Enabling Collaborative Use of Data for Mobile Devices under Physical and Privacy Constraints

Submitted in partial fulfillment of the requirements for
the degreee of
Doctor of Philosophy
in
Electrical & Computer Engineering

Yichen Ruan
B.S., Civil Engineering, Tsinghua University
M.S., Civil & Environmental Engineering, University of California, Berkeley

Carnegie Mellon University
Pittsburgh, PA

Mar 2022

Abstract

In the mobile Internet era, data has become the essential ingredient for numerous mobile services such as video streaming, GPS navigation, and on-device intelligence. Nevertheless, the resource-restricted nature of mobile devices often obstructs them from collecting and processing all the data as needed individually. Enabling the collaborative use of data among mobile devices hence greatly extends the potential reach of mobile users for external datasets and computation resources, allowing both public and private data to be more efficiently delivered to and processed by various mobile applications. Implementing this collaboration in the real-world however faces both the physical capacity limits and the privacy protection requirements. Fortunately, the state-of-the-art edge computing framework empowered by 5G-based heterogeneous networks and the recent advances in privacy-preserving federated learning algorithms provide powerful tools based on which innovative systems and algorithms can be developed to address these constraints.

This thesis aims to enable the collaborative use of data for mobile devices through five approaches that fall into two categories: the direct sharing of public or desensitized data, and variants of privacy-preserving federated learning algorithms that are optimized for the mobile environment. For the first category, this thesis proposes to empower the direct data sharing by 1) temporal- and spatial-adaptive mobile caching and 2) topology-aware device-to-device data offloading. For the second category, this thesis improves the existing federated learning implementations by 3) the optimal client recruitment that balances both accuracy and efficiency metrics, 4) innovative extensions to federated training algorithms that incorporate flexible device participation patterns, and 5) soft clustered federated learning that also learns personalized models for clients that collect samples from multiple data distributions.

Acknowledgments

This thesis would not have been possible without the help and support of countless people. First and foremost, I would like to thank my advisor Carlee Joe-Wong for her excellent guidance, constructive feedback, inspiring encouragement, generous support, and insightful advice. Her patience, enthusiasm, knowledge, and ingenuity carried me through all the stages of my Ph.D. study.

I am also grateful to my committee members Yuejie Chi, Osman Yağan, and Xiaoxi Zhang, who provided great suggestions and help for my research. It was their valuable feedback that brought the thesis to completion.

My research was also partially supported by the CMU CyLab IoT Initiative, NSF (CNS-1751075, CNS-1909306 and CNS-2106891), the David Barakat and LaVerne Owen-Barakat Fellowship, and the Ann and Martin McGuinn Graduate Fellowship.

Lastly, my warm and heartfelt thanks go to my family and friends for their ultimate encouragement and unconditional love in my everyday life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The increased user demands for smooth and intelligent mobile Internet services have brought higher requirements for today's mobile applications to utilize their data more efficiently. The forms of data utilization include not only the acquisition of desired content through the wireless network, but also the processing of internally generated and externally fetched data to provide corresponding services. For example, the video streaming applications and GPS navigation services need to speedily download requested video frames and the latest road condition information respectively from the Internet to support seamless user experience. Similarly, state-of-the-art mobile services such as augmented reality and on-device intelligence entail massive data processing and consequent computation to be conducted promptly to generate real-time reactions to user behaviors. However, given the common usage scenarios and hardware specifications of the resource-constrained mobile devices, it is difficult to expect a single user device to acquire and process such a huge amount of data in a reasonable time on its own. Fortunately, modern mobile devices are connected with rich external data sources and computation resources that can possibly be accessed with low communication cost through the ever-evolving mobile network infrastructure. These data and computation resources, which can reside in either publicly available locations or private devices of other users, pave the way for the *collaborative use of data* among distributed user devices.

Mobile users may benefit from the collaborative use of data when one data item attracts common interest from multiple users. In fact, the Internet data traffic has been found to be heavily skewed with popular content being requested much more frequently than the majority of data published on the Internet [2]. For example, trending YouTube videos generally attract much more viewers compared to less noticed uploads [18]. Thus, significant network traffic can possibly be saved by smartly co-using popular data so that all users experience reduced waiting time and communication expenditure.

In recent years, the emerging on-device machine learning (ML) technology features thriving user demands for intensive data collection and processing capacities. Mobile applications equipped with this function, such as voice assistants and recommendation systems, entail massive input data to guarantee the accuracy of their prediction results. While a single device can hardly generate all the required data points, it may benefit from the data collected by other devices that have similar user profiles. Moreover, fitting machine learning models with the training data, also known as model training, consumes tremendous computation resources, which is rarely affordable for ordinary user devices. With the collaboration of mobile users, this training burden can be split by distributing the workload to multiple devices. Empowering the collaborative collection and processing of training data thus greatly extends the reach of common users for more high-quality training samples and relieves the computation burden of individual devices, which guarantees the production of accurate and timely machine learning outcomes to fulfill the user needs.

Enabling the collaborative data utilization requires innovative design in both network systems and software algorithms. On the one hand, supporting infrastructure and communication protocols are needed to facilitate the fast exchange of data and intermediate computation results, which may however produce unwanted interference with existing communication devices without careful management of the network resources. On the other hand, it is the algorithm in the mobile applications that eventually drives the data flow and coordinates all user devices, which determines the effectiveness and efficiency of this distributed working paradigm. The algorithm development is especially important for machine learning tasks since the quality of their final output is largely determined by the way user clients cooperate as we will see in Chapters 4 to 6.

