Rensselaer

why not change the world?®

SCRUB the Shard model unlearning approach

presented by Ronnakorn Rattanakornphan | 04/19/2024

In this work, we present

- Model unlearning approach, combining
  - SISA: model unlearning by retrain a model slice
  - SCRUB: model unlearning by teacher-student learning

Background

What does model unlearning even mean?

Bob previously allows his data to be included in AIDS prediction model, but now he doesn't want his data to be included anymore.

We want to not only remove Bob from training set,

but also any influence of Bob in the model

- Remove influence => model unlearning

Model unlearning (formally)

Output on unlearned model trained on D_{all} the same as

Output on model trained on D_r

$$Pr(A(D \setminus D_f)) = Pr(U(D, D_f, A(D)))$$

- given dataset
  - D_{all} = $D_u \cup D_f$
  - D_{all} = all data
  - D_f = (to be) forget data
  - D_r = retained data
  - A = learning algorithm

Challenge in unlearning

1. Hard to understand impact of a data point
   a. Especially worse in NN
2. Stochastic training
   a. Training is non-deterministic
3. Incremental training
   a. An update on a training point affect later update following that training point

Model unlearning method (naive)

- Just remove the unlearn data D_f and retrain the whole model
- Golden standard (this is exactly what we want)
- Disadvantage
  - Slow to retrain the model from scratch
  - Might not have initial data, D_r
    - Common in federated learning

Model unlearning approaches

1. Data reorganization
    a. self-poison
2. Data pruning
    a. divide dataset into multiple pieces, then aggregate them later
3. Data replacement
    a. transform training data

Model unlearning approaches

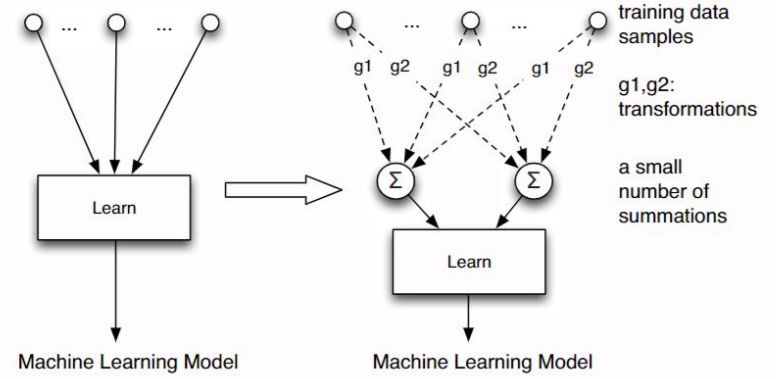
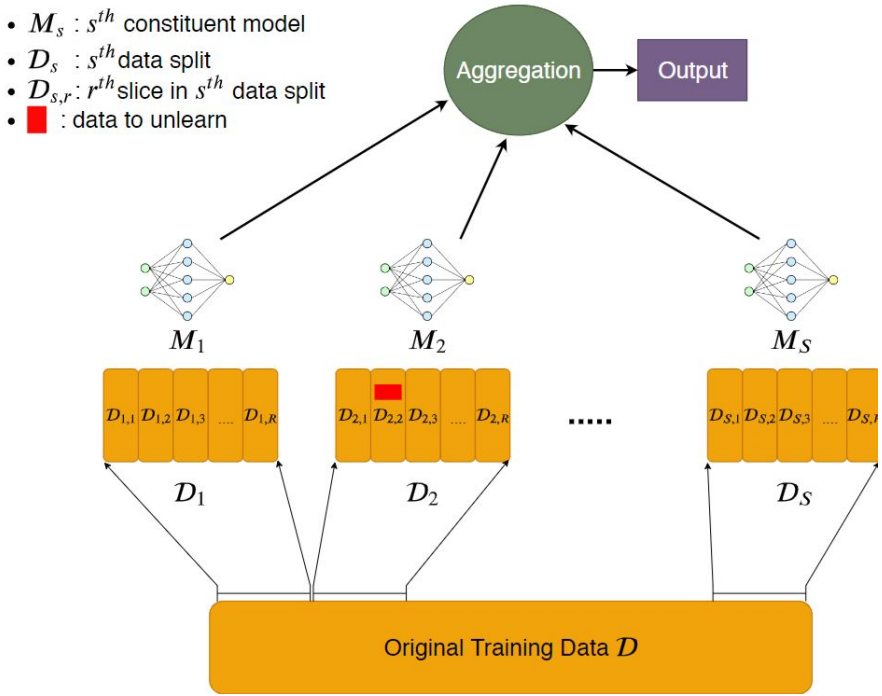


Fig. 1: Unlearning idea. Instead of making a model directly depend on each training data sample (left), we convert the learning algorithm into a summation form (right). Specifically, each summation is the sum of transformed data samples, where the transformation functions g_i are efficiently computable. There are only a small number of summations, and the learning algorithm depends only on summations. To forget a data sample, we simply update the summations and then compute the updated model. This approach is asymptotically much faster than retraining from scratch.

