Continual Learning + Machine Unlearning

Pengxiang Wang

Peking University, School of Mathematical Sciences

University of Bristol, School of Engineering Mathematics and Technology

2024-10-28

Machine Unlearning

Machine Unlearning Motivation

What is **machine unlearning**:

> ***Machine unlearning*** *is the process of deliberately removing specific training data from a machine learning model to ensure that the removed data no longer influences the model's predictions. It offers undo option of machine learning process.*

Data Deletion:

- Traditionally: delete from databases
- AI: delete both from back-end databases and from trained models

Application Movitation:

- **Privacy**:
  - Regulations: GDPR, CCPA, etc. when the user withdraw the consent, "the right to be forgotten"
  - Delete the requested data by users
- **Security**:
  - Adversarial attacks are possible to extract private information from the trained model. E.g., model inversion attacks, membership inference attacks, etc.

Machine Unlearning Framework

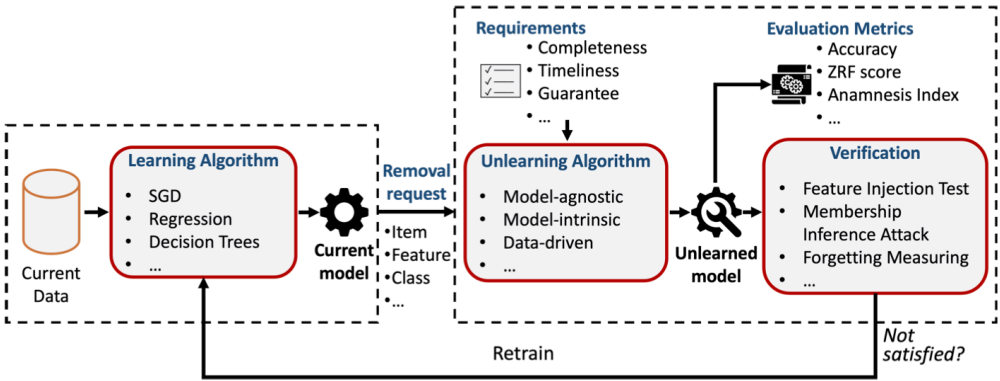


Figure 1: Machine Unlearning Framework

Formal Definition

Notations:

- Training data D, training algorithm A, model trained $A(D)$
- Unlearning data $D_f \subset D$, unlearning algorithm U
- Unlearned model $U(D, D_f, A(D))$

Objective:

- Unlearned model $U(D, D_f, A(D))$ is expected to be the same or similar to a retrained model $A(D\ D_f)$
- The similarity is measured by indistinguishability metrics
- Distance bewteen model parameters (l_2 distance), distributions (K-L divergence), etc.
-

Assumptions:

- The unlearning data are small compared to the training data
  - True in real-world applications
  - Otherwise, retrain can solve everything

Retraining

The problem makes unlearning difficult:

- Neural networks parameters do not tend to show any clear connection to the training data. AI models have to be considered as a whole.
- Stochasticity and Incrementality of training
- Catastrophic Unlearning: the holistic character of neural networks can easily lead to excessive unlearning too much then reduce performance

Retraining:

- Delete target data and re-train the model with the rest of data from scratch
- A naive way, but not always feasible
- The only exact unlearning method, and achieves upper bound

The problem of retraining:

- Doesn't worth, computation cost
- Not always having aceess to all training data

Methodology

The forgotten set could be:

- Item removal: data points
- Class removal
- Feature removal

The unlearning process could be:

- Model-agnostic or model-intrinsic
- Data-driven approaches (most model-agnostic)

Method: SISA

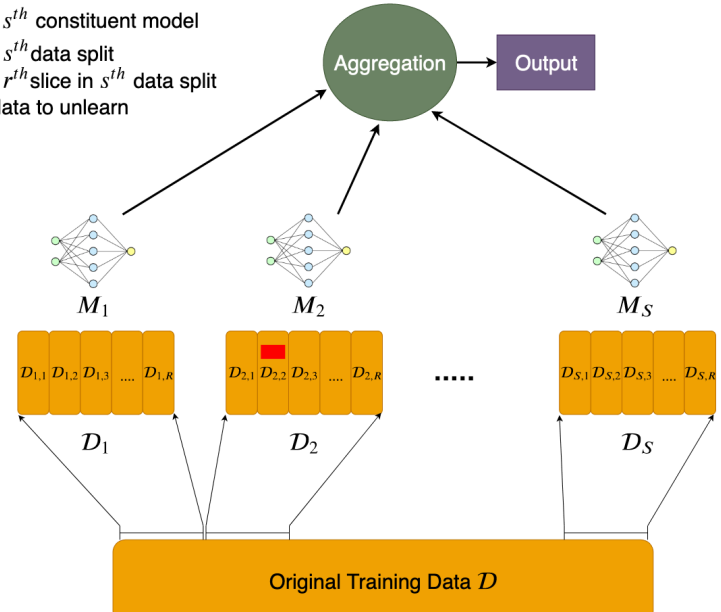
SISA (Sharded, Isolated, Sliced, Aggregated), 2021:

- Item removal
- Isolate network into constituent networks, divide data into shards
- Build up correspondance bewteen divided network and data, trained correspondingly
- Unlearning: retraining the corresponding network of the data shard to be forgotten

Core idea: fractionizing the retraining process into smaller units, reduce the cost of full retraining

Method: SISA

- M_s : s^{th} constituent model
- $\mathcal{D}_s$: s^{th} data split
- $\mathcal{D}_{s,r}$: r^{th} slice in s^{th} data split
- : data to unlearn



SISA: Technical Details

Q: How the divided networks predict?

A: Label-based majority vote: each constiuent network predicts the label, and the final label is determined by majority vote

Q: How model is trained from the original data?

A: A data batch is assigned to a shard, and the corresponding network is trained on the shard just like normal training. The constituent networks are isolated from each other, and the training process is independent.

Q: How to unlearn?

A: Delete the forgotten data from the corresponding shard, and retrain the

Method: AmnesiacML

Amnesiac Machine Learning, 2021:

- Item removal
- Keep a record update on parameters (i.e. the step term `-lr \cdot g` in gradient descent, the difference of parameters before and after updating) during training each batch of data
- Unlearning: simply substract the parameter update from the corresponding batch

The problem: - The stochasticity and incrementality of the training process - Storing the parameter updates is as expensive as storing the model itself

Some assumptions:

- The data holder is only concerned about possible potential removal of a subset of data
- Only need to keep the parameter updates from batches containing that data

Method: Error-Max Anti-sample generation

Error-Max Anti-sample generation - Class Removal - Max instead of Min the loss - Do some repair

2.

Impair

Repair