This thesis studies methods that enable the collaborative use of data from both the system and algorithmic dimensions. The application scenarios we will discuss cover both regular data consumption cases and the training of machine learning models, but with an emphasis on the latter since it faces more challenging data sharing requirements as will be discussed in Section 1.2.

1.2 Physical and Privacy Constraints

Despite the potential advantages discussed in Section 1.1, realizing the collaborative use of data in the real world entails overcoming constraints from both the physical and privacy aspects:

- **Physical Constraint**. The speed and throughput of today's network infrastructure degrade when the transmission distance increases. In reality, two user clients that collaboratively use shared data may be far apart physically, which can yield long latency for intensive data exchange tasks. In addition, general mobile devices, including smart phones or tablets, have limited hardware resources such as CPU, memory, storage, battery etc., and are thus incapable of processing complex tasks. Moreover, the unreliable working environment of portable devices also makes mobile devices prone to unexpected failures (e.g., due to low battery or unstable network connectivity). The wireless transmission of data items and computation results may therefore be interrupted occasionally, which may block the running of the whole application without good countermeasures. Designing systems and algorithms under the physical constraint requires innovative optimization of the network topology and prominent improvement in the underlying distributed computing algorithms.
- **Privacy Constraint**. The sharing and reusing of information is the key component and the most direct way of collaborative data use. For example, we may share information to train machine learning models and reuse information to retrieve content as discussed in Section 1.1. However, general data in mobile devices, especially that collected by the users, is usually privacy-sensitive and may not be simply disclosed. Allowing the direct sharing of raw user data, either through a central server or in an device to device manner, brings the risk of privacy leakage and is likely to violate the ever strict government regulations such as the European General Data Protection Regulation (GDPR) [96]. The privacy constraint is particularly crucial for machine learning applications that

heavily rely on user-generated data to create personalized products that best fit the preference of the device owner. To enable the collaborative use of data under the privacy constraint, an innovative framework is needed to guarantee the privacy-preserving information exchange among users. Such a framework should protect the safety of the raw user data, without incurring overwhelming communication overhead to mobile devices. System and algorithm development are also needed to keep such a framework robust and efficient.

The form of collaborative data use discussed in this thesis is always assumed to be restricted by the physical constraint. In words, we assume the network resources are sparse and user devices have much weaker hardware specifications compared to common cloud servers. Regarding the privacy aspect, we consider both privacy insensitive data (e.g. publicly downloadable data on the Internet, or other forms of user information made shareable with appropriate user consent) and privacy sensitive data (e.g. user generated texts, photos, videos etc. that are meant to be kept secret). Thus, the privacy constraint may or may not be active depending on the application scenarios we will discuss. This differentiation is useful because privacy sensitive data needs additional treatment to prevent the information leakage, which will inevitably reduce our approaches' overall performance measured by resource usage, user utility, accuracy of trained models etc. Taking data as privacy insensitive on the other hand allows us to develop more efficient systems and algorithms to better handle the physical constraint.

1.3 Existing Tools

This section introduces two tools based on which we will build our systems and algorithms to empower the collaborative use of data: edge computing and federated learning. The two tools can be respectively applied to solve the physical and privacy constraints as discussed in Section 1.2. However, they both have disadvantages and need further improvement to be applicable in the real world. This thesis helps to provide such improvements through five approaches as we will see in Section 1.4.

1.3.1 Tool 1: Edge Computing

The traditional cloud paradigm, in which applications utilize servers in a remote data center, may not meet new performance requirements in the Internet of Things (IoT), 5G, artificial intelligence, and other emerging technologies. For instance, homeowners may use surveillance cameras to remotely track movement in their backyards in real time. Constantly streaming camera footage to the cloud, from which it can be viewed on homeowners' apps, requires significant amounts of network bandwidth. In another example, running data analytics algorithms on the cloud incurs latency due to the need to transfer data from local devices to cloud servers. Real-time analytics, e.g. analyzing heartbeat patterns collected by a smartwatch to detect medical emergencies, may not be able to tolerate this latency.

Edge computing [81] aims to meet these new performance requirements by distributing some application functionality to edge devices, instead of running them solely in remote datacenters. Thus, instead of applications only utilizing resources at a client and server, they can leverage computing devices that lie across the cloud-to-things (C2T) continuum. For instance, instead of sending data to a cloud server to be analyzed, a smartwatch might send data to a nearby smartphone. Devices' placement on the C2T continuum can be roughly determined by their computational capabilities, e.g. devices like low-power sensors would lie at one extreme, with powerful yet battery-constrained smartphones, connected vehicles, laptops, local servers, and finally remote servers lying close to the edge of the network [84]. Figure 1.1 illustrates this hierarchy of devices. Each device may contribute computing, storage, and communication capabilities; and the devices are generally connected to each other over the Internet. Cloudlets and points-of-presence may be located at the edge of the network, e.g. colocated with mobile base stations. Edge computing is predicted to be the next multi-billion-dollar business [9]. It is a computing architecture that locates functionalities (e.g. storage, computation, communication, and management) at two or more types of devices along the C2T continuum. Edge computing thus represents a generalization of traditional cloud computing, in which application functionality resides at a local device and a remote cloud server.