Model unlearning approaches

4. Model shifting
    a. directly update the model parameter by offsetting the computed impact of data point(s)
5. Model replacement
    a. directly replaces some parameters with pre-calculated parameters
6. Model pruning
    a. prunes some parameters from the trained models
    b. require fine-tuning process to recover performance

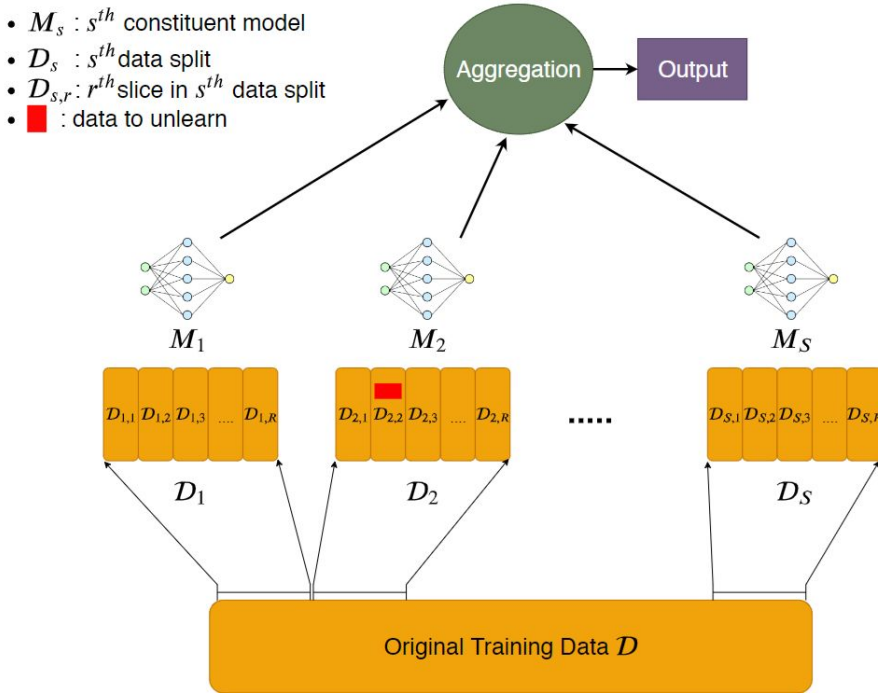
Reference work: *SISA



Instead of naively retrain the whole model, what if we just retrain part of it?

*published: 2021 IEEE Symposium on Security and Privacy (SP)

Reference work: SISA



1. Divide dataset into shards
2. Divide each shards into slice
3. Train one model for each shard
4. Train slice one by one, save checkpoints
5. At inference, aggregate the output

When unlearn, retrain only the shard from checkpoint onwards

Reference work: SISA

Pros	Cons
Guarantee exact unlearning	Still slow
Less unlearn time, compared to naive unlearning	Would not help if too many unlearn request that spread across all shard. - Would be the same as naive unlearning
Model agnostic (can be use in any model)	Each model is now “Weak learner”
Simple to implement	Require more storage (many model to store)
Does not require all initial training sample. Can work in federated learning setting, if the unlearning happen on the owned shard.	Require more dataset (to maintain good performance)

Reference work: *SCRUB

Teacher-student framework for model unlearning

Recall

D_f = forget data, D_r = retained data

Goal:

Forget D_f, while remember D_r, that is

Listen to the teacher when talking about D_r

Disobey teacher when talking about D_f

*published: 37th Conference on Neural Information Processing Systems (NeurIPS 2023)

Reference work: SCRUB

Objective function (building block):

$$d(x; w^u) = D_{\mathrm{KL}}(p(f(x; w^o)) \| p(f(x; w^u)))$$

$d(x; w^u)$ is KL-divergence between the student and teacher output distributions (softmax probabilities) for the example x

w^o weight of original model, the teacher

w^u weight of unlearned model, the student, this is the same as w^o at the beginning

$f(\cdot; w)$ function parameterized by trainable weights w, the “model”

