A Repository of Jupyter Notebooks on Unlearning in Federated Learning

Final Presentation

FYP22002 @ HKUCS

Sheng “Victor” HUANG

Supervisor: Prof. S M YIU

20 Apr 2023

Background

- Example
  - ChatGPT -> what data is used?
    - Online public text databases, 570GB, 300 billion words [BBC23]
    - Also user conversations [OpenAI23]

6. **Will you use my conversations for training?**
   - Yes. Your conversations may be reviewed by our AI trainers to improve our systems.

[BBC23] https://www.sciencefocus.com/future-technology/gpt-3/
[OpanAI] https://help.openai.com/en/articles/6783457-what-is-chatgpt

Background

- Security?

Zero-Shot Information Extraction via Chatting with ChatGPT

Xiang Wei[1], Xingyu Cui[1], Ning Cheng[1], Xiaobin Wang[2], Xin Zhang, Shen Huang[2], Pengjun Xie[2], Jinan Xu[1], Yufeng Chen[1], Meishan Zhang, Yong Jiang[2], and Wenjuan Han[1]
[1] Beijing Jiaotong University, Beijing, China
[2] DAMO Academy, Alibaba Group, China

Exploring the Feasibility of ChatGPT for Event Extraction

Jun Gao[1]* Huan Zhao[2] Changlong Yu[3] Ruifeng Xu[1]
[1]Harbin Institute of Technology (Shenzhen)
[2]4Paradigm. Inc. [3]HKUST, Hong Kong, China
imgaojun@gmail.com zhaohuan@4paradigm.com cyuaq@cse.ust.hk

Extracting Accurate Materials Data from Research Papers with Conversational Language Models and Prompt Engineering - Example of ChatGPT

Maciej P. Polak* and Dane Morgan†
Department of Materials Science and Engineering,
University of Wisconsin-Madison, Madison, Wisconsin 53706-1595, USA

Background

- Privacy?

Respecting privacy

Our large language models are trained on a broad corpus of text that includes publicly available content, licensed content, and content generated by human reviewers. We don't use data for selling our services, advertising, or building profiles of people—we use data to make our models more helpful for people. ChatGPT, for instance, improves by further training on the conversations people have with it.

While some of our training data includes personal information that is available on the public internet, we want our models to learn about the world, not private individuals. So we work to remove personal information from the training dataset where feasible, fine-tune models to reject requests for personal information of private individuals, and respond to requests from individuals to delete their personal information from our systems. These steps minimize the possibility that our models might generate responses that include the personal information of private individuals.

Background

- Usability & Fidelity?

4. **Can I trust that the AI is telling me the truth?**
   - ChatGPT is not connected to the internet, and it can occasionally produce incorrect answers. It has limited knowledge of world and events after 2021 and may also occasionally produce harmful instructions or biased content.

     We'd recommend checking whether responses from the model are accurate or not. If you find an answer is incorrect, please provide that feedback by using the "Thumbs Down" button.

Background

- Inappropriate use of data -> delete data
  - Right to be Forgotten
  - OpenAI based in San Francisco, CA

How can I delete my account?

I'd like to delete my account, but I can't find any way to do this.

Written by Johanna C.. Updated yesterday

Account Deletion (Two Methods)

⚠️ **UPDATE**: As of April 11, 2023 ChatGPT users on either free or plus plans can delete their own accounts themselves:

Background

- Is that enough?
  - Data deleted from database may still exist in ChatGPT's models
  - May still create problems of security, privacy, usability or fidelity