While edge computing shows promise for addressing applications' bandwidth and latency challenges, it also raises new research challenges. Typically, edge devices

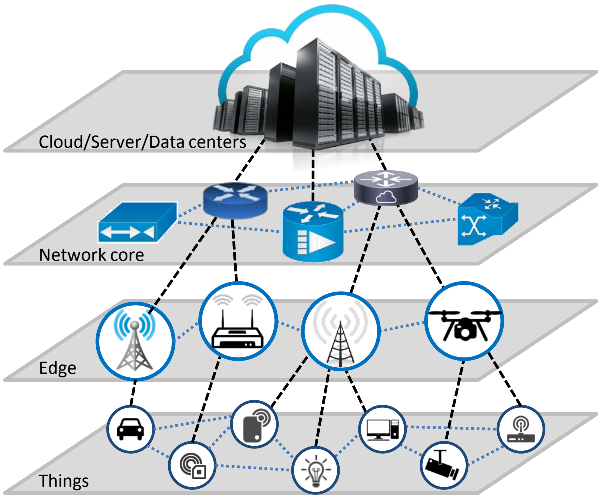


Figure 1.1: Illustration of devices on the cloud-to-things continuum

are equipped with limited processing power and storage capacity compared to the powerful servers in the cloud setting, and the bandwidth and spectrum resources are also sparse. The form of collaborative data use we will discuss is assumed to happen in such an edge computing setting where the network and computation resources are always constrained.

1.3.2 Tool 2: Federated Learning

Federated learning (FL) is a cutting-edge privacy-preserving learning framework that allows distributed devices to train a shared machine learning model cooperatively with their own data. During the training process, the user data will never leave the user device, thus protecting the user privacy. Recently, federated learning has exhibited remarkable performance in many applications such as next word suggestion, fault detection, and learning on private medical data [51].

Traditional federated learning involves a coordinator and a collection of devices (also known as clients). Depending on the type of the devices, federated learning can be classified into cross-device federated learning and cross-silo federated learning [45]. The devices in cross-silo federated learning are usually powerful servers, and are thus out of the scope of our discussion. This thesis focuses mostly on cross-device federated learning, where participating devices are the ordinary mobile devices such as smart phones, tablets and laptops. The coordinators are typically companies or organizations that hope to train machine learning models using user data in a privacy-preserving way. The coordinator is responsible for initiating the training and synchronizing all participating clients through a central server. Clients and the server communicate through the wireless network to exchange some necessary information periodically. The training procedure usually consists of multiple rounds, where each round includes the following four steps:

- **Client selection**: The coordinator selects a random subset of clients to participate in the training for this round
- **Synchronization**: the coordinator synchronizes the latest global model with all selected devices
- **Local updates**: each selected device trains a local model for a few local epochs, using samples from its local dataset
- **Aggregation**: the coordinator aggregates local models from selected clients to produce the next global model

Federated learning was first introduced with the **FedAvg** algorithm [61]. Many experiments since then have shown that **FedAvg** can indeed produce accurate machine learning models (e.g. [105]). The convergence of **FedAvg** has been proved for strongly convex and Lipschitz continuous objectives [53, 99]. These works also prove that the non-IID distribution of local datasets can greatly obstruct convergence. Recently, several variants of **FedAvg** are proposed to enhance the power of federated learning. For example, **FedProx** [52] allows clients to use different local solvers, and to upload partially finished models for aggregation, increasing the robustness of the system. [89] on the other hand proposes a multi-task learning framework that allows individual clients to train their own models. Some other papers also incorporate system characteristics. For example, [99] proposes an adaptive control algorithm that jointly considers the accuracy and the cost of training. [15] considers federated learning over wireless networks, incorporating wireless communication and energy consumption.

Federated learning enables clients to collaboratively use their data while protecting the privacy of participating clients. It is also a typical edge computing system where most computation (i.e. local updates) are conducted on user devices instead of powerful cloud servers. However, traditional federated learning suffers from both the physical restrictions of participating devices and the non-IID (not independently and identically distributed) distribution pattern of user data [89] . In this thesis, we will design more efficient federated learning implementations to better serve the demands of the mobile devices while keeping the promise of privacy protection. Specifically, our approaches aim to tackle both the resource limitations and the statistical challenges imposed by the non-IID effect.

1.4 Overview of Our Approaches

Depending on the privacy characteristics of the user data, mobile applications can be classified into *two types*. For both types, we develop innovative systems and algorithms to enable the collaborative data use in the mobile environment and conduct extensive analysis and simulations to verify the effectiveness of our methods.

Type 1: The direct sharing of public or desensitized data. For applications of this type, e.g. GPS navigation services that use public road information, we neglect the privacy constraint and assume all data can be freely exchanged among users and shared with trusted third parties. Our approaches thus focus on improving the efficiency of the data movement with restricted resources in the edge environment. We propose two approaches for Type 1:

- **Mobile caching** [77]: We propose to use mobile (i.e., moving) devices to supplement existing network infrastructure according to users' data needs at different times and locations. For instance, vehicles can be used as communication relays or computation points. However, it is unclear how much value these devices add relative to their deployment costs: they may, for instance, interfere with existing network infrastructure, limiting the potential benefits. We take the first step towards quantifying the value of this supplemental infrastructure by examining the use case of mobile caches. We consider a network operator using both mobile (e.g., vehicular) and stationary (small cell) caches, and find the optimal amount of both types of caches under time- and location-varying user demands, as a function of the cache prices. In doing so, we account for interference between users' connections to the different caches, which requires solving a non-convex optimization problem. We show that there exists a threshold price above which no vehicular caches are purchased. Moreover, as the network operator's budget increases, vehicular caching yields little additional value beyond that provided by small cell caches. These results may help network operators and cache providers find conditions under which vehicles add value to existing networks.