Reference work: SCRUB

Objective function:

$$\min_{w^u} \frac{\alpha}{N_r} \sum_{x_r \in \mathcal{D}_r} d(x_r; w^u) + \frac{\gamma}{N_r} \sum_{(x_r, y_r) \in \mathcal{D}_r} l(f(x_r; w^u), y_r) - \frac{1}{N_f} \sum_{x_f \in \mathcal{D}_f} d(x_f; w^u)$$

Retained data

Minimize divergence with teacher

Retained data

Minimize task loss

Forget data

Maximize divergence with teacher

We want to forget $x_f \in D_f$, while remember $x_r \in D_r$

Reference work: SCRUB

Objective function:

$$\min_{w^u} \frac{\alpha}{N_r} \sum_{x_r \in \mathcal{D}_r} d(x_r; w^u) + \frac{\gamma}{N_r} \sum_{(x_r, y_r) \in \mathcal{D}_r} l(f(x_r; w^u), y_r) - \frac{1}{N_f} \sum_{x_f \in \mathcal{D}_f} d(x_f; w^u)$$

Retained data
Minimize divergence with teacher

Retained data
Minimize task loss

Forget data
Maximize divergence with teacher

For forget data, x_f, We want the student to stay away from teacher.

We do this by maximize the divergence between student and teacher output, given x_f

Reference work: SCRUB

Objective function:

$$\min_{w^u} \frac{\alpha}{N_r} \sum_{x_r \in \mathcal{D}_r} d(x_r; w^u) + \frac{\gamma}{N_r} \sum_{(x_r, y_r) \in \mathcal{D}_r} l(f(x_r; w^u), y_r) - \frac{1}{N_f} \sum_{x_f \in \mathcal{D}_f} d(x_f; w^u)$$

Retained data
Minimize divergence with teacher

Retained data
Minimize task loss

Forget data
Maximize divergence with teacher

However, training only previous objective $\min_{w^u} -\frac{1}{N_f} \sum_{x_f \in \mathcal{D}_f} d(x_f; w^u)$ will degrade the performance on retained data D_r , so we want to student to “stay close” to teacher when we see a retained sample $x_r \in D_r$, by minimize the divergence between student and teacher output

Reference work: SCRUB

Objective function:

$$\min_{w^u} \frac{\alpha}{N_r} \sum_{x_r \in \mathcal{D}_r} d(x_r; w^u) + \frac{\gamma}{N_r} \sum_{(x_r, y_r) \in \mathcal{D}_r} l(f(x_r; w^u), y_r) - \frac{1}{N_f} \sum_{x_f \in \mathcal{D}_f} d(x_f; w^u)$$

Retained data

Minimize divergence with teacher

Retained data

Minimize task loss

Forget data

Maximize divergence with teacher

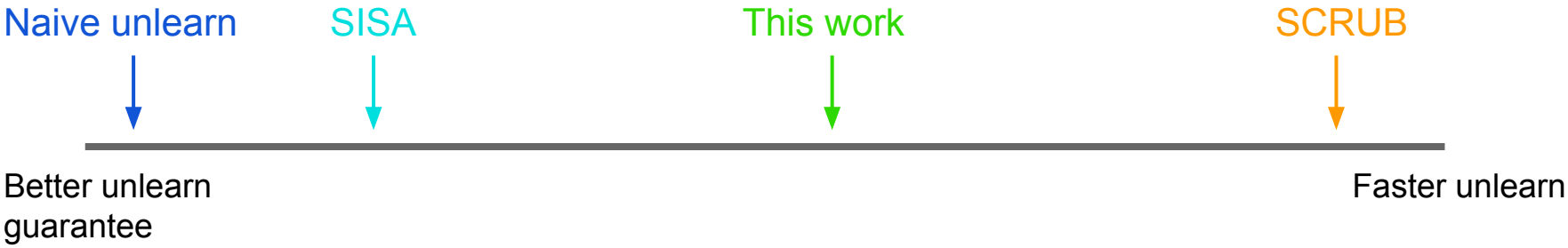
Finally, SCRUB want the student model to be good at the task, so it simultaneously optimize the model on the task loss

Reference work: SCRUB

Pros	Cons
Less unlearn time, compared to naive unlearning	Does not guarantee exact unlearning
Model agnostic (can be use in any model)	Require modification to training procedure
Faster unlearn process than SISA (require less epoch)	Require all initial training sample. Will not work in federated learning setting.
Scale to many unlearn request (does not revert to naive unlearning when there're many unlearn request like SISA)	

Method

Combining SISA and SCRUB



This work aims to improve upon SISA by using faster unlearn process of SCRUB while maintaining good unlearn guarantee under some scenario

Combining SISA and SCRUB

In simple terms, we unlearn a shard in SISA by using SCRUB instead of naive unlearn

Compared to SISA, we have faster unlearn process that does not get reverted back to naive unlearning when there’re many unlearning request.