REUTERS® World Business Markets Legal Breakingviews Technology Investigations

Technology

3 minute read · April 1, 2023 5:40 AM GMT+8 · Last Updated 19 days ago

Italy curbs ChatGPT, starts probe over privacy concerns

By Elvira Pollina and Supantha Mukherjee

中共中央网络安全和信息化委员会办公室

Office of the Central Cyberspace Affairs Commission

WWW.CAC.GOV.CN

请输入检索关键词

首页 权威发布 办公室工作 网络安全 信息化 网络传播 国际交流 地方网信 执法督查 政策法规 互动中心 教育培训 业界动态 工作专题

当前位置：首页 > 正文

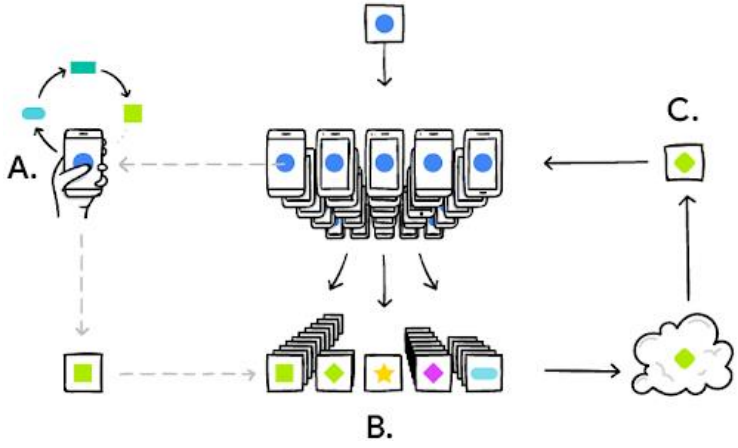
国家互联网信息办公室关于《生成式人工智能服务管理办法（征求意见稿）》公开征求意见的通知

Machine Unlearning [CY15]

- Remove data and data influence from ML models
- ML models as black boxes [KL17]
- Naïve method -> to retrain on remaining data -> slow
  - Exact unlearning, only with variations of retraining
  - Otherwise approximate unlearning
- Challenges:
  - Stochasticity of training
    - Data influence difficult to track
  - Incrementality of training
    - Training with a data point affected by prior training, and will affect later training
  - Catastrophic unlearning
    - Unlearnt models generally perform worse than retrained models

[CY15] https://ieeexplore.ieee.org/document/7163042
[KL17] https://proceedings.mlr.press/v70/koh17a

Federated Learning (FL) [McM+17]



- Multiple devices to collaboratively train a model without sharing data
- Has potential application with ChatGPT
  - E.g. Healthcare
- Not immune from privacy vulnerabilities present in other ML techniques
  - Unlearning is still needed

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
  initialize w_0
  for each round $t = 1, 2, \ldots$ **do**
    $m \leftarrow \max(C \cdot K, 1)$
    $S_t \leftarrow$ (random set of m clients)
    for each client $k \in S_t$ **in parallel do**
      $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
    $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w)**:** *// Run on client k*
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
  for each local epoch i from 1 to E **do**
    for batch $b \in \mathcal{B}$ **do**
      $w \leftarrow w - \eta \nabla \ell(w; b)$
  return w to server

[McM+17] https://proceedings.mlr.press/v54/mcmahan17a.html

Federated Unlearning

Additional Challenges

- Limited data access
  - Server doesn’t have access to training data used at the client side
- Limited client participation
  - Client goes offline and can’t participate in unlearning process
- Complicated relationship between data and clients
  - Unlearning all data of one client of just part of it?
  - What if there is data overlap on other clients?
- Data partition
  - Horizontal FL, vertical FL and federated transfer learning
- Statistical heterogeneity
  - IID (Independent and Identically Distributed)
- Adversarial model
  - Untruthful server/clients
- …

