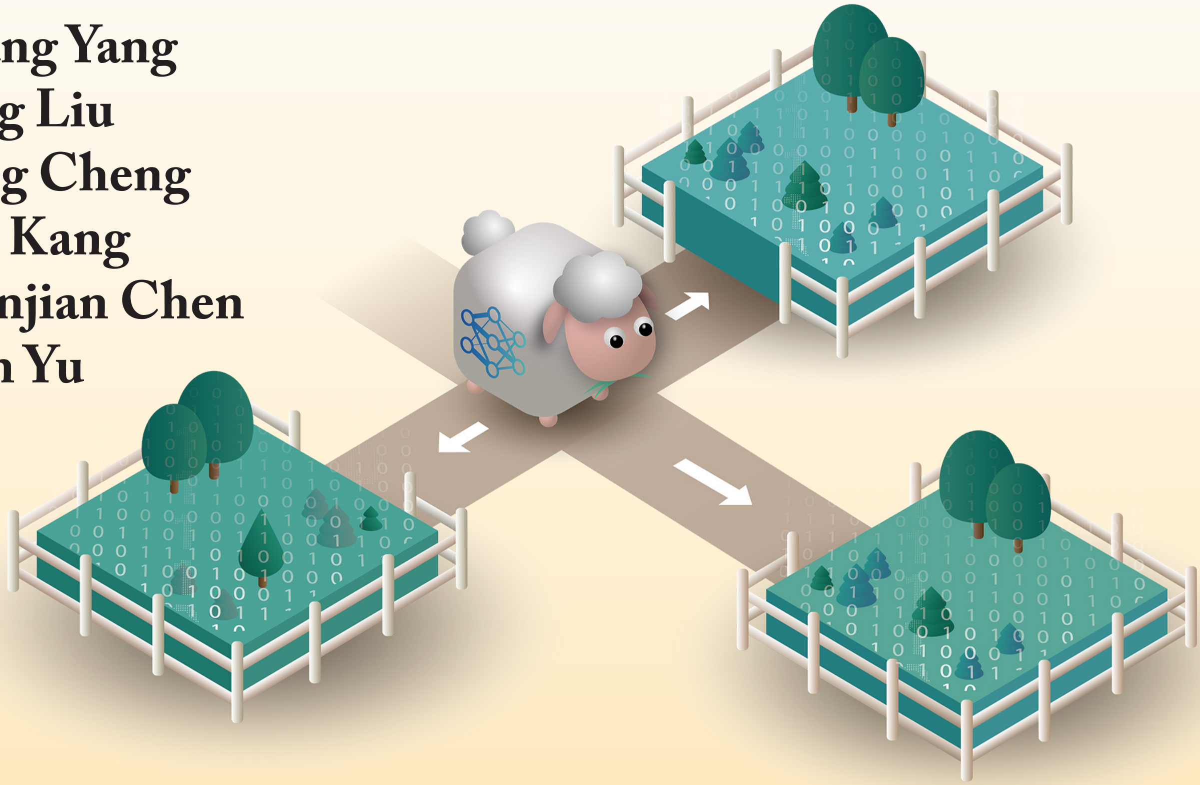
MORGAN & CLAYPOOL PUBLISHERS

Federated Learning

Qiang Yang
Yang Liu
Yong Cheng
Yan Kang
Tianjian Chen
Han Yu



SYNTHESIS LECTURES ON ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Ronald J. Brachman, Francesca Rossi, and Peter Stone, *Series Editors*

Federated Learning

Synthesis Lectures on Artificial Intelligence and Machine Learning

Editors
Ronald J. Brachman, ***Jacobs Technion-Cornell Institute at Cornell Tech***
Francesca Rossi, ***IBM Research AI***
Peter Stone, ***University of Texas at Austin***

Federated Learning
Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu
2019

An Introduction to the Planning Domain Definition Language
Patrik Haslum, Nir Lipovetzky, Daniele Magazzeni, and Christian Muise
2019

Reasoning with Probabilistic and Deterministic Graphical Models: Exact Algorithms, Second Edition
Rina Dechter
2019

Learning and Decision-Making from Rank Data
Lirong Xia
2019

Lifelong Machine Learning, Second Edition
Zhiyuan Chen and Bing Liu
2018

Adversarial Machine Learning
Yevgeniy Vorobeychik and Murat Kantarcioglu
2018

Strategic Voting
Reshef Meir
2018

Predicting Human Decision-Making: From Prediction to Action
Ariel Rosenfeld and Sarit Kraus
2018

Game Theory for Data Science: Eliciting Truthful Information
Boi Faltings and Goran Radanovic
2017

Multi-Objective Decision Making
Diederik M. Roijers and Shimon Whiteson
2017

Lifelong Machine Learning
Zhiyuan Chen and Bing Liu
2016

Statistical Relational Artificial Intelligence: Logic, Probability, and Computation
Luc De Raedt, Kristian Kersting, Sriraam Natarajan, and David Poole
2016

Representing and Reasoning with Qualitative Preferences: Tools and Applications
Ganesh Ram Santhanam, Samik Basu, and Vasant Honavar
2016

Metric Learning
Aurélien Bellet, Amaury Habrard, and Marc Sebban
2015

Graph-Based Semi-Supervised Learning
Amarnag Subramanya and Partha Pratim Talukdar
2014

Robot Learning from Human Teachers
Sonia Chernova and Andrea L. Thomaz
2014

General Game Playing
Michael Genesereth and Michael Thielscher
2014

Judgment Aggregation: A Primer
Davide Grossi and Gabriella Pigozzi
2014

An Introduction to Constraint-Based Temporal Reasoning
Roman Barták, Robert A. Morris, and K. Brent Venable
2014

Reasoning with Probabilistic and Deterministic Graphical Models: Exact Algorithms
Rina Dechter
2013

Introduction to Intelligent Systems in Traffic and Transportation
Ana L.C. Bazzan and Franziska Klügl
2013

A Concise Introduction to Models and Methods for Automated Planning
Hector Geffner and Blai Bonet
2013

Essential Principles for Autonomous Robotics
Henry Hexmoor
2013

Case-Based Reasoning: A Concise Introduction
Beatriz López
2013

Answer Set Solving in Practice
Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub
2012

Planning with Markov Decision Processes: An AI Perspective
Mausam and Andrey Kolobov
2012

Active Learning
Burr Settles
2012

Computational Aspects of Cooperative Game Theory
Georgios Chalkiadakis, Edith Elkind, and Michael Wooldridge
2011

Representations and Techniques for 3D Object Recognition and Scene Interpretation
Derek Hoiem and Silvio Savarese
2011

A Short Introduction to Preferences: Between Artificial Intelligence and Social Choice
Francesca Rossi, Kristen Brent Venable, and Toby Walsh
2011

Human Computation
Edith Law and Luis von Ahn
2011

Trading Agents
Michael P. Wellman
2011

Visual Object Recognition
Kristen Grauman and Bastian Leibe
2011

Learning with Support Vector Machines
Colin Campbell and Yiming Ying
2011

Algorithms for Reinforcement Learning
Csaba Szepesvári
2010

Data Integration: The Relational Logic Approach
Michael Genesereth
2010

Markov Logic: An Interface Layer for Artificial Intelligence
Pedro Domingos and Daniel Lowd
2009

Introduction to Semi-Supervised Learning
Xiaojin Zhu and Andrew B. Goldberg
2009

Action Programming Languages
Michael Thielscher
2008

Representation Discovery using Harmonic Analysis
Sridhar Mahadevan
2008

Essentials of Game Theory: A Concise Multidisciplinary Introduction
Kevin Leyton-Brown and Yoav Shoham
2008

A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence
Nikos Vlassis
2007

Intelligent Autonomous Robotics: A Robot Soccer Case Study
Peter Stone
2007

Federated Learning

Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu

www.morganclaypool.com

ISBN: 9781687336976 paperback
ISBN: 9781687336983 ebook
ISBN: 9781687336990 hardcover

DOI 10.2200/S00960ED2V01Y201910AIM043

A Publication in the Morgan & Claypool Publishers series
SYNTHESIS LECTURES ON ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Lecture #43
Series Editors: Ronald J. Brachman, *Jacobs Technion-Cornell Institute at Cornell Tech*
Francesca Rossi, *IBM Research AI*
Peter Stone, *University of Texas at Austin*
Series ISSN
Synthesis Lectures on Artificial Intelligence and Machine Learning
Print 1939-4608 Electronic 1939-4616

Federated Learning

Qiang Yang
WeBank and Hong Kong University of Science and Technology, China

Yang Liu
WeBank, China

Yong Cheng
WeBank, China

Yan Kang
WeBank, China

Tianjian Chen
WeBank, China

Han Yu
Nanyang Technological University, Singapore

SYNTHESIS LECTURES ON ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING #43

MORGAN & CLAYPOOL PUBLISHERS

ABSTRACT

How is it possible to allow multiple data owners to collaboratively train and use a shared prediction model while keeping all the local training data private? Traditional machine learning approaches need to combine all data at one location, typically a data center, which may very well violate the laws on user privacy and data confidentiality. Today, many parts of the world demand that technology companies treat user data carefully according to user-privacy laws. The European Union's General Data Protection Regulation (GDPR) is a prime example. In this book, we describe how federated machine learning addresses this problem with novel solutions combining distributed machine learning, cryptography and security, and incentive mechanism design based on economic principles and game theory. We explain different types of privacy-preserving machine learning solutions and their technological backgrounds, and highlight some representative practical use cases. We show how federated learning can become the foundation of next-generation machine learning that caters to technological and societal needs for responsible AI development and application.

KEYWORDS

federated learning, secure multi-party computation, privacy preserving machine learning, machine learning algorithms, transfer learning, artificial intelligence, data confidentiality, GDPR, privacy regulations

Contents

Preface

This book is about how to build and use machine learning (ML) models in artificial intelligence (AI) applications when the data are scattered across different sites, owned by different individuals or organizations, and there is no easy solution to bring the data together. Nowadays, we often hear that we are in the era of big data, and big data is an important ingredient that fuels AI advances in today's society. However, the truth is that we are in an era of small, isolated, and fragmented data silos. Data are collected and located at edge devices such as mobile phones. Organizations such as hospitals often have limited views on users' data due to their specialties. However, privacy and security requirements make it increasingly infeasible to merge the data at different organizations in a simple way. In such a context, federated machine learning (or federated learning, in short) emerges as a functional solution that can help build high-performance models shared among multiple parties while still complying with requirements for user privacy and data confidentiality.

Besides privacy and security concerns, another strong motivation for federated learning is to maximally use the computing power at the edge devices of a cloud system, where the communication is most efficient when only the computed results, rather than raw data, are transmitted between devices and servers. For example, autonomous cars can handle most computation locally and exchange the required results with the cloud at intervals. Satellites can finish most of the computation for information that they are to gather and communicate with the earth-based computers using minimal communication channels. Federated learning allows synchronization of computation between multiple devices and computing servers by exchanging only computed results.

We can explain federated learning with an analogy. That is, an ML model is like a sheep and the data is the grass. A traditional way to rear sheep is by buying the grass and transferring it to where the sheep is located, much like when we buy the datasets and move them to a central server. However, privacy concerns and regulations prevent us from physically moving the data. In our analogy, the grass can no longer travel outside its local area. Instead, federated learning employs a dual methodology. We can let the sheep graze multiple grasslands, much like our ML model that is built in a distributed manner without the data traveling outside its local area. In the end, the ML model grows from everyone's data, just like the sheep feed on everyone's grass.