Combining SISA and SCRUB

Compared to SCRUB, we can take advantage of **different unlearn likelihood** across different sample.

For example, the initial model might be trained on both public and private data. While private data is likely to get unlearned request, publicly available data might never receive unlearn request.

SCRUB by itself cannot take advantage of this information, however, in our case, we can separate the shard that has high and low likelihood of unlearn request and take advantage of that information.

Combining SISA and SCRUB

For example, we may concentrate the unlearn request into particular shard(s) while keeping portion of model ensample (which is learn on stable/no unlearn request shard) the same.

We also have the option of separating the data into shard according to **different privacy requirement**.

For example, we might want to keep a highly sensitive dataset under one model shard and use normal retraining instead of SCRUB unlearning for a better privacy guarantee.

Combining SISA and SCRUB (algo.)

1. Train the base model using SISA
    a. divide data into shards
    b. train independent model for each shard
        i. normal training
    c. This will be the base model
2. Handle unlearn request
    a. For each affected data shards, separate it into forget set, retain set
    b. Apply unlearn method on model shard, given forget set and retain set

SCRUB

Objective function:

$$\min_{w^u} \frac{\alpha}{N_r} \sum_{x_r \in \mathcal{D}_r} d(x_r; w^u) + \frac{\gamma}{N_r} \sum_{(x_r, y_r) \in \mathcal{D}_r} l(f(x_r; w^u), y_r) - \frac{1}{N_f} \sum_{x_f \in \mathcal{D}_f} d(x_f; w^u)$$

Retained data
Minimize divergence with teacher

Retained data
Minimize task loss

Forget data
Maximize divergence with teacher

We want to forget $x_f \in D_f$, while remember $x_r \in D_r$

Combining SISA and SCRUB

SCRUB inlearn: 3 optimization goal

1. Stay close to teacher on retain set
2. By good on task objective on retain set
3. Stay away from teacher on forget set

```
criterion_retain = nn.KLDivLoss()
criterion_obj = nn.CrossEntropyLoss()
criterion_forget = nn.KLDivLoss()
```

SCRUB pseudocode

Algorithm 1 SCRUB

Require: Teacher weights w^o
Require: Total max steps MAX-STEPS
Require: Total steps STEPS
Require: Learning rate ϵ
  $w^u \leftarrow w^o$
  $i \leftarrow 0$
  repeat
    if $i <$ MAX-STEPS **then**
      $w^u \leftarrow$ DO-MAX-EPOCH(w^u)
    end if
    $w^u \leftarrow$ DO-MIN-EPOCH(w^u)
  until $i <$ STEPS

SCRUB pseudocode

Algorithm 2 DO-MAX-EPOCH

Require: Student weights w^u
Require: Learning rate ϵ
Require: Batch size B
Require: Forget set D_f
Require: Procedure NEXT-BATCH
$b \leftarrow \text{NEXT-BATCH}(D_f, \text{B})$
repeat
$$w^u \leftarrow w^u + \epsilon \nabla_{w^u} \frac{1}{|b|} \sum_{x_f \in b} d(x_f; w^u)$$
$b \leftarrow \text{NEXT-BATCH}(D_f, \text{B})$
until b

Algorithm 3 DO-MIN-EPOCH

Require: Student weights w^u
Require: Learning rate ϵ
Require: Batch size B
Require: Retain set D_r
Require: Procedure NEXT-BATCH
$b \leftarrow \text{NEXT-BATCH}(D_r, \text{B})$
repeat
$$w^u \leftarrow w^u - \epsilon \nabla_{w^u} \frac{1}{|b|} \sum_{(x_r, y_r) \in b} \alpha d(x_r; w^u) + \gamma l(f(x_r; w^u), y_r)$$
$b \leftarrow \text{NEXT-BATCH}(D_r, \text{B})$
until b

Experiment

Dataset

- CIFAR-10

Model

- ResNet50, pretrained on ImageNet

Framework

- Pytorch (v2.0.1 + cuda support)
  - For model and training process
- (python library) Adversarial Robustness Toolbox (ART)
  - For metrics