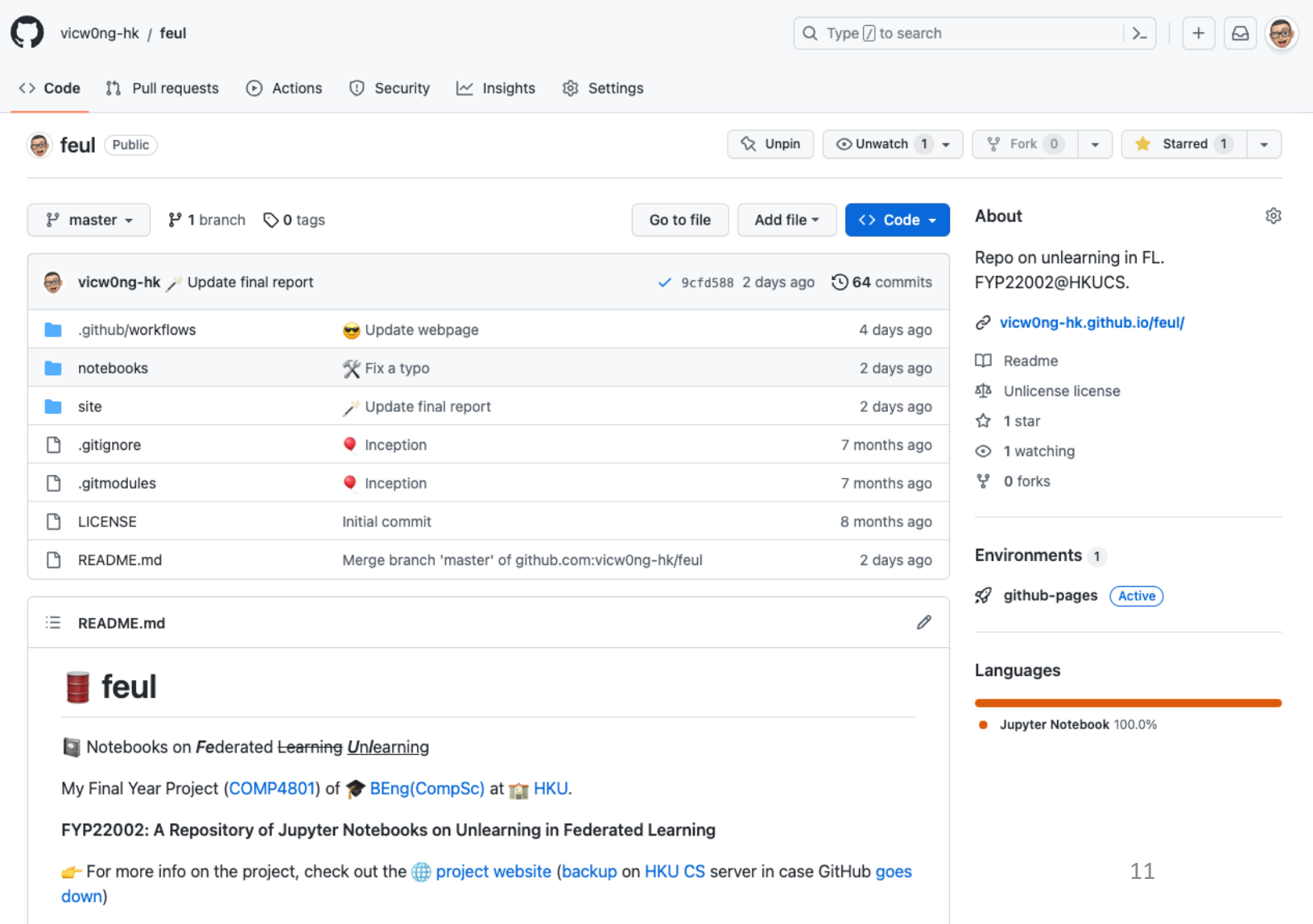
How to Study Federated Unlearning?