- **Topology-aware data offloading** [93]: We develop a network-aware distributed learning optimization methodology where devices process data for a task locally and send their learned parameters to a server for aggregation at certain time intervals. Unlike traditional federated learning frameworks, our method enables devices to offload their data processing tasks, with these decisions determined through a convex data transfer optimization problem that trades off costs associated with devices processing, offloading, and discarding data points. We analytically characterize the optimal data transfer solution for different edge network topologies, showing for example that the value of allowing device offloading is approximately linear in the range of computing costs in the network. Our subsequent experiments on both synthetic and real-world datasets we collect confirm that our algorithms are able to improve network resource utilization substantially without sacrificing the accuracy of the learned model.

Type 2: Innovative variants of federated learning that are optimized for the realistic edge environment. For applications of this type, we assume the data is highly sensitive and may not be disclosed to anyone under any circumstance, e.g. private user photos, texts etc. Our approaches extend the existing federated learning algorithms to increase the learning efficiency in the resource-constrained edge environment and overcome the difficulty in convergence imposed by the non-IID problem, while sticking to the promise of privacy protection. We propose three approaches for Type 2:

- **Optimal client recruitment** [79]: In federated learning, an operator recruits user devices (i.e., clients) to occasionally perform local iterations of the learning algorithm on their data. We propose the first approach to theoretically analyze the resulting performance tradeoffs in deciding which clients to recruit for federated learning, complementing other works on the selection of recruited clients in each iteration. Specifically, we define and optimize the tradeoffs between both accuracy (training and testing) and efficiency (completion time and cost) metrics. We provide efficient solutions to this NP-Hard optimization problem, and verify the value of client recruitment in experiments on synthetic and real-world data. Our results can serve as guidelines for the real-world deployment of federated learning and an initial investigation of the client recruitment problem.

- **Flexible device participation in federated learning** [80]: Traditional federated learning algorithms impose strict requirements on the participation rates of devices, which limit the potential reach of federated learning. This approach extends the current learning paradigm to include devices that may become inactive, compute incomplete updates, and depart or arrive in the middle of training. We derive analytical results to illustrate how allowing more flexible device participation can affect the learning convergence when data is non-IID. We then propose a new federated aggregation scheme that converges even when devices may be inactive or return incomplete updates. We also study how the learning process can adapt to early departures or late arrivals, and analyze their impacts on the convergence.
- **Soft clustered federated learning** [78]: Traditionally, clustered federated learning groups clients with the same data distribution into a cluster, so that every client is uniquely associated with one data distribution and helps train a model for this distribution. We relax this hard association assumption to soft clustered federated learning, which allows every local dataset to follow a mixture of multiple source distributions. We propose **FedSoft**, which trains both locally personalized models and high-quality cluster models in this setting. **FedSoft** limits client workload by using proximal updates to require the completion of only one optimization task from a subset of clients in every communication round. We show, analytically and empirically, that **FedSoft** effectively exploits similarities between the source distributions to learn personalized and cluster models that perform well.

1.5 Structure of the Thesis

In this chapter, we introduce the motivation and background for enabling the collaborative data use in the mobile environment. We highlight the physical and privacy challenges associated with this problem, and give an overview of our five approaches that leverage the state-of-the-art edge computing and federated learning tools. Chapters 2 to 6 respectively elaborate these five approaches following the order they are presented in Section 1.4. We finally conclude the thesis in Chapter 7 and discuss possible future directions.

Chapter 2

Data Co-use via Mobile Caching

2.1 Introduction

In recent years, network operators have made efforts to move 5G network functionalities to the network edge [6], e.g., caching popular contents on small cells or user devices. These contents can then be retrieved directly from the network edge, without passing through the network core. Such practices have yielded concrete benefits for network operators, e.g., caching in small cells can reduce backhaul traffic by over 45% [91]. Despite these benefits, relying on edge infrastructure also has drawbacks. Small cells have limited range, requiring dense deployments, and cannot adapt to changes in user demand over time [116]. For instance, user demands in a business region of a city are generally much larger during the day than at night, making small cells stay idle for almost half a day. Network infrastructure with more temporal flexibility could then yield further benefits to network operators by adapting to changes in user demands.

Some recent work [27, 63, 65, 86, 95, 109] has proposed to deploy mobile devices such as drones and vehicles as relay nodes and cache carriers in wireless networks. The physical mobility of these devices allows them to respond to temporal variations in user demands and network conditions at different locations. However, it is not clear whether the benefits of this flexibility outweigh the costs. Using vehicles or other mobile devices, for instance, may be more expensive than simply over-provisioning stationary devices to handle peak user demands at each location, particularly if user demands do not vary enough over time. Users' communication with such devices may also interfere with connectivity to existing network infrastructure. Yet it is hard to say exactly what constitutes “too expensive” or “not varied enough.” In this chapter, we **quantify the value of mobile caching** as a first step towards assessing the value of using vehicles or other mobile devices in network infrastructure. Our results can guide network operators in determining whether and to what extent they should incorporate such devices into their networks.

Mobile devices like vehicles and drones can perform a number of network functions, ranging from caching content [109] to serving as relays [65] to collecting data in urban crowdsensing [74]. We focus on vehicular caching due to caching’s known benefits in mobile networks, and vehicles’ naturally high density compared to small cells or other infrastructure in an urban environment. One might, for instance, pay public buses, private car owners, or other mobility-as-a-service providers to carry cache servers; some bus systems already carry WiFi access points [33]. Our economic model, however, is general enough to cover any type of cache provider that charges the network operator that uses the caches.