Code reference

SISA

- https://github.com/cleverhans-lab/machine-unlearning

SCRUB

- https://github.com/meghdadk/SCRUB

Hardware

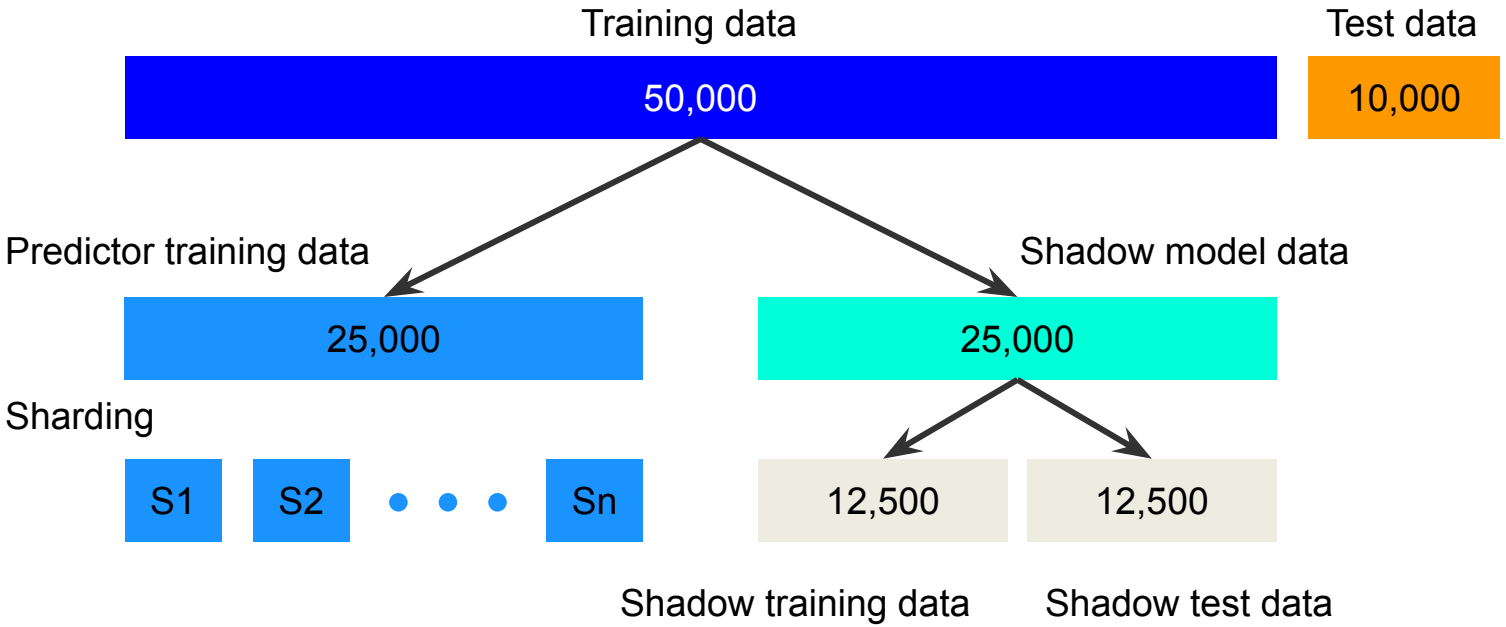
Processor	Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz 2.21 GHz
Installed RAM	16.0 GB (15.9 GB usable)
System type	64-bit operating system, x64-based processor
GPU	NVIDIA GeForce RTX 2070, 16GB

Experiment setup

- 3 sharding scenario
  - 1 shard, 3 shards, 10 shards
- Using normal unlearning vs SCRUB unlearning on the shards
- 2 unlearn scenario
  - 100 samples (~0.8%)
  - 1000 samples (~8%)

Experiment setup

- Data split



Experiment setup

- Training parameter
  - resnet50, pretrained on ImageNet
  - 10 epoch training
  - SGD optimizer, lr = 1e-3, momentum=0.9

Metric

- Classification accuracy
  - Check how much unlearned model performance has deteriorated compared to base model (naive unlearn model)
- Membership inference attack (MIA) using shadow model
  - A substitute measure for unlearn success
  - attack success should be about the same as naive unlearn model
  - If attack success is high when naive unlearn model attack success is low, then it is a bad unlearning.
- Training time
  - If training time is slow compared to naive unlearning, then it's an inefficient unlearning

Metric

- Model classification loss (cross entropy loss)
  - Member vs non-member
- Model average confidence (of predicted class)
  - Member vs non-member

Looking for (in unlearned model)

1. Unlearn process should not hard accuracy
2. Unlearn process should be faster than naive retrain
3. Model loss on **retain data** should be similar to training data
    a. Same goes for prediction confidence
4. Model loss on **forget data** should be similar to test data
    a. Same goes for prediction confidence
5. Attack success rate on **retain data** should be similar to training data
6. Attack success rate on **forget data** should be similar to test data

Baseline model

- 1, 3, 10 SISA model with naive retrain on each shard
- (1 shard) SCRUB unlearn

Overall process

1. Split the dataset
2. Train independent predictor model for each shard (record the training time)
3. Train shadow model
    a. Randomly split 50-50 on shadow dataset
    b. Train each shadow model on different split (same data, just split at different location)
4. Use shadow model, generate attack data (actual data, model's label, actual label)
5. Train attack model

Overall process (cont.)