Introducing:

A Repository of Jupyter Notebooks on Unlearning in Federated Learning

Objective:

To organize knowledge and study unlearning in FL



Methods

- Literature Review
  - Google Scholar “federated+unlearning”
    - total 17 papers on federated unlearning, 2021-2023
  - Several papers on unlearning in general and other privacy-preserving techniques are also reviewed for context
- Jupyter Notebooks
  - Summarizing and referencing key content from reviewed papers
- Experiments
  - Adapting code from respective papers
  - testing in different environments – HKUCS GPU Farm & Google Colab
  - Fixing bugs and issues
  - Record instructions in notebooks
- Repo and website hosting
  - Git, GitHub, Hugo, GitHub Pages, GitHub Actions

Results

📝 Description

This repo aims to provide materials, in the form of Jupyter Notebooks, for studying machine unlearning in federated learning (FL), both of which are new research areas that are developing rapidly.

> ⓘ Note The data collection ended in early April 2023 and I have done my best to scrape the internet on this topic. If you are checking out this repo in the future, you may find it incomplete. However, it is helpful that we have this snapshot, so that future researchers can see how this new area grew and take inspirations from earlier research.

The contents of the notebooks consist of

1. general introduction to the area and it's background (0-2);
2. research progress in 2021 (3-5);
3. research progress in 2022 (6-15);
4. research progress in 2023 (until early April) (16-19).

The notebooks, except those in the 1st part, are ordered according to the publication dates (or conference dates, or last edit date of preprints) of the papers.

📖-marked notebooks are summaries of the paper with key concepts, figures and tables presented.
💻-marked notebooks are instructions on how to run the code for the design algorithms, with tweaks made to get rid of bugs that usually exist in the original code.
Clicking on 📖 or 💻 can open the corresponding notebook in Google Colab.

📚 Recommended prerequisites

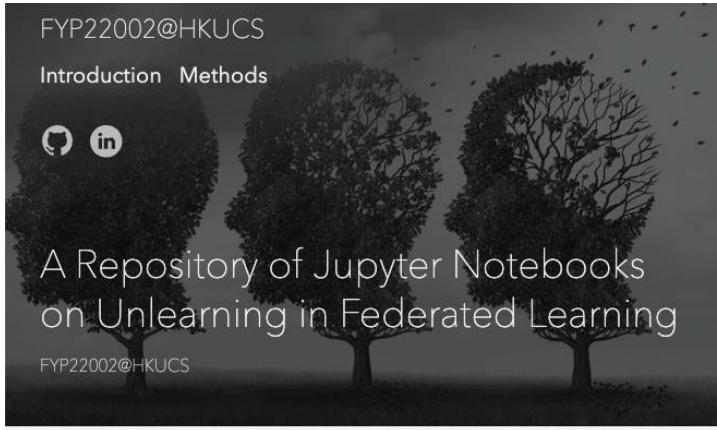
- Machine Learning
- Computer Security
- Operating Systems

🎲 **Optional**:

- Cryptography
- Distributed Systems

🗂 Contents

#	Topics	Contents	References
0	👋 Intro to unlearning and FL	📖 intro-unlearning.ipynb 📖 intro-fl.ipynb 💻 code-amnesiac-ml.ipynb 💻 code-flwr.ipynb	[1][2][3][4][5][6][7][8][9]
1	🔍 More on unlearning	📖 unlearning-definition.ipynb 📖 unlearning-framework.ipynb	[1]
2	🌉 Unlearning in FL	📖 ul-in-fl.ipynb	[1][8]
3	🎉 FedEraser: 1st attempt	📖 federaser.ipynb 💻 code-federaser.ipynb	[10]
4	🌳 RevFRF: federated unlearning in RF	📖 revfrf.ipynb	[11]
5	🎲 Bayesian variational FL and unlearning	📖 bayesian-variational.ipynb	[12]
6	🧪 Federated unlearing with distillation	📖 distillation.ipynb	[13]
7	✂️ Federated unlearning with class-discriminative pruning	📖 channel-prune.ipynb	[14]
8	🏋 Federated unlearing with rapid retraining	📖 rapid-retrain.ipynb 💻 code-rapid-retrain.ipynb	[15]
9	💫 Forget-SVGD: particle-based Bayesian federated unlearning	📖 forget-svgd.ipynb	[16]
10	✔️ VeriFi: Verifiable federated unlearning	📖 verifi.ipynb	[17]
11	👷 Client opt-out	📖 opt-out-unlearning.ipynb	[18]
12	💊 FedRecover: recover from poison	📖 fedrecover.ipynb	[19]
13	👥 Unlearning of federated clusters	📖 unlearning-cluster.ipynb	[20]
14	🚀 Unlearning in federated optimization	📖 sequential-informed.ipynb	[21]
15	🧩 General Pipeline	📖 federated-unlearning.ipynb	[22]
16	🌌 Subspace-based federated unlearning	📖 subspace.ipynb	[23]
17	📊 Federated knowledge graph embedding learning and unlearning	📖 heterogeneous-kg-embedding.ipynb	[24]
18	👍 Federated unlearning for on-device recommendation	📖 on-device-recommend.ipynb	[25]
19	🔗 Knot: asynchronous federated unlearning	📖 knot.ipynb 💻 code-knot.ipynb	[26]