Most prior works on wireless caching focus on finding the optimal caching policies, with little attempt at analyzing the competition between, and relative value of, mobile vehicular caches and other types of caching. We thus encounter several new research challenges in doing so. These involve both *finding the right model* to quantify the value of vehicular caching and *solving the resulting optimization problems*, which we show are in general non-convex. We solve these challenges to make three key research contributions:

Our first contribution lies in **modeling interactions between caching tiers**: Vehicular caching must compete *economically* with existing caching products, such as small cells. The relative costs of these different types of caching will then influence the additional value that vehicular caching brings. We quantify this effect by jointly optimizing the caching purchased from each tier. Vehicular points may also experience *physical interference* from small cells and other network traffic, which affects their ability to serve users and thus their value. Quantifying the effect of such interference between caching tiers is itself a challenging problem, and we use stochastic geometry theories to do so.

Our second research contribution is to develop a **unified framework for different network operator objectives** given the above system model. These include the rate of cache hits and the delivery rate, which quantifies the spectral efficiency of users' connections to different caching tiers. We formulate tractable optimization problems for both objectives in a general framework (*Theorems 2.1-2.3*).

Our third major research contribution lies in solving these optimization problems to **concretely quantify the value of vehicular caching**. This value depends on the distribution of user needs over time and space, which is itself hard to quantify given the many variables involved. We show that it also depends on solving for the optimal cache provisioning and provide solutions for this (non-convex) optimization problem (Theorems 2.4-2.5). In particular, we will evaluate the value of vehicular caching by answering three questions:

- **Demand functions**: We solve for network operators' optimal caching demands (*Theorems 2.4 and 2.5*), and quantify the dependence of vehicle demands at each time on the prices and user needs at other times (*Corollary 2.1*);
- **Economic viability of vehicular caching**: We quantify the maximum price under which operators would wish to purchase vehicular caching capacities, i.e., under which the value added by vehicular caching exceeds its cost relative to other caching methods (*Corollary 2.2*);
- **Gain from competition**: We show that as the network operator's budget increases, vehicular caching yields little marginal value above small cell caching (*Corollary 2.3*), but that the optimal amounts of both vehicular and small cell caching increase linearly with the budget (*Corollary 2.4*).

2.2 Related Work

Many existing works on network caching consider a fixed caching capacity that the network operator can sell to different content providers (CPs) in order to reduce the latency of delivering content to these CPs' users. Economic techniques such as contract theory [37], auction theory [41], and game theory [115] can be used to find the optimal or equilibrium outcomes. Unlike these works, in this chapter, we focus on *network operators purchasing cache devices*, whose capacity may later be sold to CPs.

Many works on wireless caching attempt to design optimal content placement schemes. Popular methods include small cell caching [25, 68], device-to-device (D2D) cache sharing [44, 98], mobility-aware caching [98], and content sharing in vehicular ad hoc networks [111]. Comparing these methods reveals that caching in different locations of the wireless network can result in different levels of economic gains [11], but these benefits are not rigorously analyzed. To do so here, we make use of the content placement, device mobility, and wireless connectivity models introduced in these works.

Early works on vehicular caching view vehicles as moving relay nodes that connect user devices to base stations [5, 107]. Later works have considered optimizing user connectivity to vehicles that host caches themselves [39, 109, 112]. In [86, 95], the optimal content placement policy is studied for a 2-tier network with cache-enabled vehicular points and macro cells. Similar models are studied for specific applications, such as the delivery of delay-tolerant contents [63], and video streaming [95]. These works focus only on vehicles and do not consider the existence of other caching methods, e.g., small cells. In contrast, we will explore how vehicular, and more generally mobile, caching adds value to the existing caching methods.

2.3 System Model

We first show the overall system (Figure 2.1) and our user and vehicle mobility model in Section 2.3.1, and then describe our models for the content allocation policy (Section 2.3.2) and users' connectivity to the caching devices (Section 2.3.3).

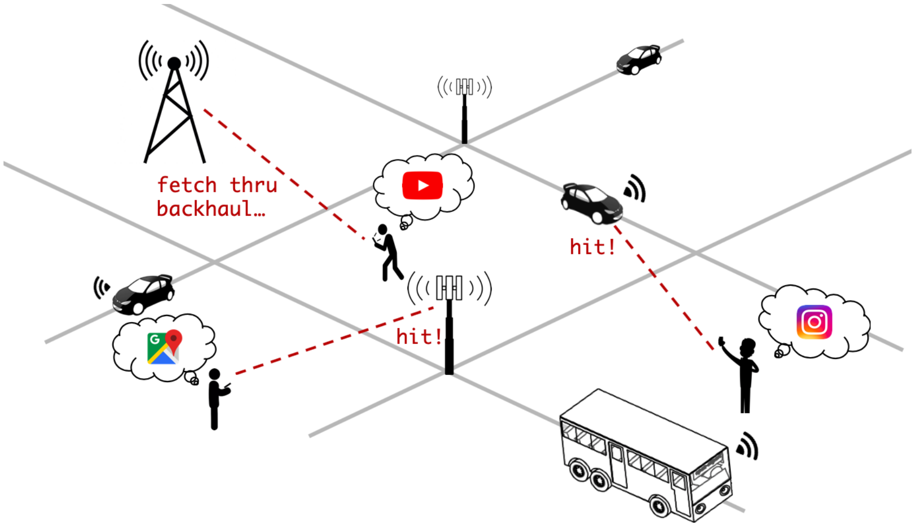


Figure 2.1: Overview of the heterogeneous network composed of three caching tiers

2.3.1 Content Caching Market

We consider a network operator, e.g., an Internet service providers (ISP) or content delivery network (CDN) provider, that wishes to utilize caching capacity from a variety of cache providers. We consider two tiers of caching products: small cell caching (which has fixed locations) and vehicular caching. These products can come from different providers. For example, equipment manufacturers may sell cache capacity at small cells, while ride-sharing networks can sell caching capacity in their vehicles. The network operator subscribes to a combination of these two products to maximize its profit, i.e., the benefit from caching, less the payment to the cache providers. To focus on the value that vehicular caching adds to mobile networks and not on the price competition between the small cell and vehicular providers, we assume the prices that the network operator pays for each type of caching are given. For simplicity, we do not consider D2D caching as it is not yet widely deployed. However, our framework can be easily extended to include more caching tiers.

