models). In this project we propose a variation of Amnesiac Unlearning, a technique which promises to bring unlearning with little to no compromises to neural networks. Our method lowers the memory requirements of the original technique while maintaining its effectiveness. We test it on MNIST and CIFAR-100, showing that it can make a model forget samples and classes while maintaining its performance on the rest of the dataset.

1. Introduction

Machine Unlearning is an ever-more important topic in the field of Machine Learning. Not only because privacy concerns regarding deep learning techniques are increasing, but also because regulations such as the EU's GDPR oblige companies to delete user's data at their request. Because of this, research in trying to produce an unlearning procedure that requires the less possible amount of retraining is increasing.

Our work consists of:

- A new unlearning technique with lower memory requirements than the original Amnesiac Unlearning technique (**?**).
- Tests on MNIST and CIFAR-100 showing that our technique can make a model forget samples and classes while maintaining its performance on the rest of the dataset.

Email: Davide Marincione <marincione.1927757@studenti.uniroma1.it>.

Deep Learning and Applied AI 2024, Sapienza University of Rome, 2nd semester a.y. 2023/2024.

2. Related Work

As far as single-technique driven methods go (as combined procedures are often used in practice), the *model shifting* (**?**) family of techniques is one of the most promising. These techniques are based on the idea of directly updating the model parameters to offset the impact of the samples to forget; by finding an update δ for $w_u = w + \delta$, where w are the parameters of the original model. The reason for why this solution is so promising is that it does not require a major retraining of the model, which is often infeasible for big models: both in terms of time and computational resources.

One of the most successful *model shifting* techniques is Amnesiac Unlearning (**?**), which defines the update δ with the gradient updates in which the samples to forget took part. To do so, the gradient update of each batch must be stored on disk, leading to a high memory requirement, as each update is of the same size as the model. This is a major drawback, as it makes the technique unfeasible for big models. An extremely recent work (**?**) proposes an approximation which only stores a random subset of the gradients in the update, therefore reducing the memory requirements. However, this leads to a decrease in the effectiveness of the technique, which requires a short retraining phase to completely forget the samples.

3. Method

Our method consists of storing the gradients not for each batch, but for each sample. On the long run, this leads to a lower memory requirement than the original technique, as we only need to store a single set of gradients for each sample (rather than a new set for each batch). If the space requirements of the original technique are $O(\frac{NME}{B})$, our technique requires $O(NMp)$ space, where N is the number of samples in the dataset, E is the number of epochs, B is the batch size, M is the number of parameters in the model, and p is percentage of parameters kept in the random subset. It is easy to see that our method requires less space than the original technique if $p < \frac{E}{B}$ (which can be easily achieved in practice, as if we heuristically set $p = .1$ and have a typical $B = 64$ then we just require $E = 7$).

Not only can our method require less space than the original, but it also has the potential to be more effective than the approximation proposed in (**?**), as it only removes the gradients of the samples to forget, rather than the gradients of the full batch (where the influence of samples that are not to forget is removed as well).

4. Experiments

We modify the code provided by the authors of (**?**) to test our method as well as the original Amnesiac Unlearning technique, and the approximation proposed in (**?**) (as they have yet to publish their own code).

Our implementation slightly differs from the original: in it, they store the difference between the model before and after the gradient update, making it invariant to the choice of optimizer. Our case instead, since `pytorch` allows the calculation of per-sample gradients through `call_for_per_sample_grads` but not the usage of its gradients through an optimizer, we had to directly store the gradients, and not the difference between the model before and after the update. This implies that, since optimization usually doesn't remove the full magnitude of a gradient during a step and in some cases doesn't actually remove the gradient itself (but a moving average of it), our implementation is not invariant to the choice of optimizer. Still, as in the original technique we trained the model with `Adam`, and nonetheless we obtained good results, therefore we believe that this is not a major drawback.