Welcome to the BEng(CompSc) FYP of Victor S. HUANG, supervised by Prof. S.M. YIU.

🎉👉 Check out the repo 👈🎉

Introduction | Methods

Update

1st Presentation
Final Presentation

Project Plan | Interim Report | Final Report

Made with ❤️ by Victor

Results

vicw0ng-hk / feul

Code | Pull requests | Actions | Security | Insights | Settings

Code

master

Go to file

- .github
- notebooks
  - 00-intro-ul-fl
    - code-amnesiac-ml.ipynb
    - code-flwr.ipynb
    - intro-fl.ipynb
    - intro-unlearning.ipynb
  - 01-ul-more
  - 02-un-in-fl
  - 03-liu+21a
  - 04-liu+21b
  - 05-gsk21
  - 06-wzm22
  - 07-wan+22
  - 08-liu+22
  - 09-gon+22
  - 10-gao+22
  - 11-hal+22
  - 12-cao+22
  - 13-pan+22
  - 14-fra+22
  - 15-wu+22
  - 16-li+23
  - 17-zlh23
  - 18-yua+23
  - 19-sl23
- site
- .gitignore
- .gitmodules
- LICENSE
- README.md

feul / notebooks / 00-intro-ul-fl / intro-unlearning.ipynb

vicw0ng-hk 😎 Simple name change 4615777 · 5 days ago History

Preview 1 lines (1 loc) · 8.15 KB Raw

Machine Unlearning

This Notebook gives you a brief introduction to machine unlearning.

Background

Today, we put a lot of data into artificial intelligence (AI), especially for the training of machine learning (ML) models. Some of the data, though, can be of personal and private nature, such as your location, biometrics and medical records. It is natural that we want our data to be used and stored securely.

At the same time, ML systems are also prone to attacks like other types of systems, such as membership inference, property inference, model stealing, data poisoning, etc. These attacks could lead to problems such as data leakage and wrong predictions. Or, in some cases such as poor data quality or system design, these problems can still occur even without an adversarial third-party. If you know something about ML, you'll know that ML models can sometimes "memorize" instead of "learn from" data, often as a result of overfitting. And that's dangerous! Because it makes it so much easier to run model extraction attacks against such a model and data may well be leaked. Thus, it makes sense for us to want to delete some of the data, or erase some of the training, from our trained model.

Deleting data can be as easy as a single SQL execution in some systems, but it's not that straightforward in ML models. In an ML model, the connection between parameters and data is not clearly shown, hence we are unable to map a single data point's influence throughout the training process. Thus, it is difficult to remove information relating to a single data point from a trained ML model. In other words, it can be said that it is challenging to induce targeted "memory loss" in ML models, or to make ML models "forget" that it ever trained with specific data.

Definition

The process of deleting certain data point(s) from ML systems is called **machine unlearning**, or simply **unlearning**, which is a term first proposed by Cao and Yang in 2015.