We assume that network operators purchase caching capacity in terms of the expected number of devices in a unit area, i.e., they decide the intensity of small cells L_s and vehicles L_v to be deployed in each region of their networks. We use a stochastic model for the number of vehicles and small cells available in each region and time interval to abstract away from individual device mobility. Indeed, these devices may not be fully controlled by network operators: for instance, taxi vehicles may follow varied mobility patterns. Small cell caches may become inaccessible due to network congestion or other demands. We also assume an existing macro cell tier with fixed, exogenous intensity L_m, leading to a 3-tier heterogeneous network as in [3]. The macro cell can access all content through the network backhaul. Devices of the same tier share the same configuration, i.e., they have the same transmit power p_k, cache capacity N_k and price P_k, where $k \in \mathcal{N} = \{v, s, m\}$ represents vehicles, small cells and macro cells respectively. This price represents the amount that operators need to pay vehicle or small cell owners to cache content for them and may vary with time and location.

To address temporal and spatial dynamics, we divide the decision space into T time slots and D regions. Inside each $(t, d) \in [T] \times [D]$, we assume the locations of all caching points and users follow a homogeneous Poisson point process with constant intensity $L_k(t, d), k \in \mathcal{N} \cup \{u(\text{users})\}$ [7, 98]. Since macro cells and small cells are stationary, their intensity matrices are fixed over time, i.e. $\forall t_1, t_2, L_k(t_1, d) = L_k(t_2, d), k \in \{m, s\}$. The intensity of users $L_u(t, d)$ at each time t and location d is exogenous, and we assume that it is not affected by the cache intensities. The (exogenous) prices $P_k(t, d)$ can also vary with the time t and region d. The mobility of mobile users and vehicles is reflected in the change of intensities across time and space.

2.3.2 Content Placement Policies

The value of vehicular caching depends on how content is cached at both the vehcles and small cells. Contents are assumed to be cached as chunks with the same size [95]. Each chunk is associated with an exogenous preference (probability of being requested) $f_i, i = 1, 2, \ldots, N$, where N is the size of the content catalog. Without loss of generality, we assume $f_1 \geq f_2 \geq \ldots \geq f_N$ following a power-law distribution as in [2]. Chunks from the same content source may have different preferences, e.g., some segments of a video may be more popular than others [95]. The preference, and thus the indices i, can also depend on (t, d), e.g., news videos might be more popular in the morning. For clarity, we generally omit the dependence on (t, d) in our notation.

The content placement policy is modeled in terms of the caching probability. In tier k, the content i is cached with probability $H_{k,i}(t, d)$. All chunks are accessible to the macro cell tier, i.e. $N_m = N$ and $H_{m,i}(t, d) \equiv 1$. By choosing the values of H, we can model different placement policies, provided that the cache does not exceed its capacity, i.e., $\sum_{i=1}^{N} H_{k,i} \leq N_k$. We will consider policies of the form

$$H_{k,i}(t, d) = \begin{cases} u_k \triangleq \frac{N_k}{\mu_k N}, & 1 \leq i \leq \mu_k N \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

where μ_k is an exogenous parameter satisfying $\frac{N_k}{N} \leq \mu_k \leq 1$. A larger μ_k means that less popular content (with higher index i) is more likely to be cached. We make the reasonable assumption that $N_v \leq N_s$ and $\mu_v \leq \mu_s$, i.e., vehicles cache fewer chunks than small cells, since they have less space to install storage units. When μ_k takes its extreme values, we obtain two easily interpretable policies:

- **Greedy placement policy** ($\mu_k = N_k/N, u_k = 1$). Devices in each tier cache chunks in descending order of preferences until they reach their capacity N_k.
- **Uniform placement policy** ($\mu_k = 1, u_k = N_k/N$). For each tier, all chunks are cached with the same probability.

We assume that content is replaced when its popularity changes. Since content popularity changes infrequently compared to user requests, we do not include it in our model.

2.3.3 Network Interference and Connectivity

We next characterize the network connectivity of each caching tier in order to quantify its benefits to the network operator in the next section. We consider the standard power law propagation model and Rayleigh fading for all tiers [85], i.e., a receiver with distance r from the transmitter receives $p \cdot h \cdot r^{-\alpha}$ of signal power, where p is the transmit power, $h \sim \exp(1)$ represents the fading effect, and $\alpha > 2$ is the path-loss coefficient. Communication inside a region d is assumed to only interfere with devices within that region. Different tiers may or may not interfere with each other, depending on the spectrum allocation scheme. We will consider spectrum sharing, in which all tiers share the same spectrum and may interfere with each other; and an orthogonal partition in which different tiers use different spectrum bands, e.g., cellular connectivity to small cells and WiFi connectivity to vehicles.