5. Train attack model
6. Run evaluation on original model
    a. Accuracy
        i. member (training) set, non-member (test) set, retain set, forget set
    b. Cross Entropy Loss
    c. Average prediction confidence (softmax output)
    d. Membership attack success rate
7. Make unlearn dataset (forget set, retain set)
8. Repeat 6

Result

Not very good

- Attacker model doesn't learn
  - attacker too weak
  - Target too strong
    - The accuracy gap is small

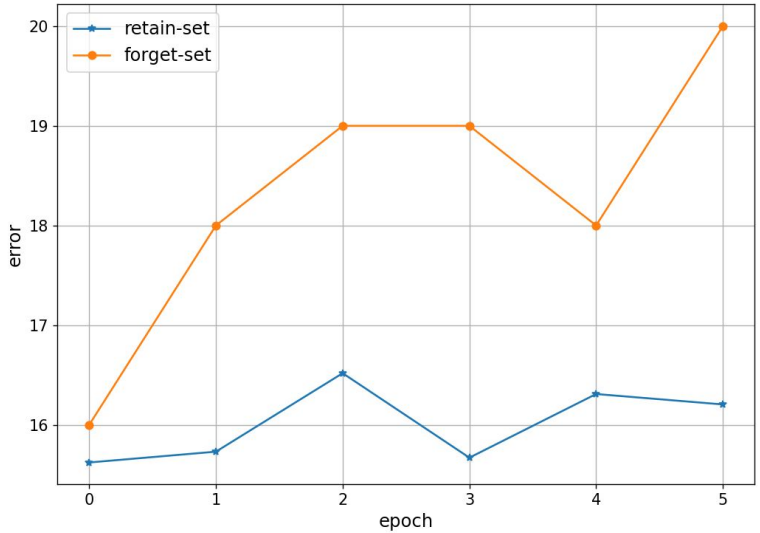
```
Epoch 11/20, Loss:  0.693002,Train Acc:  50.161331 Val Acc:  49.475960
Epoch 12/20, Loss:  0.692967,Train Acc:  50.166668 Val Acc:  49.467957
Epoch 13/20, Loss:  0.692940,Train Acc:  50.172890 Val Acc:  49.427956
Epoch 14/20, Loss:  0.692928,Train Acc:  50.189777 Val Acc:  49.523964
Epoch 15/20, Loss:  0.692909,Train Acc:  50.217331 Val Acc:  49.459957
Epoch 16/20, Loss:  0.692870,Train Acc:  50.203110 Val Acc:  49.475960
Epoch 17/20, Loss:  0.692861,Train Acc:  50.217331 Val Acc:  49.483959
Epoch 18/20, Loss:  0.692860,Train Acc:  50.214664 Val Acc:  49.483959
Epoch 19/20, Loss:  0.692843,Train Acc:  50.213776 Val Acc:  49.515961
Epoch 20/20, Loss:  0.692837,Train Acc:  50.237778 Val Acc:  49.499962
[INFO] attack model training accuracy = 0.5021733045578003
```

Attacker model doesn’t learn

- Model is pre-trained
- Most layer was frozen, and only the last layer changes

Evaluating model loss

Model's error rate on retain/forget set during SCRUB unlearn process



Result

model type	1 shard	1 shard (SCRUB)	3 shard	3 shard (SCRUB)
Task acc				
original (train)	0.7561		0.7239	
original (test)	0.7356		0.6964	
100-point unlearn (retrain)	0.7563	0.8408	0.7242	0.7246
100-point unlearn (forget)	0.74	0.85	0.6965	0.6971
1000-point unlearn (retrain)	0.755	0.8453	0.723	0.724
1000-point unlearn (forget)	0.716	0.846	0.6944	0.6976