Today, our modern society demands more responsible use of AI, and user privacy and data confidentiality are important properties of AI systems. In this direction, federated learning is already making significant positive impact, ranging from securely updating user models on mobile phones to improving medical imaging performance with multiple hospitals. Many existing works in different computer science areas have laid the foundation for the technology, such as distributed optimization and learning, homomorphic encryption, differential privacy, and secure multi-party computation.

There are two types of federated learning, horizontal and vertical. The Google GBoard system adopts horizontal federated learning and shows an example of B2C (business-to-consumer) applications. It can also be used to support edge computing, where the devices at the edge of a cloud system can handle many of the computing tasks and thus reduce the need to communicate via raw data with the central servers. Vertical federated learning, proposed and advanced by WeBank, represents the B2B (business-to-business) model, where multiple organizations join an alliance in building and using a shared ML model. The model is built while ensuring that no local data leaves any sites and maintaining the model performance according to business requirements. In this book, we cover both the B2C and B2B models.

To develop a federated learning system, multiple disciplines are needed, including ML algorithms, distributed machine learning (DML), cryptography and security, privacy-preserving data mining, game theory and economic principles, incentive mechanism design, laws and regulatory requirements, etc. It is a daunting task for someone to be well-versed in so many diverse disciplines, and the only sources for studying this field are currently scattered across many research papers and blogs. Therefore, there is a strong need for a comprehensive introduction to this subject in a single text, which this book offers.

This book is an introduction to federated learning and can serve as one's first entrance into this subject area. It is written for students in computer science, AI, and ML, as well as for big data and AI application developers. Students at senior undergraduate or graduate levels, faculty members, and researchers at universities and research institutions can find the book useful. Lawmakers, policy regulators, and government service departments can also consider it as a reference book on legal matters involving big data and AI. In classrooms, it can serve as a textbook for a graduate seminar course or as a reference book on federated learning literature.

The idea of this book came about in our development of a federated learning platform at WeBank known as Federated AI Technology Enabler (FATE), which became the world's first open-source federated learning platform and is now part of the Linux Foundation. WeBank is a digital bank that serves hundreds of millions of people in China. This digital bank has a business alliance across diverse backgrounds, including banking, insurance, Internet, and retail and supply-chain companies, just to name a few. We observe firsthand that data cannot be easily shared, but the need to collaborate to build new businesses supported by ML is very strong.

Federated learning was practiced by Google at large-scale in its mobile services for consumers as an example of B2C applications. We took one step further in expanding it to enable partnerships between multiple businesses in a partnership for B2B applications. The horizontal, vertical, and transfer learning-based federated learning categorization was first summarized in our survey paper published in *ACM Transactions on Intelligent Systems and Technology (ACM TIST)* [Yang et al., 2019] and was also presented at the 2019 AAAI Conference on Artificial Intelligence (organized by the Association for the Advancement of Artificial Intelligence) in Hawaii. Subsequently, various tutorials were given at conferences such as the 14th Chinese Computer Federation Technology Frontier in 2019. In the process of developing this book, our open-source federated learning system, FATE, was born and publicized [WeBank FATE, 2019] (see `https://www.fedai.org`), and the first international standard on federated learning via IEEE is being developed [IEEE P3652.1, 2019]. The tutorial notes and related research papers served as the basis for this book.

Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu
November 2019, Shenzhen, China

Acknowledgments

The writing of this book involved huge efforts from a group of very dedicated contributors. Besides the authors, different chapters were contributed by Ph.D. students, researchers, and research partners at various stages. We express our heartfelt gratitude to the following people who have made contributions toward the writing and editing of this book.

- Dashan Gao helped with writing Chapters 2 and 3.
- Xueyang Wu helped with writing Chapters 3 and 5.
- Xinle Liang helped with writing Chapters 3 and 9.
- Yunfeng Huang helped with writing Chapters 5 and 8.
- Sheng Wan helped with writing Chapters 6 and 8.
- Xiguang Wei helped with writing Chapter 9.
- Pengwei Xing helped with writing Chapters 8 and 10.

Finally, we thank our family for their understanding and continued support. Without them, the book would not have been possible.

Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu
November 2019, Shenzhen, China

CHAPTER 1

Introduction

1.1 MOTIVATION

We have witnessed the rapid growth of machine learning (ML) technologies in empowering diverse artificial intelligence (AI) applications, such as computer vision, automatic speech recognition, natural language processing, and recommender systems [Pouyanfar et al., 2019, Hatcher and Yu, 2018, Goodfellow et al., 2016]. The success of these ML technologies, in particular deep learning (DL), has been fueled by the availability of vast amounts of data (a.k.a. the big data) [Trask, 2019, Pouyanfar et al., 2019, Hatcher and Yu, 2018]. Using these data, DL systems can perform a variety of tasks that can sometimes exceed human performance; for example, DL empowered face-recognition systems can achieve commercially acceptable levels of performance given millions of training images. These systems typically require a huge amount of data to reach a satisfying level of performance. For example, the object detection system from Facebook has been reported to be trained on 3.5 billion images from Instagram [Hartmann, 2019].

In general, the big data required to empower AI applications is often large in size. However, in many application domains, people have found that big data are hard to come by. What we have most of the time are "small data," where either the data are of small sizes only, or they lack certain important information, such as missing values or missing labels. To provide sufficient labels for data often requires much effort from domain experts. For example, in medical image analysis, doctors are often employed to provide diagnosis based on scan images of patient organs, which is tedious and time consuming. As a result, high-quality and large-volume training data often cannot be obtained. Instead, we face silos of data that cannot be easily bridged.

The modern society is increasingly made aware of issues regarding the data ownership: who has the right to use the data for building AI technologies? In an AI-driven product recommendation service, the service owner claims ownership over the data about the products and purchase transactions, but the ownership over the data about user purchasing behaviors and payment habits is unclear. Since data are generated and owned by different parties and organizations, a traditional and naive approach is to collect and transfer the data to one central location where powerful computers can train and build ML models. Today, this methodology is no longer valid.

While AI is spreading into ever-widening application sectors, concerns regarding user privacy and data confidentiality expand. Users are increasingly concerned that their private information is being used (or even abused) by commercial and political purposes without their permission. Recently, several large Internet corporations have been fined heavily due to their leakage of users' private data to commercial companies. Spammers and under-the-table data exchanges are often punished in court cases.

In the legal front, law makers and regulatory bodies are coming up with new laws ruling how data should be managed and used. One prominent example is the adoption of the General Data Protection Regulation (GDPR) by the European Union (EU) in 2018 [GDPR website, 2018]. In the U.S., the California Consumer Privacy Act (CCPA) will be enacted in 2020 in the state of California [DLA Piper, 2019]. China's Cyber Security Law and the General Provisions of Civil Law, implemented in 2017, also imposed strict controls on data collection and transactions. Appendix A provides more information about these new data protection laws and regulations.

Under this new legislative landscape, collecting and sharing data among different organizations is becoming increasingly difficult, if not outright impossible, as time goes by. In addition, the sensitive nature of certain data (e.g., financial transactions and medical records) prohibits free data circulation and forces the data to exist in isolated data silos maintained by the data owners [Yang et al., 2019]. Due to industry competition, user privacy, data security, and complicated administrative procedures, even data integration between different departments of the *same company faces heavy resistance*. The prohibitively high cost makes it almost impossible to integrate data scattered in different institutions [WeBank AI, 2019]. Now that the old privacy-intrusive way of collecting and sharing data is outlawed, data consolidation involving different data owners is extremely challenging going forward.

How to solve the problem of data fragmentation and isolation while complying with the new stricter privacy-protection laws is a major challenge for AI researchers and practitioners. Failure to adequately address this problem will likely lead to a new AI winter [Yang et al., 2019].

Another reason why the AI industry is facing a data plight is that the benefit of collaborating over the sharing of the big data is not clear. Suppose that two organizations wish to collaborate on medical data in order to train a joint ML model. The traditional method of transferring the data from one organization to another will often mean that the original data owner will lose control over the data that they owned in the first place. The value of the data decreases as soon as the data leaves the door. Furthermore, when the better model as a result of integrating the data sources gained benefit, it is not clear how the benefit is fairly distributed among the participants. This fear of losing control and lack of transparency in determining the distribution of values is causing the so-called data fragmentation to intensify.

With edge computing over the Internet of Things, the big data is often not a single monolithic entity but rather distributed among many parties. For example, satellites taking images of the Earth cannot expect to transmit all data to data centers on the ground, as the amount of transmission required will be too large. Likewise, with autonomous cars, each car must be able to process much information locally with ML models while collaborate globally with other cars and computing centers. How to enable the updating and sharing of models among the multiple sites in a secure and yet efficient way is a new challenge to the current computing methodologies.

1.2 FEDERATED LEARNING AS A SOLUTION

As mentioned previously, multiple reasons make the problem of data silos become impediment to the big data needed to train ML models. It is thus natural to seek solutions to build ML models that do not rely on collecting all data to a centralized storage where model training can happen. An idea is to train a model at each location where a data source resides, and then let the sites communicate their respective models in order to reach a consensus for a global model. In order to ensure user privacy and data confidentiality, the communication process is carefully engineered so that no site can second-guess the private data of any other sites. At the same time, the model is built as if the data sources were combined. This is the idea behind "federated machine learning" or "federated learning" for short.

Federated learning was first practiced in an edge-server architecture by McMahan et al. in the context of updating language models on mobile phones [McMahan et al., 2016a,b, Konecný et al., 2016a,b]. There are many mobile edge devices each holding private data. To update the prediction models in the Gboard system, which is the Google's keyboard system for auto-completion of words, researchers at Google developed a federated learning system to update a collective model periodically. Users of the Gboard system gets a suggested query and whether the users clicked the suggested words. The word-prediction model in Gboard keeps improving based on not just a single mobile phone's accumulated data but all phones via a technique known as federated averaging (`FedAvg`). Federated averaging does not require moving data from any edge device to one central location. Instead, with federated learning, the model on each mobile device, which can be a smartphones or a tablet, gets encrypted and shipped to the cloud. All encrypted models are integrated into a global model under encryption, so that the server at the cloud does not know the data on each device [Yang et al., 2019, McMahan et al., 2016a,b, Konecný et al., 2016a,b, Hartmann, 2018, Liu et al., 2019]. The updated model, which is under encryption, is then downloaded to all individual devices on the edge of the cloud system [Konecný et al., 2016b, Hartmann, 2018, Yang et al., 2018, Hard et al., 2018]. In the process, users' individual data on each device is not revealed to others, nor to the servers in the cloud.

Google's federated learning system shows a good example of B2C (business-to-consumer), in designing a secure distributed learning environment for B2C applications. In the B2C setting, federated learning can ensure privacy protection as well as increased performance due to a speedup in transmitting the information between the edge devices and the central server.

Besides the B2C model, federated learning can also support the B2B (business-to-business) model. In federated learning, a fundamental change in algorithmic design methodology is, instead of transferring data from sites to sites, we transfer model parameters in a secure way, so that other parties cannot "second guess" the content of others' data. Below, we give a formal categorization of the federated learning in terms of how the data is distributed among the different parties.

1.2.1 THE DEFINITION OF FEDERATED LEARNING

Federated learning aims to build a joint ML model based on the data located at multiple sites. There are two processes in federated learning: model training and model inference. In the process of model training, information can be exchanged between parties but not the data. The exchange does not reveal any protected private portions of the data at each site. The trained model can reside at one party or shared among multiple parties.

At inference time, the model is applied to a new data instance. For example, in a B2B setting, a federated medical-imaging system may receive a new patient who's diagnosis come from different hospitals. In this case, the parties collaborate in making a prediction. Finally, there should be a fair value-distribution mechanism to share the profit gained by the collaborative model. Mechanism design should done in such a way to make the federation sustainable.