Clients can retrieve contents from a caching point only if the requested content is cached and the signal-to-interference ratio (SIR) is greater than some threshold τ_k, which we assume is the same for all tiers ($\tau_k \equiv \tau$). We assume that user requests are always directed to the point with the highest SIR among all points that have the requested content, regardless of its tier. This assumption guarantees that users always benefit from caching, and can be easily relaxed by assigning to each tier a user association preference [54].

2.4 Problem Formulation

We propose to use our system model to answer three specific questions regarding the value of vehicular caches:

- **Demand functions.** How do the optimal demands for vehicular and small cell caches change with respect to the time, region, prices, and user intensities?
- **Economic viability.** Can operators make money by subscribing to vehicular caching? How much value does vehicular caching add to existing caching products?
- **Value of competition.** Compared to using only small cells or vehicles, how much gain in utility, or reduction in cost, can the operator obtain when both are available?

The first two questions can be answered by solving the operator's profit maximization problem to find the optimal intensities L_v^*, L_s^* to which it should subscribe.

Problem 2.1 (Profit Maximization)**.**

$$\begin{aligned} \underset{L_v, L_s \in \mathbb{R}^{T\times D}}{\textit{maximize}} \quad & \gamma U(L_v, L_s) - \textit{tr}(P_v^T L_v) - \textit{tr}(P_s^T L_s) \\ \textit{subject to} \quad & L_v(t,d) \geq 0, L_s(t_1,d) = L_s(t_2,d) \geq 0 \end{aligned}$$

Here $U(L_v, L_s)$ is a utility function, to be defined in Section 2.5, quantifying the benefits of different caching intensities L_v and L_s for the operator, and γ is a scaling coefficient. The remaining terms represent the cost of using each type of cache.

To quantify the added value of vehicular compared to small cell caches, we solve a variant of Problem 2.1:

Problem 2.2 (Utility Maximization)**.**

$$\begin{aligned} \underset{L_v, L_s \in \mathbb{R}^{T\times D}}{\textit{maximize}} \quad & U(L_v, L_s) \\ \textit{subject to} \quad & \textit{tr}(P_v^T L_v) + \textit{tr}(P_s^T L_s) \leq P_0 \\ & L_v(t,d) \geq 0, L_s(t_1,d) = L_s(t_2,d) \geq 0 \end{aligned}$$

This problem quantifies the maximum attainable utility subject to a fixed budget constraint. The value of competition is thus the difference of the achieved utilities for the competitive market and single-product markets (i.e., when only vehicular or small cell caches are available). In reality, vehicular and small cell caching providers may decrease their prices in order to better compete with each other, leading to higher utility for the network operator; thus, the solution given by Problem 2.2 can be interpreted as a lower bound of the real utility gain.

We will solve these optimization problems to answer our research questions analytically and numerically in Sections 2.6 and 2.7. First, we define the utility function $U(L_v, L_s)$.

2.5 Defining Operator Utilities

The utility function measures the benefits that an operator obtains from caching. Different types of operators may then have different metrics for utility. We first define a general network operator utility model and then discuss three special cases that are particularly realistic models for operators that are CDN providers that sell cache capacity to CPs or ISPs who use caching to improve their users' quality-of-service.

General utility model. All of our utility models can be written in terms of the probability that users can connect to each caching tier. Thus, we define C_k as an indicator variable of whether a user chooses tier k and connects to it. Formally, $C_k = \mathbb{1}(SIR_k > \tau, Sel = k)$, where Sel is a random variable denoting the tier selected for serving a typical request. We then write the general utility functions

$$U_0 = \mathbb{E}[a(C_v + C_s) + bC_m] \tag{2.2}$$

where a and b are scalar weights; we use specific values of a and b to define three cases of particular interest below. We also write Sel_i as the tier selected to provide content i. We can now solve for the *tier k connectivity*, defined as probability that users can access tier k, $\mathbb{E}[C_k]$:

Theorem 2.1 (Expected Tier Connectivity)**.**

$$\mathbb{E}[C_k] = \frac{1}{\|L_u\|_1} \sum_{t,d} L_u(t,d) \sum_{i=1}^{N} f_i(t,d) \mathbb{E}[C_{k,i}(t,d)] \tag{2.3}$$

where $C_{k,i}(t,d) = \mathbb{1}(SIR > \tau, Sel_i = k|t,d)$ indicates if a typical request for content i is served by tier k at (t,d), and $\|\cdot\|_1$ is the ℓ_1 norm. For a typical user, let $X_{k,i}$ be the distance to the nearest tier-k point that has content i cached and $R_k = \min_i X_{k,i}$.

If $R_k|Sel_i = X_{k,i}|Sel_i, \forall i$*, then*

$$\mathbb{E}[C_{k,i}(t,d)] = \frac{H_{k,i}(t,d)p_k^{\frac{2}{\alpha}}L_k(t,d)}{\sum\limits_{j\in\mathcal{N}}(\rho + H_{j,i}(t,d))p_j^{\frac{2}{\alpha}}L_j(t,d)} \tag{2.4}$$

$$\mathbb{E}[C_{k,i}(t,d)] = \frac{H_{k,i}(t,d)p_k^{\frac{2}{\alpha}}L_k(t,d)}{\rho p_k^{\frac{2}{\alpha}}L_k(t,d) + \sum\limits_{j\in\mathcal{N}}H_{j,i}(t,d)p_j^{\frac{2}{\alpha}}L_j(t,d)} \tag{2.5}$$

for the sharing and orthogonal spectrum schemes respectively, with $\rho = \tau^{2/\alpha}\int_{\tau^{-\frac{2}{\alpha}}}^{\infty}(1+u^{\frac{\alpha}{2}})^{-1}du$ *a function of* τ *and* α*.*