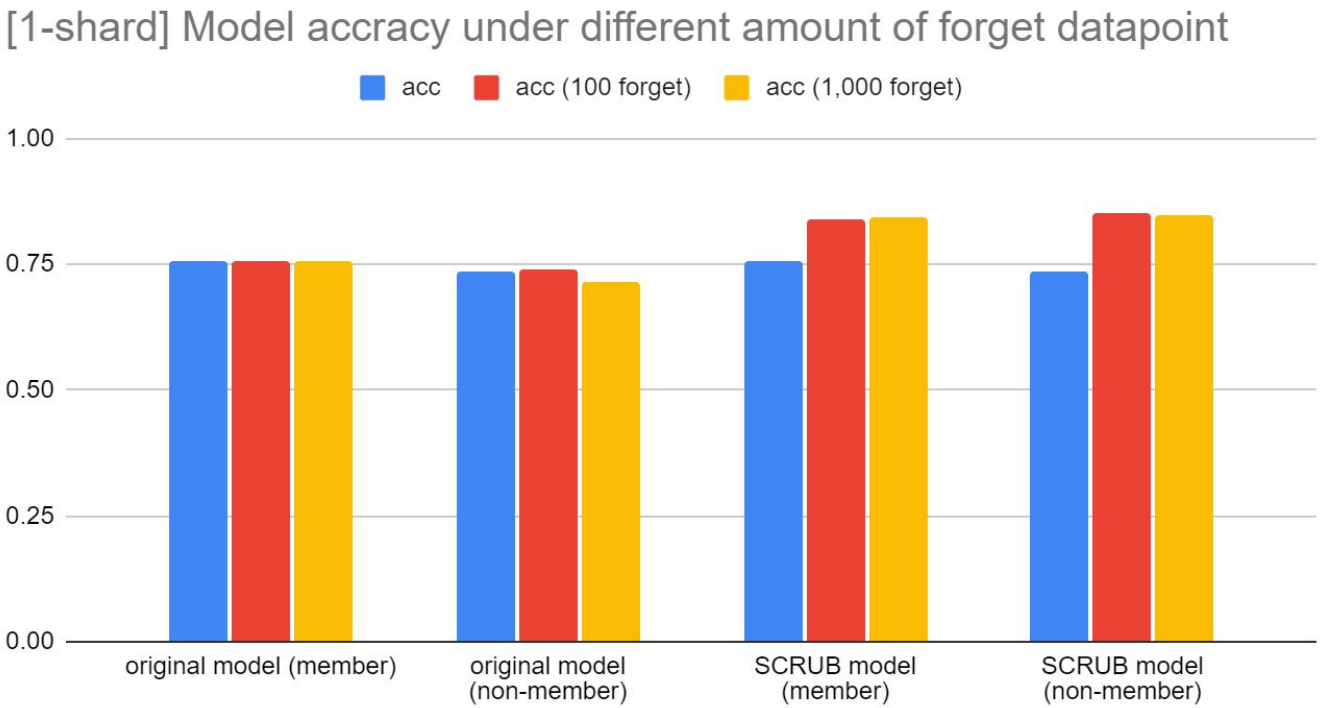
Result

model type	1 shard	1 shard (SCRUB)	3 shard	3 shard (SCRUB)
Model loss				
original (train)	0.7546		0.9234	
original (test)	0.7978		0.9746	
100-point unlearn (retrain)	0.7821	0.4514	0.9245	0.9197
100-point unlearn (forget)	0.826	0.4127	0.9752	0.9725
1000-point unlearn (retrain)	0.7604	0.4432	0.9315	0.9233
1000-point unlearn (forget)	0.8293	0.4527	0.9831	0.9724

Result

model type	1 shard	1 shard (SCRUB)	3 shard	3 shard (SCRUB)
Model confidence				
original (train)	0.6602		0.5466	
original (test)	0.6563		0.5402	
100-point unlearn (retrain)	0.6327	0.8342	0.5471	0.5487
100-point unlearn (forget)	0.6133	0.8314	0.5405	0.5421
1000-point unlearn (retrain)	0.6561	0.8382	0.5428	0.5478
1000-point unlearn (forget)	0.6481	0.8402	0.5358	0.5413
Training Time (seconds)				
original training time	2045.0419		1513.5583	
100-point unlearn	1901.6681	1412.399	615.7019	517.8624
1000-point unlearn	1607.2881	1421.4885	524.1072	467.7787

Result


[1-shard] Model accracy under different amount of forget datapoint
acc
acc (100 forget)
acc (1,000 forget)
1.00
0.75
0.50
0.25
0.00
original model (member)
original model (non-member)
SCRUB model (member)
SCRUB model (non-member)

Analysis and interpretation

- SISA+SCRUB achieves the goal of unlearning
  - metrics on retain set is similar to train set
  - metrics on forget set is similar to test set
- However, the unlearn time of SCRUB remains constant regardless of unlearn size
  - On other hand, naive retrain is faster when unlearn size is large.
- SCRUB process also optimize the model in the process
  - Lead to higher metric performance
  - Not visible in 3-shard scenario due to voting