In broad terms, federated learning is an algorithmic framework for building ML models that can be characterized by the following features, where a model is a function mapping a data instance at some party to an outcome.

- There are two or more parties interested in jointly building an ML model. Each party holds some data that it wishes to contribute to training the model.

- In the model-training process, the data held by each party does not leave that party.

- The model can be transferred in part from one party to another under an encryption scheme, such that other parties cannot re-engineer the data at any given party.

- The performance of the resulting model is a good approximation of ideal model built with all data transferred to a single party.

More formally, consider N data owners $\{\mathcal{F}_i\}_{i=1}^N$ who wish to train a ML model by using their respective datasets $\{\mathcal{D}_i\}_{i=1}^N$. A conventional approach is to collect all data $\{\mathcal{D}_i\}_{i=1}^N$ together at one data server and train a ML model $\mathcal{M}_{SUM}$ on the server using the centralized dataset. In the conventional approach, any data owner $\{\mathcal{F}_i$ will expose its data $\{\mathcal{D}_i$ to the server and even other data owners.

Federated learning is a ML process in which the data owners collaboratively train a model $\mathcal{M}_{FED}$ without collecting all data $\{\mathcal{D}_i\}_{i=1}^N$. Denote $\mathcal{V}_{SUM}$ and $\mathcal{V}_{FED}$ as the performance measure (e.g., accuracy, recall, and F1-score) of the centralized model $\mathcal{M}_{SUM}$ and the federated model $\mathcal{M}_{FED}$, respectively.

We can capture what we mean by performance guarantee more precisely. Let δ be a non-negative real number. We say that the federated learning model $\mathcal{M}_{FED}$ has δ-performance loss if

$$|\mathcal{V}_{SUM} - \mathcal{V}_{FED}| < \delta. \tag{1.1}$$

The previous equation expresses the following intuition: if we use secure federated learning to build a ML model on distributed data sources, this model's performance on future data is approximately the same as the model that is built on joining all data sources together.

We allow the federated learning system to perform a little less than a joint model because in federated learning data owners do not expose their data to a central server or any other data owners. This additional security and privacy guarantee can be worth a lot more than the loss in accuracy, which is the δ value.

A federated learning system may or may not involve a central coordinating computer depending on the application. An example involving a coordinator in a federated learning architecture is shown in Figure 1.1. In this setting, the coordinator is a central aggregation server (a.k.a. the parameter server), which sends an initial model to the local data owners A–C (a.k.a. clients or participants). The local data owners A–C each train a model using their respective dataset, and send the model weight updates to the aggregation server. The aggregation sever then combines the model updates received from the data owners (e.g., using federated averaging [McMahan et al., 2016a]), and sends the combined model updates back to the local data owners. This procedure is repeated until the model converges or until the maximum number of iterations is reached. Under this architecture, the raw data of the local data owners never leaves the local data owners. This approach not only ensures user privacy and data security, but also saves communication overhead needed to send raw data. The communication between the central aggregation server and the local data owners can be encrypted (e.g., using homomorphic encryption [Yang et al., 2019, Liu et al., 2019]) to guard against information leakage.

The federated learning architecture can also be designed in a peer to peer manner, which does not require a coordinator. This ensures further security guarantee in which the parties communicate directly without the help of a third party, as illustrated in Figure 1.2. The advantage of this architecture is increased security, but a drawback is potentially more computation to encrypt and decrypt messages.

Federated learning brings several benefits. It preserves user privacy and data security by design since no data transfer is required. Federated learning also enables several parties to collaboratively train a ML model so that each of the parties can enjoy a better model than what it can achieve alone. For example, federated learning can be used by private commercial banks to detect multi-party borrowing, which has always been a headache in the banking industry, especially in the Internet finance industry [WeBank AI, 2019]. With federated learning, there is no need to establish a central database, and any financial institution participating in federated learning can initiate new user queries to other agencies within the federation. Other agencies only need to answer questions about local lending without knowing specific information of the user. This not only protects user privacy and data integrity, but also achieves an important business objective of identifying multi-party lending.

While federated learning has great potential, it also faces several challenges. The communication link between the local data owner and the aggregation server may be slow and un-

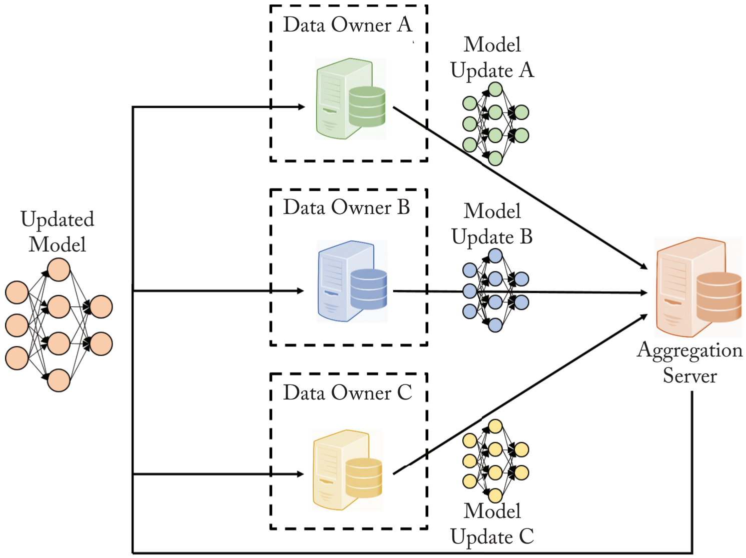


Figure 1.1: An example federated learning architecture: client-server model.

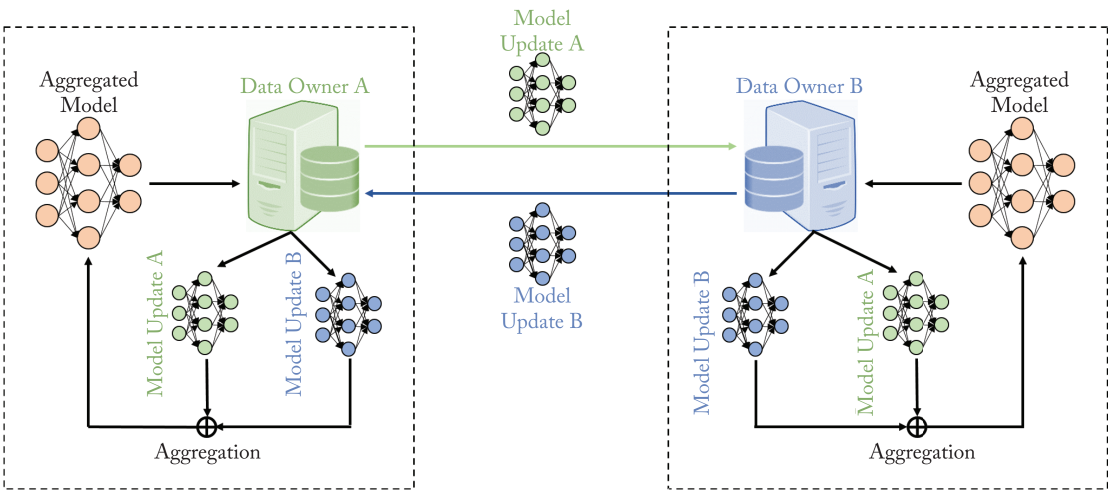


Figure 1.2: An example federated learning architecture: peer-to-peer model.

stable [Hartmann, 2018]. There may be a very large number of local data owners (e.g., mobile users). In theory, every mobile user can participate in federated learning, making the system unstable and unpredictable. Data from different participants in federated learning may follow non-identical distributions [Zhao et al., 2018, Sattler et al., 2019, van Lier, 2018], and different participants may have unbalanced numbers of data samples, which may result in a biased model or even failure of training a model. As the participants are distributed and difficult to authenticate, federated learning model poisoning attacks [Bhagoji et al., 2019, Han, 2019], in which one or more malicious participants send ruinous model updates to make the federated model useless, can take place and confound the whole operation.

1.2.2 CATEGORIES OF FEDERATED LEARNING

Let matrix $\mathcal{D}_i$ denote the data held by the i th data owner. Suppose that each row of the matrix $\mathcal{D}_i$ represents a data sample, and each column represents a specific feature. At the same time, some datasets may also contain label data. We denote the feature space as $\mathcal{X}$, the label space as $\mathcal{Y}$, and we use $\mathcal{I}$ to denote the sample ID space. For example, in the financial field, labels may be users' credit. In the marketing field labels may be the user's purchasing desire. In the education field, $\mathcal{Y}$ may be the students' scores. The feature $\mathcal{X}$, label $\mathcal{Y}$, and sample IDs $\mathcal{I}$ constitute the complete training dataset $(\mathcal{I}, \mathcal{X}, \mathcal{Y})$. The feature and sample spaces of the datasets of the participants may not be identical. We classify federated learning into horizontal federated learning (HFL), vertical federated learning (VFL), and federated transfer learning (FTL), according to how data is partitioned among various parties in the feature and sample spaces. Figures 1.3–1.5 show the three federated learning categories for a two-party scenario [Yang et al., 2019].

HFL refers to the case where the participants in federated learning share overlapping data features, i.e., the data features are aligned across the participants, but they differ in data samples. It resembles the situation that the data is horizontally partitioned inside a tabular view. Hence, we also call HFL as sample-partitioned federated learning, or example-partitioned federated learning [Kairouz et al., 2019]. Different from HFL, VFL applies to the scenario where the participants in federated learning share overlapping data samples, i.e., the data samples are aligned amongst the participants, but they differ in data features. It resembles the situation that data is vertically partitioned inside a tabular view. Thus, we also name VFL as feature-partitioned federated learning. FTL is applicable for the case when there is neither overlapping in data samples nor in features.

For example, when the two parties are two banks that serve two different regional markets, they may share only a handful of users but their data may have very similar feature spaces due to similar business models. That is, with limited overlap in users but large overlap in data features, the two banks can collaborate in building ML models through horizontal federated learning [Yang et al., 2019, Liu et al., 2019].

When two parties providing different services but sharing a large amount of users (e.g., a bank and an e-commerce company), they can collaborate on the different feature spaces that they own, leading to a better ML model for both. That is, with large overlap in users but little overlap in data features, the two companies can collaborate in building ML models through vertical federated learning [Yang et al., 2019, Liu et al., 2019]. Split learning, recently proposed by Gupta and Raskar [2018] and Vepakomma et al. [2019, 2018], is regarded here as a special case of vertical federated learning, which enables vertically federated training of deep neural

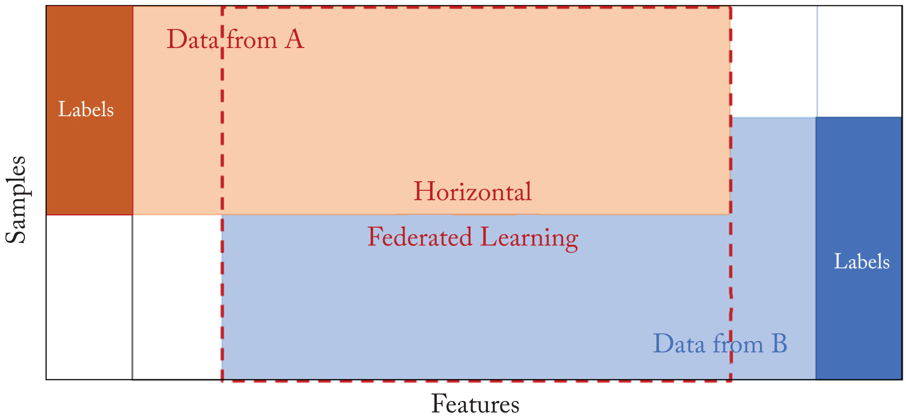


Figure 1.3: Illustration of HFL, a.k.a. sample-partitioned federated learning where the overlapping features from data samples held by different participants are taken to jointly train a model [Yang et al., 2019].