As an obvious solution, we can retrain an ML model from scratch, using all the data points but the ones we want to erase. This will ensure that the targeted data points are not in the ML system, but this process is very slow. So, ideally an unlearning paradigm should be reasonably efficient.

We will talk about more aspects of unlearning defitions in a later notebook.

vicw0ng-hk / feul

Code | Pull requests | Actions | Security | Insights | Settings

Code

master

Go to file

- .github
- notebooks
  - 00-intro-ul-fl
    - code-amnesiac-ml.ipynb
    - code-flwr.ipynb
    - intro-fl.ipynb
    - intro-unlearning.ipynb
  - 01-ul-more
  - 02-un-in-fl
  - 03-liu+21a
  - 04-liu+21b
  - 05-gsk21
  - 06-wzm22
  - 07-wan+22
  - 08-liu+22
  - 09-gon+22
  - 10-gao+22
  - 11-hal+22
  - 12-cao+22
  - 13-pan+22
  - 14-fra+22
  - 15-wu+22
  - 16-li+23
  - 17-zlh23
  - 18-yua+23
  - 19-sl23
- site
- .gitignore
- .gitmodules
- LICENSE
- README.md

feul / notebooks / 00-intro-ul-fl / intro-fl.ipynb

vicw0ng-hk 😎 Simple name change 4615777 · 5 days ago History

Preview 1 lines (1 loc) · 4.72 KB Raw

Federated Learning (FL)

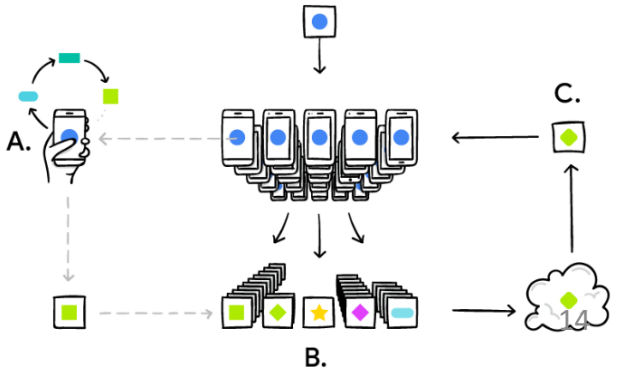
This Notebook gives you a brief introduction to FL.

Background

Similar to machine unlearning, FL also tries to solve security and privacy problems arising from the fact that a large amount of data is used in ML. You may check the `intro-unlearning.ipynb` Notebook for detailed implications of data security and privacy in ML systems. FL, however, takes a different approach to achieve privacy and security, which means there are some overlapping issues in the cross section between FL and unlearning (that is the main topic of this project). For now, let's keep focusing on FL.

In traditional settings, data is stored and processed at a centralized location in a centralized manner. All this centralization used with data can lead to security and privacy problems. Despite efforts to mitigate such problems in centralized settings, decentralized methods have been introduced to provide an alternative pathway towards better security and privay protection. However, as one can clearly see that data collection and processing capabilities on individual decentralized machine (client) may not be adequate to train an ML model with good enough performance, there probably needs to be a machanism for orchestrating multiple clients. FL is one such method.

In a typical FL setting (there are, of course, many variations), an ML model is initialized at a central server and then sent to participating clients. The clients train the model with the data that they have, generating an update on the model (sub-update), and then send the sub-update to the server. The server then use the received sub-updates to update (ususally by aggregating) its own model. The updated central model is then sent to clients for further training.