Conclusion

Conclusion

1. Model unlearning using SISA+SCRUB
    a. Model agnostic
    b. can handle different unlearn sensitivity profile on different data points
        i. Some data might be more likely to get unlearn (sensitive), while some data might never get unlearn (public)
2. Metric profile on retain set is similar to training set
    a. on forget set is similar to test set

Flaw

Note: will refer to fully-trained model as saturated model

1. Attack model not working as intended
    a. Could be that attack is not strong enough
    b. Attack with only 5 shadow model
    c. Both shadow model and original model are not saturated, which could lead to larger difference between shadow model and target model
    d. The difference in training error and test error on target model is quite low (~2%), which lead to smaller gap the attacker can exploit

Flaw

2. SCRUB model has high metric score than original data
    a. SCRUB process also optimize the model
    b. Not as pronounced in saturated model, but very noticeable effect on non-saturated model
    c. Hard to make fair comparison on non-saturated model

Interesting question

1. How to guarantee unlearning in model with differential privacy?
2. What should be the result when unlearn the whole dataset?
    a. Exact unlearn
    b. Approximate unlearn

Bonus: https://awesome-machine-unlearning.github.io/

Reference

- Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C. A., Jia, H., Travers, A., Zhang, B., ... & Papernot, N. (2021, May). Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)* (pp. 141-159). IEEE.
- Nguyen, T. T., Huynh, T. T., Nguyen, P. L., Liew, A. W. C., Yin, H., & Nguyen, Q. V. H. (2022). A survey of machine unlearning. *arXiv preprint arXiv:2209.02299*.
- Kurmanji, M., Triantafillou, P., Hayes, J., & Triantafillou, E. (2024). Towards unbounded machine unlearning. *Advances in Neural Information Processing Systems*, *36*.
- Shokri, R., Stronati, M., Song, C., & Shmatikov, V. (2017, May). Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)* (pp. 3-18). IEEE.

Appendix

SISA pseudocode

Specifically, each shard's data $\mathcal{D}_k$ is further uniformly partitioned into R disjoint *slices* such that $\cap_{i\in[R]}\mathcal{D}_{k,i} = \varnothing$ and $\cup_{i\in[R]}\mathcal{D}_{k,i} = \mathcal{D}_k$. We perform training for e epochs to obtain M_k as follows:

1) At step 1, train the model *using random initialization* using only $\mathcal{D}_{k,1}$, for e_1 epochs. Let us refer to the resulting model as $M_{k,1}$. Save the state of parameters associated with this model.
2) At step 2, train the model $M_{k,1}$ using $\mathcal{D}_{k,1} \cup \mathcal{D}_{k,2}$, for e_2 epochs. Let us refer to the resulting model as $M_{k,2}$. Save the parameter state.
3) At step R, train the model $M_{k,R-1}$ using $\cup_i \mathcal{D}_{k,i}$, for e_R epochs. Let us refer to the resulting *final* model as $M_{k,R} = M_k$. Save the parameter state.

SCRUB pseudocode

Algorithm 1 SCRUB

Require: Teacher weights w^o
Require: Total max steps MAX-STEPS
Require: Total steps STEPS
Require: Learning rate ϵ
  $w^u \leftarrow w^o$
  $i \leftarrow 0$
  repeat
    if $i <$ MAX-STEPS **then**
      $w^u \leftarrow$ DO-MAX-EPOCH(w^u)
    end if
    $w^u \leftarrow$ DO-MIN-EPOCH(w^u)
  until $i <$ STEPS

SCRUB pseudocode

Algorithm 2 DO-MAX-EPOCH

Require: Student weights w^u
Require: Learning rate ϵ
Require: Batch size B
Require: Forget set D_f
Require: Procedure NEXT-BATCH
$b \leftarrow \text{NEXT-BATCH}(D_f, \text{B})$
repeat
$$w^u \leftarrow w^u + \epsilon \nabla_{w^u} \frac{1}{|b|} \sum_{x_f \in b} d(x_f; w^u)$$
$b \leftarrow \text{NEXT-BATCH}(D_f, \text{B})$
until b

Algorithm 3 DO-MIN-EPOCH

Require: Student weights w^u
Require: Learning rate ϵ
Require: Batch size B
Require: Retain set D_r
Require: Procedure NEXT-BATCH
$b \leftarrow \text{NEXT-BATCH}(D_r, \text{B})$
repeat
$$w^u \leftarrow w^u - \epsilon \nabla_{w^u} \frac{1}{|b|} \sum_{(x_r, y_r) \in b} \alpha d(x_r; w^u) + \gamma l(f(x_r; w^u), y_r)$$
$b \leftarrow \text{NEXT-BATCH}(D_r, \text{B})$
until b