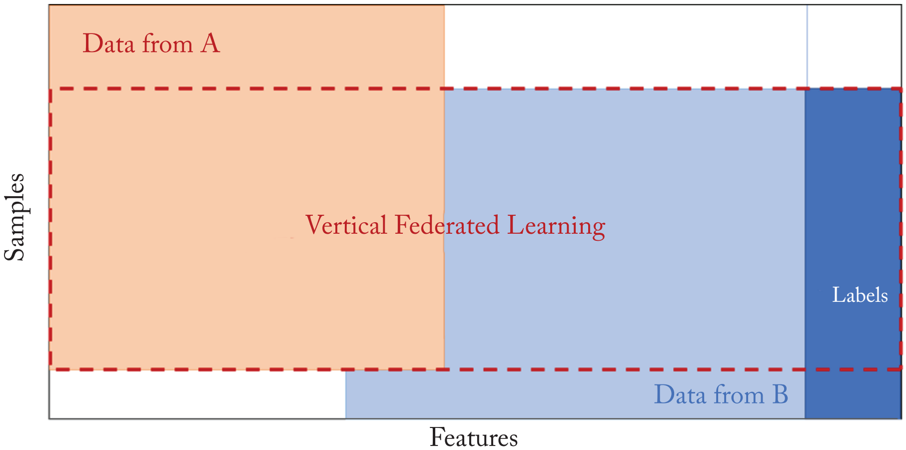


Figure 1.4: Illustration of VFL, a.k.a feature-partitioned federated learning where the overlapping data samples that have non-overlapping or partially overlapping features held by multiple participants are taken to jointly train a model [Yang et al., 2019].

networks (DNNs). That is, split learning facilitates training DNNs in federated learning settings over vertically partitioned data [Vepakomma et al., 2019].

In scenarios where participating parties have highly heterogeneous data (e.g., distribution mismatch, domain shift, limited overlapping samples, and scarce labels), HFL and VFL may not be able to build effective ML models. In those scenarios, we can leverage transfer learning

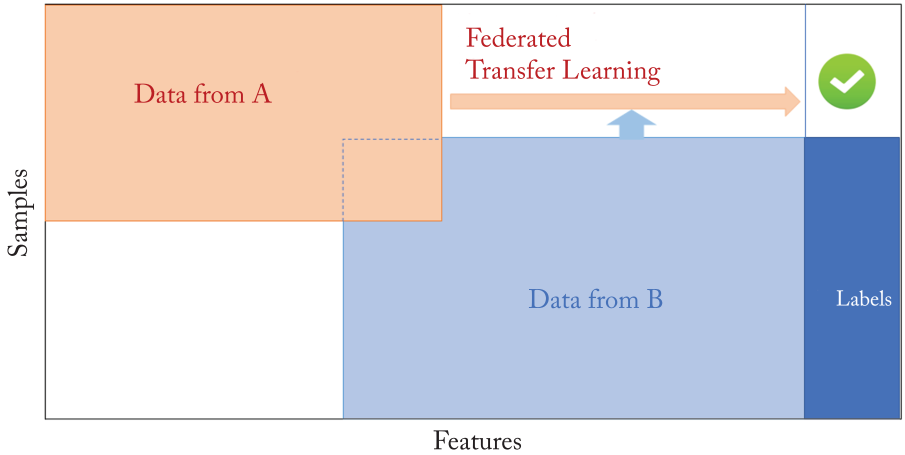


Figure 1.5: Federated transfer learning (FTL) [Yang et al., 2019]. A predictive model learned from feature representations of aligned samples belonging to party A and party B is utilized to predict labels for unlabeled samples of party A.

techniques to bridge the gap between heterogeneous data owned by different parties. We refer to federated learning leveraging transfer learning techniques as FTL.

Transfer learning aims to build effective ML models in a resource-scarce target domain by exploiting or transferring knowledge learned from a resource-rich source domain, which naturally fits the federated learning setting where parties are typically from different domains. Pan and Yang [2010] divides transfer learning into mainly three categories: (i) instance-based transfer, (ii) feature-based transfer, and (iii) model-based transfer. Here, we provide brief descriptions on how these three categories of transfer learning techniques can be applied to federated settings.

- **Instance-based FTL.** Participating parties selectively pick or re-weight their training data samples such that the distance among domain distributions can be minimized, thereby minimizing the objective loss function.

- **Feature-based FTL.** Participating parties collaboratively learn a common feature representation space, in which the distribution and semantic difference among feature representations transformed from raw data can be relieved and such that knowledge can be transferable across different domains to build more robust and accurate shared ML models.

  Figure 1.5 illustrates an FTL scenario where a predictive model learned from feature representations of aligned samples belonging to party A and party B is utilized to predict labels for unlabeled samples of party A. We will elaborate on how this FTL is performed in Chapter 6.

- **Model-based FTL.** Participating parties collaboratively learn shared models that can benefit for transfer learning. Alternatively, participating parties can utilize pre-trained models as the whole or part of the initial models for a federated learning task.

We will further explain in detail the HFL and VFL in Chapter 4 and Chapter 5, respectively. In Chapter 6, we will elaborate on a feature-based FTL framework proposed by Liu et al. [2019].

1.3 CURRENT DEVELOPMENT IN FEDERATED LEARNING

The idea of federated learning has appeared in different forms throughout the history of computer science, such as privacy-preserving ML [Fang and Yang, 2008, Mohassel and Zhang, 2017, Vaidya and Clifton, 2004, Xu et al., 2015], privacy-preserving DL [Liu et al., 2016, Phong, 2017, Phong et al., 2018], collaborative ML [Melis et al., 2018], collaborative DL [Zhang et al., 2018, Hitaj et al., 2017], distributed ML [Li et al., 2014, Wang, 2016], distributed DL [Vepakomma et al., 2018, Dean et al., 2012, Ben-Nun and Hoefler, 2018], and federated optimization [Li et al., 2019, Xie et al., 2019], as well as privacy-preserving data analytics [Mangasarian et al., 2008, Mendes and Vilela, 2017, Wild and Mangasarian, 2007, Bogdanov et al., 2014]. Chapters 2 and 3 will present some examples.

1.3.1 RESEARCH ISSUES IN FEDERATED LEARNING

Federated learning was studied by Google in a research paper published in 2016 on arXiv.[1] Since then, it has been an area of active research in the AI community as evidenced by the fast-growing volume of preprints appearing on arXiv. Yang et al. [2019] provide a comprehensive survey of recent advances of federated learning.

Recent research work on federated learning are mainly focused on improving security and statistical challenges [Yang et al., 2019, Mancuso et al., 2019]. Cheng et al. [2019] proposed SecureBoost in the setting of vertical federated learning, which is a novel lossless privacy-preserving tree-boosting system. SecureBoost provides the same level of accuracy as the non-privacy-preserving approach. It is theoretically proven that the SecureBoost framework is as accurate as other non-federated gradient tree-boosting algorithms that rely on centralized datasets [Cheng et al., 2019].

Liu et al. [2019] presents a flexible federated transfer learning framework that can be effectively adapted to various secure multi-party ML tasks. In this framework, the federation allows knowledge to be shared without compromising user privacy, and enables complimentary knowledge to be transferred in the network via transfer learning. As a result, a target-domain party can build more flexible and powerful models by leveraging rich labels from a source-domain party.

[1] arXiv is a repository of electronic preprints (e-prints) hosted by Cornell University. For more information, visit arXiv website `https://arxiv.org/`.

In a federated learning system, we can assume that participating parties are honest, semi-honest, or malicious. When a party is malicious, it is possible for a model to taint its data in training. The possibility of model poisoning attacks on federated learning initiated by a single non-colluding malicious agent is discussed in Bhagoji et al. [2019]. A number of strategies to carry out model poisoning attack were investigated. It was shown that even a highly constrained adversary can carry out model poisoning attacks while simultaneously maintaining stealth. The work of Bhagoji et al. [2019] reveals the vulnerability of the federated learning settings and advocates the need to develop effective defense strategies.

Re-examining the existing ML models under the federated learning settings has become a new research direction. For example, combining federated learning with reinforcement learning has been studied in Zhuo et al. [2019], where Gaussian differentials on the information shared among agents when updating their local models were applied to protect the privacy of data and models. It has been shown that the proposed federated reinforcement learning model performs close to the baselines that directly take all joint information as input [Zhuo et al., 2019].

Another study in Smith et al. [2017] showed that multi-task learning is naturally suited to handle the statistical challenges of federated learning, where separate but related models are learned simultaneously at each node. The practical issues, such as communication cost, stragglers, and fault tolerance in distributed multi-task learning and federated learning, were considered. A novel systems-aware optimization method was put forward, which achieves significant improved efficiency compared to the alternatives.

Federated learning has also been applied in the fields of computer vision (CV), e.g., medical image analysis [Sheller et al., 2018, Liu et al., 2018, Huang and Liu, 2019], natural language processing (NLP) (see, e.g., Chen et al. [2019]), and recommender systems (RS) (see, e.g., Ammad-ud-din et al. [2019]). This will be further reviewed in Chapter 8.

Regarding applications of federated learning, the researchers at Google have applied federated learning in mobile keyboard prediction [Bonawitz and Eichner et al., 2019, Yang et al., 2018, Hard et al., 2018], which has achieved significant improvement in prediction accuracy without exposing mobile user data. Researchers at Firefox have used federated learning for search word prediction [Hartmann, 2018]. There is also new research effort to make federated learning more personalizable [Smith et al., 2017, Chen et al., 2018].

1.3.2 OPEN-SOURCE PROJECTS

Interest in federated learning is not only limited to theoretical work. Research on the development and deployment of federated learning algorithms and systems is also flourishing. There are several fast-growing open-source projects of federated learning.

• **Federated AI Technology Enabler (FATE)** [WeBank FATE, 2019] is an open-source project initiated by the AI department of WeBank[2] to provide a secure computing framework to support the federated AI ecosystem [WeBank FedAI, 2019]. It implements secure computation protocols based on homomorphic encryption (HE) and secure multi-party computation (MPC). It supports a range of federated learning architectures and secure computation algorithms, including logistic regression, tree-based algorithms, DL (artificial neural networks), and transfer learning. For more information on FATE, readers can refer to the GitHub FATE website [WeBank FATE, 2019] and the FedAI website [WeBank FedAI, 2019].

• **TensorFlow[3] Federated** project [Han, 2019, TFF, 2019, Ingerman and Ostrowski, 2019, Tensorflow-federated, 2019] (TFF) is an open-source framework for experimenting with federated ML and other computations on decentralized datasets. TFF enables developers to simulate existing federated learning algorithms on their models and data, as well as to experiment with novel algorithms. The building blocks provided by TFF can also be used to implement non-learning computations, such as aggregated analytics over decentralized data. The interfaces of TFF are organized in two layers: (1) the federated learning (FL) application programming interface (API) and (2) federated Core (FC) API. TFF enables developers to declaratively express federated computations, so that they can be deployed in diverse runtime environments. Included in TFF is a single-machine simulation run-time for experimentation.

• **TensorFlow-Encrypted** [TensorFlow-encrypted, 2019] is a Python library built on top of TensorFlow for researchers and practitioners to experiment with privacy-preserving ML. It provides an interface similar to that of TensorFlow, and aims to make the technology readily available without requiring user to be experts in ML, cryptography, distributed systems, and high-performance computing.