Results

vicw0ng-hk / feul

Type / to search

Code | Pull requests | Actions | Security | Insights | Settings

Code

master

Go to file t

- .github
- notebooks
  - 00-intro-ul-fl
    - code-amnesiac-ml.ipynb
    - code-flwr.ipynb
    - intro-fl.ipynb
    - intro-unlearning.ipynb
  - 01-ul-more
  - 02-un-in-fl
  - 03-liu+21a
  - 04-liu+21b
  - 05-gsk21
  - 06-wzm22
  - 07-wan+22
  - 08-liu+22
  - 09-gon+22
  - 10-gao+22
  - 11-hal+22
  - 12-cao+22
  - 13-pan+22
  - 14-fra+22
  - 15-wu+22
  - 16-li+23
  - 17-zlh23
  - 18-yua+23
  - 19-sl23
- site
- .gitignore
- .gitmodules
- LICENSE
- README.md

feul / notebooks / 00-intro-ul-fl / code-amnesiac-ml.ipynb

vicw0ng-hk 😎 Simple name change 4615777 · 5 days ago History

Preview 1 lines (1 loc) · 111 KB Raw

Machine Unlearning by AmnesiacML

This Notebook is adapted from Amnesiac Machine Learning.

You are strongly advised to run this Notebook with CUDA-enabled GPU on your machine or with a GPU runtime on Colab, due to a large amount of tensor calculation.

Imports and Setup

In [1]:
```
import torch
import torchvision
import numpy as np
import matplotlib.pyplot as plt
from torchvision import datasets, transforms, models
from torch import nn, optim
from torch.nn import functional as F
from torch.autograd import Variable
from scipy import ndimage
import copy
import random
import time
import pickle

torch.set_printoptions(precision=3)
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
```

Data Entry and Processing

In [2]:
```
# Transform image to tensor and normalize features from [0,255] to [0,1]
transform = transforms.Compose([transforms.ToTensor(),
                                transforms.Normalize((0.5,),(0.5,),(0.5)),
                                ])
```

In [3]:
```
# Using CIFAR100
traindata = datasets.CIFAR100('./data', download=True, train=True, transform=transform)
testdata = datasets.CIFAR100('./data', download=True, train=False, transform=transform)
```

```
Downloading https://www.cs.toronto.edu/~kriz/cifar-100-python.tar.gz to ./data/cifar-100-python.tar.gz
  0%|          | 0/169001437 [00:00<?, ?it/s]
Extracting ./data/cifar-100-python.tar.gz to ./data
Files already downloaded and verified
```

In [4]:
```
# Loaders that give 64 example batches
all_data_train_loader = torch.utils.data.DataLoader(traindata, batch_size=64, shuffle=True)
all_data_test_loader = torch.utils.data.DataLoader(testdata, batch_size=64, shuffle=True)
```

In [5]:
```
# Train dataloaders
```

vicw0ng-hk / feul

Type / to search

Code | Pull requests | Actions | Security | Insights | Settings

Code

master

Go to file t

- .github
- notebooks
  - 00-intro-ul-fl
    - code-amnesiac-ml.ipynb
    - code-flwr.ipynb
    - intro-fl.ipynb
    - intro-unlearning.ipynb
  - 01-ul-more
  - 02-un-in-fl
  - 03-liu+21a
  - 04-liu+21b
  - 05-gsk21
  - 06-wzm22
  - 07-wan+22
  - 08-liu+22
  - 09-gon+22
  - 10-gao+22
  - 11-hal+22
  - 12-cao+22
  - 13-pan+22
  - 14-fra+22
  - 15-wu+22
  - 16-li+23
  - 17-zlh23
  - 18-yua+23
  - 19-sl23
- site
- .gitignore
- .gitmodules
- LICENSE
- README.md

feul / notebooks / 00-intro-ul-fl / code-flwr.ipynb

vicw0ng-hk 🛠 Fix a typo 75fd413 · 2 days ago History

Preview 224 lines (224 loc) · 11.8 KB Raw

Federated Learning with Flower (`flwr`)

Flower (`flwr`) is a framework for building federated learning systems. The design of Flower is based on a few guiding principles:

- **Customizable**: Federated learning systems vary wildly from one use case to another. Flower allows for a wide range of different configurations depending on the needs of each individual use case.
- **Extendable**: Flower originated from a research project at the University of Oxford, so it was built with AI research in mind. Many components can be extended and overridden to build new state-of-the-art systems.
- **Framework-agnostic**: Different machine learning frameworks have different strengths. Flower can be used with any machine learning framework, for example, PyTorch, TensorFlow, Hugging Face Transformers, PyTorch Lightning, MXNet, scikit-learn, JAX, TFLite, Pandas for federated analytics, or even raw NumPy for users who enjoy computing gradients by hand.
- **Understandable**: Flower is written with maintainability in mind. The community is encouraged to both read and contribute to the codebase.

Flower with TensorFlow

To start with, let' try to build a federated learning system using Flower and TensorFlow.

Install required packages

1. Check out Python versions supporting both TensorFlow and Flower. (Flower's and TensorFlow's requirements)
2. Create a virtual environment using conda, virtualenv or other tools. Let's say you use Anaconda.

```
conda create -n flwr python # you can specify a Python version: "python" => "python=3.10"
```

3. Activate that environemnt.

```
conda activate flwr
```

4. Install TensorFlow and Flower

```
# Install Tensorflow
# Check out https://www.tensorflow.org/install/pip for latest instruction
# You may need to change codatoolkit and cudnn version depending on your setup and any future
change to TensorFlow
conda install -c conda-forge cudatoolkit=11.2 cudnn=8.1.0
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CONDA_PREFIX/lib/
python3 -m pip install tensorflow
# Verify install:
python3 -c "import tensorflow as tf; print(tf.config.list_physical_devices('GPU'))"

# Install Flower
pip install flwr
```

And we are good to go!

Significance

- There lacks work on a comprehensive overview of unlearning in FL
- This project is more accessible than academic survey papers
  - Researchers and practitioners can both benefit
- Provides clear information and instructions to guide the study from background knowledge to advanced research frontier.
  - Seasoned researchers can also find specific materials more quickly instead of going through the whole internet
- Promotes privacy and security in ML research and applications

Limitations

- Project stops in April 2023
  - Area still in early stages
    - Could be outdated soon, but good to have this snapshot
- Most papers have very narrow scope
  - E.g. focus on specific models, data partition, etc.
  - Difficult to generalize to the broader FL realm
- Few papers have published code
  - Difficult to recreate and compare quantitatively with other methods
- Paper quality varies
  - Some have impractical designs, e.g. backdoor attacks for verification [WZM22]

[WZM22] https://arxiv.org/abs/2201.09441

Some Takeaways

- FL has many variations. It is difficult to find a universal unlearning method that performs well.
  - One of the goals of research in this area is to find an unlearning method that requires little change to the existing FL designs, while achieving good performance.
- Unlearning itself may lead to unintended privacy risks [Che+21].
  - In practice, unlearning should be used along with other privacy-preserving methods.
- As of this moment, unlearning has not been practical in most ML scenarios, so other more practical measures should be given priority in real-world applications.

[Che+21] https://dl.acm.org/doi/abs/10.1145/3460120.3484756

Some Takeaways

- There has not been consensus on the differentiation of unlearning and data deletion in ML. Some have argued that the narrowly-defined unlearning is unlikely to achieve stricter privacy goals [CS23].
- FL is still in its early stages and not a practical privacy-enhancing technology (PET) yet [Boe+23], so there will be a lot of federated learning and unlearning research work done in the future.
- Some trends and open questions on unlearning: defining success of unlearning (verification / data auditing); unified unlearning requirements; unified unlearning benchmarking; adversarial machine unlearning; interpretable machine unlearning; …

[Boe+23] https://arxiv.org/abs/2301.04017
[CS23] https://arxiv.org/abs/2210.08911

Thank you!

- Repo at https://github.com/vicw0ng-hk/feul
- Website at https://vicw0ng-hk.github.io/feul

END
完