• **coMind** [coMind.org, 2018, coMindOrg, 2019] is an open-source project for training privacy-preserving federated DL models. The key component of coMind is the implementation of the federated averaging algorithm [McMahan et al., 2016a, Yu et al., 2018], which is training ML models in a collaborative way while preserving user privacy and data security. coMind is built on top of TensorFlow and provides high-level APIs for implementing federated learning.

• **Horovod** [Sergeev and Balso, 2018, Horovod, 2019], developed by Uber, is an open-source distributed training framework for DL. It is based on the open message passing interface

[2]WeBank, opened in December 2014 upon receiving its banking license in China. It is the first digital-only bank in China. WeBank is devoted to offering individuals and SMEs under-served by the current banking system with a variety of convenient and high-quality financial services. For more information on WeBank, please visit `https://www.webank.com/en/`.

[3]TensorFlow is an open-source DL framework, developed and maintained by Google Inc. TensorFlow is widely used in research and implementation of DL. For more information on TensorFlow, readers can refer to its project website `https://www.tensorflow.org/` and its GitHub website `https://github.com/tensorflow`.

(MPI) and works on top of popular DL frameworks, such as TensorFlow and PyTorch.[4] The goal of Horovod is to make distributed DL fast and easy to use. Horovod supports federated learning via open MPI and currently, encryption is not yet supported.

• **OpenMined/PySyft** [Han, 2019, OpenMined, 2019, Ryffel et al., 2018, PySyft, 2019, Ryffel, 2019] provides two methods for privacy preservation: (1) federated learning and (2) differential privacy. OpenMined further supports two methods of secure computation through multi-party computation and homomorphic encryption. OpenMined has made available the PySyft library [PySyft, 2019], which is the first open-source federated learning framework for building secure and scalable ML models [Ryffel, 2019]. PySyft is simply a hooked extension of PyTorch. For users who are familiar with PyTorch, it is very easy to implement federated learning systems with PySyft. Federated learning extension based on the TensorFlow framework is currently being developed within OpenMined.

• **LEAF Beanchmark** [LEAF, 2019, Caldas et al., 2019], maintained by Carnegie Mellon University and Google AI, is a modular benchmarking framework for ML in federated settings, with applications in federated learning, multi-task learning, meta-learning, and on-device learning. LEAF includes a suite of open-source federated datasets (e.g., FEMNIST, Sentiment140, and Shakespeare), a rigorous evaluation framework, and a set of reference implementations, aiming to capture the reality, obstacles, and intricacies of practical federated learning environments. LEAF enables researchers and practitioners in these domains to investigate new proposed solutions under more realistic assumptions and settings. LEAF will include additional tasks and datasets in its future releases.

1.3.3 STANDARDIZATION EFFORTS

As more developments are made in the legal front on the secure and responsible use of users' data, technical standard needs to be developed to ensure that organizations use the same language and follow a standard guideline in developing future federated learning systems. Moreover, there is increasing need for the technical community to communicate with the regulatory and legal communities over the use of the technology. As a result, it is important to develop international standards that can be adopted by multiple disciplines.

For example, companies striving to satisfy the GDPR requirements need to know what technical developments are needed in order to satisfy the legal requirements. Standards can provide a bridge between regulators and technical developers.

One of the early standards is initiated by the AI Department at WeBank with the Institute of Electrical and Electronics Engineers (IEEE) P3652.1 Federated Machine Learning Working Group (known as Federated Machine Learning (C/LT/FML)) was established in December

[4] PyTorch is a popular DL framework and is widely used in research and implementation. For more information, visit the official PyTorch website `https://pytorch.org/` and the GitHub PyTorch website `https://github.com/pytorch/pytorch`.

2018 [IEEE P3652.1, 2019]. The objective of this working group is to provide guidelines for building the architectural framework and applications of federated ML. The working group will define the architectural framework and application guidelines for federated ML, including:

1. The description and definition of federated learning;
2. The types of federated learning and the application scenarios to which each type applies;
3. Performance evaluation of federated learning; and
4. The associated regulatory requirements.

The purpose of this standard is to provide a feasible solution for the industrial application of AI without exchanging data directly. This standard is expected to promote and facilitate collaborations in an environment where privacy and data protection issues have become increasingly important. It will promote and enable to the use of distributed data sources for the purpose of developing AI without violating regulations or ethical concerns.

1.3.4 THE FEDERATED AI ECOSYSTEM

The Federated AI (FedAI) ecosystem project was initiated by the AI Department of WeBank [WeBank FedAI, 2019]. The primary goal of the project is to develop and promote advanced AI technologies that preserve user privacy, data security, and data confidentiality. The federated AI ecosystem features four main themes.

- Open-source technologies: FedAI aims to accelerate open-source development of federated ML and its applications. The FATE project [WeBank FATE, 2019] is a flagship project under FedAI.
- Standards and guidelines: FedAI, together with partners, are drawing up standardization to formulate the architectural framework and application guidelines of federated learning, and facilitate industry collaboration. One representative work is the IEEE P3652.1 federated ML working group [IEEE P3652.1, 2019].
- Multi-party consensus mechanisms: FedAI is studying incentive and reward mechanisms to encourage more institutions to participate in federated learning research and development in a sustainable way. For example, FedAI is undertaking work to establish a multiparty consensus mechanism based on technologies like blockchain.
- Applications in various verticals: To open up the potential of federated learning, FedAI endeavors to showcase more vertical field applications and scenarios, and to build new business models.

1.4 ORGANIZATION OF THIS BOOK

The organization of this book is as follows. Chapter 2 provides background information on privacy-preserving ML, covering well-known techniques for data security. Chapter 3 describes distributed ML, highlighting the difference between federated learning and distributed ML. Horizontal federated learning, vertical federated learning, and federated transfer learning are elaborated in detail in Chapter 4, Chapter 5, and Chapter 6, respectively. Incentive mechanism design for motivating the participation in federated learning is discussed in Chapter 7. Recent work on extending federated learning to the fields of computer vision, natural language processing, and recommender systems are reviewed in Chapter 8. Chapter 9 presents federated reinforcement learning. The prospect of applying federated learning into various industrial sectors is summarized in Chapter 10. Finally, we provide a summary of this book and looking ahead in Chapter 11. Appendix A provides an overview of recent data protection laws and regulations in the European Union, the United States, and China.

CHAPTER 2

Background

In this chapter, we introduce the background knowledge related to federated learning, covering privacy-preserving machine learning techniques and data analytics.

2.1 PRIVACY-PRESERVING MACHINE LEARNING

Data leakage and privacy violation incidents have brought about heightened public awareness of the need for AI systems to be able to preserve user privacy and data confidentiality. Researchers are interested in developing techniques for privacy-preserving properties to be built inside machine learning (ML) systems. The resulting systems are known as privacy-preserving machine learning systems (PPML). In fact, 2018 was considered a breakout year for PPML [Mancuso et al., 2019]. PPML is a broad term that generally refers to ML equipped with defense measures for protecting user privacy and data security. The system security and cryptography community has also proposed various secure frameworks for ML.

In Westin [1968], Westin defined information privacy as follows: "the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others." This essentially defines the right to control the access and handling of one's information. The main idea of information privacy is to have control over the collection and handling of one's personal data [Mendes and Vilela, 2017].

In this chapter, we will introduce several popular approaches used in PPML including secure multi-party computation (MPC), homomorphic encryption (HE) for privacy-preserving model training and inference, as well as differential privacy (DP) for preventing unwanted data disclosure. Privacy-preserving gradient descent methods will also be discussed.

2.2 PPML AND SECURE ML

Before going into the details of PPML, we first clarify the difference between PPML and secure ML. PPML and secure ML differ mainly in the types of security violations that they are designed to deal with [Barreno et al., 2006]. In secure ML, the adversary (i.e., attacker) is assumed to violate the *integrity and availability* of a data-analytic system, while in PPML, the adversary is assumed to violate the *privacy and confidentiality* of an ML system.

Most of the time, compromise in security is caused by the intentional attack by a third party. We are concerned with three major types of attacks in ML.

- **Integrity attack.** An attack on *integrity* may result in intrusion points being classified as normal (i.e., false negatives) by the ML system.
- **Availability attack.** An attack on *availability* may lead to classification errors (both false negatives and false positives) such that the ML system becomes unusable. This is a broader type of integrity attacks.
- **Confidentiality attack.** An attack on *confidentiality* may result in sensitive information (e.g., training data or model) of an ML system being leaked.

Table 2.1 gives a comparison between PPML and secure ML in terms of security violations, adversary attacks, and defense techniques.

Table 2.1: Comparison between PPML and secure ML

	Security Violations	Adversary Attacks	Defence Techniques
PPML	Privacy Confidentiality	Reconstruction attack Inversion attack Membership-inference attack	Secure multi-party computation Homomorphic encryption Differential privacy
Secure ML	Integrity Availability	Poisoning attack Adversarial attack Oracle attack	Defensive distillation Adversarial training Regularization

In this chapter, we mainly focus on PPML and defense techniques against privacy and confidentiality violations in ML. Interested readers can refer to Barreno et al. [2006] for a more detailed explanation of secure ML.

2.3 THREAT AND SECURITY MODELS

2.3.1 PRIVACY THREAT MODELS

In order to preserve privacy and confidentiality in ML, it is important to understand the possible threat models. In ML tasks, the participants usually take up three different roles: (1) as the input party, e.g., the data owner; (2) as the computation party (e.g., the model builder and inference service provider); and (3) as the result party (e.g., the model querier and user) [Bogdanov et al., 2014].

Attacks on ML may happen in any stage, including data publishing, model training, and model inference. *Attribute-inference attacks* can happen in the data publishing stage, where adversaries may attempt to de-anonymize or target data-record owners for malevolent purposes. The attacks during ML model training are called *reconstruction attacks*, where the computation party aims to reconstruct the raw data of the data providers or to learn more information about the data providers than what the model builders intend to reveal.

For federated learning, reconstruction attacks are the major privacy concerns. In the inference phase of ML models, an adversarial result party may conduct *reconstruction attack*, *model inversion attacks*, or *membership-inference attacks*, using reverse engineering techniques to gain extra information about the model or raw training data.

Reconstruction Attacks. In reconstruction attacks, the adversary's goal is to extract the training data or feature vectors of the training data during ML model training or model inference. In centralized learning, raw data from different data parties are uploaded to the computation party, which makes the data vulnerable to adversaries, such as a malicious computation party. Large companies may collect raw data from users to train an ML model. However, the collected data may be used for other purposes or sent to a third-party without informed consent from the users. In federated learning, each participating party carries out ML model training using their local data. Only the model weight updates or gradient information are shared with other parties. However, the gradient information may also be leveraged to reveal extra information about the training data [Aono et al., 2018]. Plain-text gradient updating may also violate privacy in some application scenarios. To resist reconstruction attacks, ML models that store explicit feature values such as support vector machine (SVM) and k-nearest neighbors (kNN) should be avoided. During model training, secure multi-party computation (MPC) [Yao, 1982] and homomorphic encryption (HE) [Rivest et al., 1978] can be used to defend against such attacks by keeping the intermediate values private. During model inference, the party computing the inference result should only be granted black-box access to the model. MPC and HE can be leveraged to protect the privacy of the user query during model inference. MPC, HE, and their corresponding applications in PPML will be introduced in Sections 2.4.1 and 2.4.2, respectively.

Model Inversion Attacks. In model inversion attacks, the adversary is assumed to have either white-box access or black-box access to the model. For the case of white-box access, the adversary knows the clear-text model without stored feature vectors. For the case of black-box access, the adversary can only query the model with data and collect the responses. The adversary's target is to extract the training data or feature vectors of the training data from the model. The black-box access adversary may also reconstruct the clear-text model from the response by conducting an *equation solving attack*. Theoretically, for an N-dimensional linear model, an adversary can steal it with $N+1$ queries. Such a problem can be formalized as solving θ from $(x, h_\theta(x))$. The adversary can also learn a similar model using the query-response pairs to simulate the original model. To resist model inversion attacks, less knowledge of the model should be exposed to the adversary. The access to model should be limited to black-box access, and the output should be limited as well. There are several strategies proposed to reduce the success rate of model inversion attack. Fredrikson et al. [2015] choose to report only rounded confidence values. Al-Rubaie and Chang [2016] take the predicted class labels as response, and the aggregated prediction results of multiple testing instances are returned to further enhance model protection. Bayesian neural networks combined with homomorphic encryption have been developed [Xie et al., 2019], to resist such attacks during secure neural network inference.

Membership-Inference Attacks. In membership-inference attacks, the adversary has black-box access to a model, as well as a certain sample, as its knowledge. The adversary's target is to learn if the sample is inside the training set of the model. The adversary infers whether a sample belongs to the training set or not based on the ML model output. The adversary conducts such attacks by finding and leveraging the differences in the model predictions on the samples belonging to the training set vs. other samples. Defense techniques that are proposed to resist model inversion attacks, such as result generalization by reporting rounded prediction results are shown to be effective to thwart such attacks [Shokri et al., 2017]. Differential privacy (DP) [Dwork et al., 2006] is a major approach to resist membership inference attacks, which will be introduced in Section 2.4.3.

Attribute-Inference Attacks. In attribute-inference attacks, the adversary tries to de-anonymize or target record owners for malevolent purpose. Anonymization by removing personally identifiable information (PII) (also known as sensitive features), such as user IDs and names, before data publishing appears to be a natural approach for protecting user privacy. However, it has been shown to be ineffective. For example, Netflix, the world's largest online movie rental service provider, released a movie rating dataset, which contains anonymous movie ratings from 500,000 subscribers. Despite anonymization, Narayanan and Shmatikov [2008] managed to leverage this dataset along with the Internet Movie Database (IMDB) as background knowledge to re-identify the Netflix users in the records, and further managed to deduce the user's apparent political preferences. This incident shows that anonymization fails in the face of strong adversaries with access to alternative background knowledge. To deal with attribute-inference attacks, group anonymization privacy approaches have been proposed in Mendes and Vilela [2017]. Privacy preservation in group anonymization privacy is achieved via generalization and suppression mechanisms.

Model Poisoning Attacks. It has been shown that federated learning may be vulnerable to model poisoning attacks [Bhagoji et al., 2019], also known as backdoor attacks [Bagdasaryan et al., 2019]. For example, a malicious participant in federated learning may inject a hidden backdoor functionality into the trained federated model, e.g., to cause a trained word predictor to complete certain sentences with an attacker-chosen word [Bagdasaryan et al., 2019]. Bhagoji et al. [2019] proposed a number of strategies to carry out model poisoning attacks, such as boosting of the malicious participant's model update, an alternating minimization strategy that alternately optimizes for the legit training loss and the adversarial backdoor objective, and using parameter estimation for the benign updates to improve attack success. Bagdasaryan et al. [2019] developed a new model-poisoning methodology using model replacement, where a constrainand-scale technique is proposed to evade anomaly detection-based defenses by incorporating the evasion into the attacker's loss function during model training. Possible solutions against model poisoning attacks include blockchain-based approaches [Preuveneers et al., 2018] and trusted execution environment (TEE) based approaches [Mo and Haddadi, 2019].

2.3.2 ADVERSARY AND SECURITY MODELS

For cryptographic PPML techniques, including MPC and HE, two types of adversaries are concerned in the literature.

- **Semi-honest adversaries.** In the semi-honest (a.k.a honest-but-curious, and passive) adversary model, the adversaries abide by the protocol honestly, but also attempt to learn more information beyond the output from the received information.

- **Malicious adversaries.** In the malicious (a.k.a. active) adversary model, the adversaries deviate from the protocol and can behave arbitrarily.

The semi-honest adversary model is widely considered in most PPML studies. The main reason is that, in federated learning, it is beneficial to each party to honestly follow the ML protocol, since malicious behaviors also break the benefits of the adversaries themselves. The other reason is that, in cryptography, it is a standard method to build a protocol secure against semi-honest adversaries first, then modify it to be secure against malicious adversaries via zero-knowledge proof.

For both security models, the adversaries corrupt a fraction of the parties, and the corrupted parties may collude with each other. The corruption of parties can be static or adaptive. The complexity of an adversary can be either polynomial-time or computational unbounded, corresponding to information-theoretic secure and computational secure, respectively. The security in cryptography is based on the notion of indistinguishability. Interested readers can refer to Lindell [2005] and Lindell and Pinkas [2009] for detailed analysis of adversary and security models.

2.4 PRIVACY PRESERVATION TECHNIQUES

In this section, we discuss privacy preservation techniques. We cover three types of such approaches, namely (1) MPC, (2) HE, and (3) DP.

2.4.1 SECURE MULTI-PARTY COMPUTATION

Secure Multi-Party Computation (MPC), a.k.a. secure function evaluation (SFE), was initially introduced as a secure two-party computation problem (the famous Millionaire's Problem), and generalized in 1986 by Andrew Yao [1986]. In MPC, the objective is to jointly compute a function from the private input by each party, without revealing such inputs to the other parties.

MPC tells us that for any functionality, it is possible to compute it without revealing anything other than the output.

Definition

MPC allows us to compute functions of private input values so that each party learns only the corresponding function output value, but not input values from other parties. For example, given a secret value x that is split into n shares so that a party P_i only knows x_i, all parties can collaboratively compute

$$y_1, \ldots, y_n = f(x_1, \ldots, x_n)$$

so that party P_i learns nothing beyond the output value y_i corresponding to its own input x_i.

The standard approach to prove that an MPC protocol is secure is the *simulation paradigm* [Lindell, 2017]. To prove an MPC protocol is secure against adversaries that corrupt t parties under the simulation paradigm, we build a simulator that, when given inputs and outputs of t colluding parties, generates t transcripts, so that the generated transcripts are *indistinguishable* to that generated in the actual protocol.

In general, MPC can be implemented through three different frameworks, namely: (1) Oblivious Transfer (OT) [Keller et al., 2016, Goldreich et al., 1987]; (2) Secret Sharing (SS) [Shamir, 1979, Rabin and Ben-Or, 1989]; and (3) Threshold Homomorphic Encryption (THE) [Cramer et al., 2001, Damgård and Nielsen, 2003]. From a certain point of view, both oblivious transfer protocols and threshold homomorphic encryption schemes use the idea of secret sharing. This might be the reason why secret sharing is widely regarded as the core of MPC. In the rest of this section, we will introduce oblivious transfer and secret sharing.

Oblivious Transfer

OT is a two-party computation protocol proposed by Rabin in 1981 [Rabin, 2005]. In OT, the sender owns a database of message-index pairs $(M_1, 1), \ldots, (M_N, N)$. At each transfer, the receiver chooses an index i for some $1 \leq i \leq N$, and receives M_i. The receiver does not learn any other information about the database, and the sender does not learn anything about the receiver's selection i. Here, we give the definition of 1-out-of-n OT.

Definition 2.1 1-out-of-n OT: Suppose Party A has a list $(x_1, \ldots, x_n)$ as the input, Party B has $i \in 1, \ldots, n$ as the input. 1-out-of-n OT is an MPC protocol where A learns nothing about i and B learns nothing else but x_i.

When $n = 2$, we get 1-out-of-2 OT which has the following property: 1-out-of-2 OT is universal for two-party MPC [Ishai et al., 2008]. That is, given a 1-out-of-2 OT protocol, one can conduct any secure two-party computation.

Many Constructions of OT has been proposed such as Bellare–Micali's [Bellare and Micali, 1990], Naor–Pinka's [Naor and Pinkas, 2001], and Hazay–Lindell's [Hazay and Lindell, 2010] approaches. Here, we demonstrate the Bellare-Micali's construction of OT, which utilizes Diffie–Hellman key exchange and is based on the computational Diffie–Hellman (CDH) assumption [Diffie and Hellman, 1976]. The Bellare–Micali's construction works as follows: the receiver sends two public keys to the sender. The receiver only holds one private key corresponding to one of the two public keys, and the sender does not know which public key it is. Then, the sender encrypts the two massages with their corresponding public keys, and sends the ciphertexts to the receiver. Finally, the receiver decrypts the target ciphertext with the private key.

Bellare–Micali Construction. In a discrete logarithm setting $(\mathbb{G}, g, p)$, where $\mathbb{G}$ is a group of prime order p, $g \in \mathbb{G}$ is a generator, and $H : G \rightarrow \{0,1\}^n$ is a hash function. Suppose the sender A has $x_0, x_1 \in \{0,1\}^n$, and the receiver B has $b \in \{0,1\}$.

1. A chooses a random element $c \leftarrow G$ and sends it to B.

2. B chooses $k \leftarrow \mathbb{Z}_p$ and sets $PK_b = g^k$, $PK_{1-b} = c/PK_b$, then sends PK_0 to A. A sets $PK_1 = c/PK_0$.

3. A encrypts x_0 with ElGamal scheme using PK_0 (i.e., setting $C_0 = [g^{r_0}, HASH(PK_0^{r_0}) * x_0]$ and encrypting x_1 using PK_1). Then, A sends (C_0, C_1) to B.

4. B decrypts C_b using private key k to obtain $x_b = PK_b^{r_b} * x_b / g^{r_b k}$.

Yao's Garbled Circuit (GC). [Yao, 1986] is a well-known OT-based secure two-party computation protocol that can evaluate any function. The key idea of Yao's GC is to decompose the computational circuits into generation and evaluation stages. The circuits consisting of gates like AND, OR, and NOT can be used to compute any arithmetic operation. Each party is in charge of one stage and the circuit is garbled in each stage, so that any of them cannot get information from the other one, but they can still achieve the result according to the circuit. GC consists of an OT protocol and a block cipher. The complexity of the circuit grows at least linearly with the input size. Soon after GC was proposed, GMW [Goldreich et al., 1987] extended GC to the multi-party setting against malicious adversaries. For more detailed survey of GC, readers can refer to Yakoubov [2017].

OT Extension. Impagliazzo and Rudich [1989] showed that OT provably requires "public-key" type of assumptions (such as factoring, discrete log, etc.). However, Beaver [1996] pointed out that OT can be "extended" in the sense that it is enough to generate a few "seed" OTs based on public-key cryptography, which can then be extended to any number of OTs using symmetric-key cryptosystems only. OT extension is now widely applied in MPC protocols [Keller et al., 2016, Mohassel and Zhang, 2017, Demmler et al., 2015] to improve efficiency.

Secret Sharing

Secret sharing is a concept of hiding a secret value by splitting it into random parts and distributing these parts (a.k.a. shares) to different parties, so that each party has only one share and thus only one piece of the secret [Shamir, 1979, Beimel, 2011]. Depending on the specific secret sharing schemes used, all or a known threshold of shares are needed to reconstruct the original secret value [Shamir, 1979, Tutdere and Uzunko, 2015]. For example, Shamir's Secret Sharing is constructed based on polynomial equations and provides information-theoretic security, and it is also efficient using matrix calculation speedup [Shamir, 1979]. There are several types of secret sharing, mainly including arithmetic secret sharing [Damård et al., 2011], Shamir's secret sharing [Shamir, 1979], and binary secret sharing [Wang et al., 2007]. As arithmetic secret sharing is mostly adopted by existing SMPC-based PPML approaches and binary secret sharing are closely related to OT which is discussed in Section 2.4.1, here we focus on arithmetic secret sharing.

Consider that a party P_i wants to share a secret S among n parties $\{P_i\}_{i=1}^n$ in a finite field F_q. To share S, the party P_i randomly samples $n-1$ values $\{s_i\}_{i=1}^{n-1}$ from $\mathbb{Z}_q$ and set $s_n = S - \sum_{i=1}^{n-1} s_i \mod q$. Then, P_i distributes s_k to party P_k, for $k \neq i$. We denote the shared S as $\langle S \rangle = \{s_i\}_{i=1}^n$.

The arithmetic addition operation is carried out locally at each party. The secure multiplication is performed by using Beaver triples [Beaver, 1991]. The Beaver triples can be generated in an offline phase. The offline phase (i.e., preprocessing) serves as a *trusted dealer* who generates Beaver triples $\{(\langle a \rangle, \langle b \rangle, \langle c \rangle) | ab = c\}$ and distributes the shares among the n parties.

To compute $\langle z \rangle = \langle x \rangle \cdot \langle y \rangle = \langle x * y \rangle$, $P_i{}_{i=1}^n$ first computes $\langle e \rangle = \langle x \rangle - \langle a \rangle$, $\langle f \rangle = \langle y \rangle - \langle b \rangle$. Then, e and f are reconstructed. Finally, P_i computes $\langle z \rangle = \langle c \rangle + e\langle x \rangle + f\langle y \rangle$ locally, and a random party P_j adds its share into ef. We denote element-wise multiplication of vectors as $\langle \cdot \rangle \odot \langle \cdot \rangle$.

Secure multiplication can also be performed by leveraging the Gilboa's protocol [Gilboa, 1999], in which n-bit arithmetic multiplication can be conducted via n 1-out-of-2 OTs. Suppose that Party A holds x and Party B holds y. Now we show Gilboa's protocol, which results in A holding $\langle z \rangle_A$ and B holding $\langle z \rangle_B$ such that $z = x \cdot y$. Let l be the maximum length of the binary representation of the numbers involved in our protocol. Denote the $m\times$ 1-out-of-2 OT for l-bit strings as OT_l^m. Denote the ith bit of x as $x[i]$. The secure 2-party multiplication via OT can be conducted as follows.

1. A represents x in binary format.
2. B builds OT_l^l. For the ith OT, randomly pick $a_{i,0}$ and compute $a_{i,1} = 2^i y - a_{i,0}$. Use $(-a_{i,0}, a_{i,1})$ as the input for the ith OT.
3. A inputs X$[i]$ as the choice bit in the ith OT and obtains x$[i]$ $\times 2^i y - a_{i,0}$.
4. A computes $\langle z \rangle_A = \sum_{i=1}^l (x[i] \times 2^i y - a_{i,0})$
   B computes $\langle z \rangle_B = \sum_{i=1}^l a_{i,0}$.

The offline phase can be carried out efficiently with the help of a *semi-honest dealer* who generates Beaver triples and distributes them among all the parties. To perform such a preprocessing step without a *semi-honest dealer*, there are several protocols available, such as SPDZ [Damård et al., 2011], SPDZ-2 [Damård et al., 2012], MASCOT [Keller et al., 2016], and HighGear [Keller et al., 2018].

- SPDZ is an offline protocol in the preprocessing model based on somewhat homomorphic encryption (SHE) in the form of BGV, first described in Damård et al. [2011].
- SPDZ-2 [Damård et al., 2012] is a protocol based on threshold SHE cryptography (with a shared decryption key).
- MASCOT is an oblivious-transfer-based protocol, proposed in Keller et al. [2016]. It is far more computationally efficient than SPDZ and SPDZ-2.
- In 2018, Keller et al. [2018] developed a BGV-based SHE protocol, called the HighGear protocol, which achieves better performance than the MASCOT protocol.

Application in PPML

Various MPC-based approaches have been designed and implemented for PPML in the past. Most MPC-based PPML approaches leverage a two-phase architecture, comprising of an offline phase and an online phase. The majority of cryptographic operations are conducted in the offline phase, where multiplication triples are generated. The ML model is then trained in the online phase using the multiplication triples generated in the offline phase. The DeepSecure [Rouhani et al., 2017] is a GC-based framework for secure neural network inference, where the inference function has to be represented as a Boolean circuit. The computation and communication cost in GC only depend on the total number of AND gates in the circuit.

SecureML [Mohassel and Zhang, 2017] is another two-party framework for PPML employing two-phase architecture. Parties in federated learning distributes arithmetic shared of their data among two non-colluding servers, who run secure two-party model training protocols. Both Linearly HE (LHE)-based and OT-based protocols are proposed for multiplication triples generation in offline phase. The online phase is based on arithmetic secret sharing and division GC. Therefore, only linear operations are allowed in model training, and various approximations are done to nonlinear functions.

The Chameleon framework is another hybrid MPC framework based on ABY for neural network model inference [Demmler et al., 2015]. Arithmetic secret sharing is used to conduct linear operations, and GC as well as GMW [Goldreich et al., 1987] are used for nonlinear operations. Conversion protocols are also implemented to convert data representations among different protocols.

Privacy-preserving ID3 learning based on OT is addressed in Lindell and Pinkas [2002]. Shamir's threshold secret sharing is used for secure model aggregation for PPML with security against both honest-but-curious and malicious adversaries [Bonawitz et al., 2017], where a group of clients do MPC to evaluate the average of their private input models, and disclose the average to the parameter server for model update. Recently, MPC-based approaches pursuing security against malicious corrupted majority has been studied. For example, linear regression and logistic regression training and evaluation with SPDZ is studied in Chen et al. [2019]. The authors in Damgård et al. [2019] embraces SPDZ$_{2^k}$ [Cramer et al., 2018] for actively secure private ML against a dishonest majority. It implements decision tree and SVM evaluation algorithms.

2.4.2 HOMOMORPHIC ENCRYPTION

HE is generally considered as an alternative approach to MPC in PPML. HE can also be used to achieve MPC as discussed in Section 2.4.1. The concept of HE was proposed in 1978 by Rivest et al. [1978] as a solution to perform computation over ciphertext without decrypting the ciphertext. Since then, numerous attempts have been made by researchers all over the world to design such homomorphic schemes.

The encryption system proposed by Goldwasser and Micali [1982] was a provably secure encryption scheme that reached a remarkable level of safety. It allows an additive operation over ciphertext, but is able to encrypt only a single bit. Paillier [1999] proposed a provable security encryption system that also allows an additive operation over ciphertext in 1999. It has been widely used in various applications. A few years later, in 2005, Boneh et al. [2005] invented a system of provable security encryption, which allows unlimited number of additive operations and one multiplicative operation. Gentry made a breakthrough in 2009 and proposed the first HE scheme that supports both additive and multiplicative operations for unlimited number of times [Gentry, 2009].

Definition

An HE scheme $\mathcal{H}$ is an encryption scheme that allows certain algebraic operations to be carried out on the encrypted content, by applying an efficient operation to the corresponding ciphertext (without knowing the decryption key). An HE scheme $\mathcal{H}$ consists of a set of four functions:

$$\mathcal{H} = \{KeyGen, Enc, Dec, Eval\}, \tag{2.1}$$

where

- *KeyGen*: Key generation. A cryptographic generator g is taken as the input. For asymmetric HE, a pair of keys $\{pk, sk\} = KeyGen(g)$ are generated, where pk is the public key for encryption of the plaintext and sk is the secret (private) key for decryption of the ciphertext. For symmetric HE, only a secret key $sk = KeyGen(g)$ is generated.
- *Enc*: Encryption. For asymmetric HE, an encryption scheme takes the public key pk and the plaintext m as the input, and generates the ciphertext $c = Enc_{pk}(m)$ as the output. For symmetric HE, an HE scheme takes the secret key sk and the plaintext m, and generates ciphertext $c = Enc_{sk}(m)$.

- *Dec*: Decryption. For both symmetric and asymmetric HE, the secret key sk and the ciphertext c are taken as the input to produce the corresponding plaintext $m = Dec_{sk}(c)$.

- *Eval*: Evaluation. The function *Eval* takes the ciphertext c and the public key pk (for asymmetric HE) as the input, and outputs a ciphertext corresponding to a functioned plaintext.

Let $Enc_{enk}(\cdot)$ denote the encryption function with enk as the encryption key. Let $\mathcal{M}$ denote the plaintext space and $\mathcal{C}$ denote the ciphertext space. A secure cryptosystem is called *homomorphic* if it satisfies the following condition:

$$\forall m_1, m_2 \in \mathcal{M}, \quad Enc_{enk}(m_1 \odot_{\mathcal{M}} m_2) \leftarrow Enc_{enk}(m_1) \odot_{\mathcal{C}} Enc_{enk}(m_2) \tag{2.2}$$

for some operators $\odot_{\mathcal{M}}$ in $\mathcal{M}$ and $\odot_{\mathcal{C}}$ in $\mathcal{C}$, where $\leftarrow$ indicates the left-hand side term is equal to or can be directly computed from the right-hand side term without any intermediate decryption. In this book we denote homomorphic encryption operator as $[[\cdot]]$, and we overload the addition and multiplication operators over ciphertexts as follows.

- **Addition:** $Dec_{sk}([[u]] \odot_{\mathcal{C}} [[v]]) = Dec_{sk}([[u + v]])$, where "$\odot_{\mathcal{C}}$" may represent multiplication of the ciphertexts (see, e.g., Paillier [1999]).

- **Scalar multiplication:** $Dec_{sk}([[u]] \odot_{\mathcal{C}} n) = Dec_{sk}([[u \cdot n]])$, where "$\odot_{\mathcal{C}}$" may represent taking the power of n of the ciphertext (see, e.g., Paillier [1999]).

Categorization of HE Schemes

HE schemes can be categorized into three classes: Partially Homomorphic Encryption (PHE), Somewhat Homomorphic Encryption (SHE), and Fully Homomorphic Encryption (FHE). In general, for HE schemes, the computational complexity increases as the functionality grows. Here, we provide a brief introduction to different types of HE schemes. Interested readers can refer to Armknecht et al. [2015] and Acar et al. [2018] for more details regarding different classes of HE schemes.

Partially Homomorphic Encryption (PHE). For PHE, both $(\mathcal{M}, \odot_{\mathcal{M}})$ and $(\mathcal{C}, \odot_{\mathcal{C}})$ are groups. The operator $\odot_{\mathcal{C}}$ can be applied on ciphertexts for a unlimited number of times. PHE is a *group homomorphism* technique. Specifically, if $\odot_{\mathcal{M}}$ is addition operator, the scheme is *additively homomorphic*, and if $\odot_{\mathcal{M}}$ is a multiplication operator, we say that the scheme is *multiplicative homomorphic*. The references Rivest et al. [1978] and ElGamal [1985] represent two typical multiplicative HE schemes. Examples of additive HE schemes can be found in Goldwasser and Micali [1982] and Paillier [1999].

Somewhat Homomorphic Encryption (SHE). An HE scheme is called SHE if some operations (e.g., addition and multiplication) can be applied for only a limited number of times. Some literature also refer to the schemes supporting arbitrary operations while only some limited circuits (e.g., the branching programs [Ishai and Paskin, 2007], garbled circuit [Yao, 1982]) as SHE. Examples are BV [Brakerski and Vaikuntanathan, 2011], BGN [Boneh et al., 2005], and IP [Ishai and Paskin, 2007]. SHE schemes introduce *noise* for security. Each operation on the ciphertext increases the noise of the output ciphertext, and multiplication is the main technique for increasing noise. When the noise exceeds an upper bound, decryption cannot be conducted correctly. This is the reason why most SHE schemes require a limited number of times of applying the operations.

Fully Homomorphic Encryption (FHE). FHE schemes allow both additive and multiplicative operations with unlimited number of times over ciphertexts. It is worth noticing that *additive* and *multiplicative* operations are the only two operations needed to compute arbitrary functions. Consider $A, B \in \mathbb{F}_2$. The *NAND* gate can be constructed by $1 + A * B$. Thanks to its functional completeness, the NAND gate can be used to construct any gate. Therefore, any functionality can be evaluated by FHE. There are four main families of FHE [Acar et al., 2018]: (1) Ideal Lattice-based FHE (see, e.g., Gentry [2009]); (2) Approximate-GCD based FHE (see, e.g., Dijk et al. [2010]); (3) (R)LWE-based FHE (e.g., Lyubashevsky et al. [2010] and Brakerski et al. [2011]); and (4) NTRU-like FHE (see, e.g., López-Alt et al. [2012]).

The existing FHE schemes are built on SHE, by assuming circular security and implementing an expensive *bootstrap* operation. The bootstrap operation re-encrypts the ciphertexts, by evaluating the decryption and encryption functions over the ciphertexts and the encrypted secret key, in order to reduce the noise of ciphertext for further computation. As a result of the costly bootstrap operation, FHE schemes are very slow and not competitive against general MPC approaches in practice. Researchers are now focusing on finding more efficient SHE schemes that satisfy certain requirements, instead of trying to develop an FHE scheme. In addition, FHE schemes assume circular security (a.k.a. key dependent message (KDM) security), which keeps the secret key secure by encrypting it with the public key. However, no FHE is proven to be semantically secure with respect to any function and is IND-CCA1 secure [Acar et al., 2018].

Application in PPML

Many research efforts based on HE have been devoted to PPML in the past. For example, Hardy et al. [2017] proposed algorithms for privacy-preserving two-party logistic regression for vertically partitioned data. Paillier's scheme is leveraged in secure gradient descent to train the logistic regression model, where constant-multiplication and addition operations are conducted via a mask encrypted by Paillier's scheme and the intermediate data computed by each party. The encrypted masked intermediate results are exchanged between the two parties in the secure gradient descent algorithm. Finally, the encrypted gradient is sent to a coordinator for decryption and model update.

CryptoNets [Gilad-Bachrach et al., 2016] is an HE-based methodology announced by Microsoft Research that allows secure evaluation (inference) of encrypted queries over already trained neural networks on cloud servers: queries from the clients can be classified securely by the trained neural network model on a cloud server without inferring any information about the query or the result. The CryptoDL [Hesamifard et al., 2017] framework is a leveled HE-based approach for secure neural network inference. In CryptoDL, several activation functions are approximated using low-degree polynomials and mean-pooling is used as a replacement for max-pooling. The GAZELLE [Juvekar et al., 2018] framework is proposed as a scalable and low-latency system for secure neural network inference. In GAZELLE, to conduct secure nonlinear function evaluation in neural network inference, HE and traditional secure two-party computation techniques (such as GC) are combined in an intricate way. The packed additive homomorphic encryption (PAHE) embraced in GAZELLE allows single instruction multiple data (SIMD) arithmetic homomorphic operations over encrypted data.

FedMF [Chai et al., 2019] uses Paillier's HE for secure federated matrix factorization assuming honest-but-curious server and honest clients. Secure federated transfer learning is studied via Paillier's HE scheme in Liu et al. [2019], where the semi-honest third party is into the discard by mixing HE with additive secret sharing in decryption process.

2.4.3 DIFFERENTIAL PRIVACY

DP was originally developed to facilitate secure analysis over sensitive data. With the rise of ML, DP has become an active research field again in the ML community. This is motivated by the fact that many exciting results from DP can be applied to PPML [Dwork et al., 2016, 2006]. The key idea of DP is to confuse the adversaries when they are trying to query individual information from the database so that adversaries cannot distinguish individual-level sensitivity from the query result.

Definition

DP is a privacy definition initially proposed by Dwork et al. [2006], developed in the context of statistical disclosure control. It provides an information-theoretic security guarantee that the outcome of a function to be insensitive to any particular record in the dataset. Therefore, DP can be used to resist the membership inference attack. The (ϵ, δ)-differential privacy is defined as follows.

Definition 2.2 (ϵ, δ)-differential privacy. A randomized mechanism $\mathcal{M}$ preserves (ϵ, δ)-differential privacy if given any two datasets D and D' differing by only one record, and for all $S \subset Range(\mathcal{M})$,

$$\Pr[\mathcal{M}(d) \in S] \leq \Pr[\mathcal{M}(D') \in S] \times e^{\epsilon} + \delta, \tag{2.3}$$

where ϵ is the privacy budget and δ is the failure probability.

The quantity $\ln \frac{\Pr[\mathcal{M}(D) \in S]}{\Pr[\mathcal{M}(D') \in S]}$ is called the *privacy loss*, with $\ln$ denoting natural logarithm operation. When $\delta = 0$, the stronger notion of ϵ-differential privacy is achieved.

DP has utility-privacy trade-offs as it introduces noise to data. Jayaraman and Evans [2019] found out that current mechanisms for differential privacy for ML rarely offer acceptable utility-privacy trade-offs: settings that provide limited accuracy loss provide little effective privacy, and settings that provide strong privacy result in large accuracy loss.

Categorization of DP Schemes

Typically, there are mainly two ways to achieve DP by adding noise to the data. One is the addition of noise according to the sensitivity of a function [Dwork et al., 2006]. The other is choosing noise according to an exponential distribution among discrete values [McSherry and Talwar, 2007].

The sensitivity of a real-valued function expresses the maximum possible change in its value due to the addition or removal of a single sample.

Definition 2.3 Sensitivity. For two datasets D and D' differing by only one record, and a function $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{R}^d$ over an arbitrary domain, the sensitivity of $\mathcal{M}$ is the maximum change in the output of $\mathcal{M}$ over all possible inputs:

$$\Delta\mathcal{M} = \max_{D,D'} \|\mathcal{M}(D) - \mathcal{M}(D')\|, \tag{2.4}$$

where $\|\cdot\|$ is a norm of the vector. The l_1-sensitivity or the l_2-sensitivity is defined when the l_1-norm or l_2-norm is applied, respectively.

We denote the Laplace distribution with parameter b as $Lap(b)$. $Lap(b)$ has a probability density function $P(z|b) = \frac{1}{2b}\exp(-|z|/b)$. Given a function $\mathcal{M}$ with sensitivity $\Delta\mathcal{M}$, the addition of noise drawn from a calibrated Laplace distribution $Lap(\Delta\mathcal{M}/\epsilon)$ maintains ϵ-differential privacy [Dwork et al., 2006].

Theorem 2.4 *Given a function $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{R}^d$ over an arbitrary domain D, for any input X, the function:*

$$\mathcal{M}(X) + Lap\left(\frac{\Delta\mathcal{M}}{\epsilon}\right)^d \tag{2.5}$$

provides ϵ-differential privacy. The ϵ-differential privacy can also be achieved by adding independently generated Laplace noise from distribution $Lap(\Delta\mathcal{M}/\epsilon)$ to each of the d output terms.

The addition of Gaussian or binomial noise, scaled to the l_2-sensitivity of the function, sometimes yields better accuracy, while only ensuring the weaker (ϵ, δ)-differential privacy [Dwork et al., 2006, Dwork and Nissim, 2004].

The *exponential mechanism* [McSherry and Talwar, 2007] is another way to obtain DP. The exponential mechanism is given a quality function q that scores outcomes of a calculation, where higher scores are better. For a given database and ϵ parameter, the quality function induces a probability distribution over the output domain, from which the exponential mechanism samples the outcome. This probability distribution favors high-scoring outcomes, while ensuring ϵ-differential privacy.

Definition 2.5 Let $q : (\mathcal{D}^n \times \mathcal{R}) \to \mathbb{R}$ be a quality function, which given a dataset $d \in \mathcal{D}^n$, assigns a score to each outcome $r \in \mathcal{R}$. For any two datasets D and D' differing by only one record, let $S(q) = \max_{r,D,D'} \|q(D,r) - q(D',r)\|_1$. Let $\mathcal{M}$ be a mechanism for choosing an outcome $r \in \mathcal{R}$ given a dataset instance $d \in D^n$. Then, the mechanism $\mathcal{M}$, defined as

$$\mathcal{M}(d,q) = \left\{ \text{return } r \text{ with probability } \propto \exp\left(\frac{\epsilon q(d,r)}{2S(q)}\right) \right\} \tag{2.6}$$

provides ϵ-differential privacy.

The DP algorithms can be categorized according to how and where the perturbation is applied.

1. **Input perturbation**: The noise is added to the training data.
2. **Objective perturbation**: The noise is added to the objective function of the learning algorithms.
3. **Algorithm perturbation**: The noise is added to the intermediate values such as gradients in iterative algorithms.
4. **Output perturbation**: The noise is added to the output parameters after training.

DP still exposes the statistics of a party, which are sensitive in some cases, such as financial data, medical data and other commercial and health applications. Readers who are interested in DP and willing to learn more about it can refer to the tutorial given by Dwork and Roth [2014].

Application in PPML

In federated learning, to enable model training on distributed datasets held by multiple parties, *local differential privacy* (LDP) can be used. With local differential privacy, each input party would perturb their data, then release the obfuscated data to the un-trusted server. The main idea behind local differential privacy is *randomized response* (RR).

Papernot et al. [2016] utilized the teacher ensemble framework to first learn a teacher model ensemble from the distributed datasets among all the parties. Then, the teacher model ensemble is used to make noisy predictions on a public dataset. Finally, the labeled public dataset is used to train a student model. The privacy loss is precisely controlled by the number of public data samples inferred by the teacher ensemble. Generative adversarial network (GAN) is further applied in Papernot et al. [2018] to generate synthetic training data for the training of the student model. Although this approach is not limited to a single ML algorithm, it requires adequate data quantity at each location.

Moments accountant is proposed for differentially private stochastic gradient descent (SGD), which computes the overall privacy cost in neural networks model training by taking into account the particular noise distribution under consideration [Abadi et al., 2016]. It proves less privacy loss for appropriately chosen settings of the noise scale and the clipping threshold.

The differentially private Long Short Term Memory (LSTM) language model is built with user-level differential privacy guarantees with only a negligible cost in predictive accuracy [McMahan et al., 2017]. Phan et al. [2017] proposed a private convolutional deep belief network (pCDBN) by leveraging the functional mechanism to perturb the energy-based objective functions of traditional convolutional deep belief networks. Generating differentially private datasets using GANs is explored in Triastcyn and Faltings [2018], where a Gaussian noise layer is added to the discriminator of a GAN to make the output and the gradients differentially private with respect to the training data. Finally, the privacy-preserving artificial dataset is synthesized by the generator. In addition to the DP dataset publishing, differentially private model publishing for deep learning is also addressed in Yu et al. [2019], where concentrated DP and a dynamic privacy budget allocator are embraced to improve the model accuracy.

Geyer et al. [2018] studied differentially private federated learning and proposed an algorithm for client-level DP preserving federated optimization. It was shown that DP on a client level is feasible and high model accuracy can be reached when sufficiently many participants are involved in federated learning.