Proof sketch. For any (t,d), we use the probability density functions (PDFs) of R_k and $X_{k,i}$ and the approximation $R_k|Sel_i = X_{k,i}|Sel_i$ to find the joint probability $\mathbb{P}(R_k > r, Sel_i = k)$, as well as the PDF of $R_k|Sel_i$. Then we use the law of total probability to find $\mathbb{P}(SIR > \tau|Sel_i = k)$, which we simplify by using the Laplace transform of the interference. Reorganizing, we find $\mathbb{E}[C_{k,i}(t,d)]$ as in (2.4), (2.5), and we use the law of total expectation to derive (2.3). $\square$

The approximation $R_k|Sel_i = X_{k,i}|Sel_i$ in Theorem 2.1, i.e., that content i is served by the closest physical point with that content, holds when all points in the same tier in each region cache the same items, as for the greedy placement policy. In general, it is close to reality if the requested content is cached with a high probability, i.e., when $H_{k,i}$ is close to 1. For less popular contents i, the approximation has limited influence on $\mathbb{E}[C_k]$ in (2.3) since the corresponding preferences f_i are small.

Utility from cache connectivity. Our first utility model can apply to network operators that are ISPs or CDN providers. Since CDN providers are typically paid for the amount of data directed to their devices, we can model their utility as the probability that a typical request is served by the cache, i.e., the rate of successful cache connectivity, which we define as the sum of the vehicle and small cell connectivity:

$$U_1 = \mathbb{E}[C_v + C_s] \tag{2.6}$$

where we have set $a = 1, b = 0$ in (2.2). The γ coefficient in Problem 2.1's objective then denotes the marginal value of cache connection, which is proportional to the per-byte monetary payoff to the CDN service. Similarly, an ISP might also receive a payoff proportional to the cache connectivity, as each cache connection reduces the operating cost of serving the user's request through the core network.

When the network coverage is good and interference can be ignored, we can assume the SIR threshold $\tau = 0$. In this case the cache connectivity ($C_v + C_s = \mathbb{1}(SIR > \tau, Sel_i = s, v|t, d)$) simply becomes the cache hit rate since we have $SIR_k > \tau = 0$. We thus define the **cache hit utility**, U_2:

$$A_k \triangleq \mathbb{1}(SIR_k > 0, Sel = k) = \mathbb{1}(Sel = k) \tag{2.7}$$

$$U_2 = \mathbb{E}[A_v + A_s]. \tag{2.8}$$

By assuming no interference, U_2 represents an optimistic approximation of the cache connectivity; while U_1 can be viewed as the worst-case estimation of cache connectivity as the derivation of C_k in Theorem 2.1 assumes all stations are transmitting at all time, which is unlikely to happen in reality. The gap between U_1 and U_2 gets larger as we increase the SIR threshold τ. With U_2, we are able to get closed-form solutions for Problems 2.1 and 2.2 in the next section. We first derive U_2 and prove its concavity.

Theorem 2.2 (Expected Cache Hit Rate). $\mathbb{E}[A_k]$ *is given by*

$$\mathbb{E}[A_k] = \frac{1}{\|L_u\|_1} \sum_{t,d} L_u(t,d) \sum_{i=1}^{N} f_i(t,d)\mathbb{E}[A_{k,i}(t,d)] \tag{2.9}$$

where $A_{k,i} = \mathbb{1}(Sel_i = k)$*, and*

$$\mathbb{E}[A_{k,i}(t,d)] = \frac{H_{k,i}(t,d)p_k^{\frac{2}{\alpha}} L_k(t,d)}{\sum\limits_{j \in \mathcal{N}} H_{j,i}(t,d)p_j^{\frac{2}{\alpha}} L_j(t,d)} \tag{2.10}$$

Proof. The proof is analogous to the proof of Theorem 2.1. □

Since the interference has been removed, (2.9) and (2.10) are independent of the spectrum allocation scheme, and Theorem 2.2 does not require that $X_{k,i}|Sel_i = R_k|Sel_i$ as in Theorem 2.1. In fact, (2.10) is a special case of (2.4) and (2.5) when $\tau = 0$. We find that the cache hit utility U_2 is a concave function:

Theorem 2.3 (Concavity of Cache Hit Utilities). *The utility function (2.8) is concave with respect to* L_v *and* L_s*.*

Proof. Plugging (2.9) and (2.10) into (2.8), the utility function is

$$U_2 = \sum_{t,d,i} \frac{L_u(t,d) f_i(t,d)}{\|L_u\|_1} (1 - \mathbb{E}[A_{m,i}(t,d)]) \tag{2.11}$$

From (2.10), $\mathbb{E}[A_{m,i}(t,d)]$ is a convex function. Thus $(1 - \mathbb{E}[A_{m,i}(t,d)])$ is concave, as is U_2 in (2.11). □

We finally define a third type of utility function, which is again a special case of the general utility (2.2).

Delivery rate utility. If the network operator is an ISP, caching can not only reduce its cost, as discussed above, but also improve user QoS and attract new users. ISPs might then wish to maximize the downlink delivery rate [7], which characterizes the throughput experienced by a typical user and thus the spectrum efficiency. The marginal value of this utility (i.e. the value of γ in Problem 2.1) can be determined using tools like the discrete choice model [24].

For the utility function to represent the delivery rate, we set $a = \log(1+\tau), b = r_b < \log(1+\tau)$ in the utility expression (2.2). That is, the typical user is served with constant rate $\log(1+\tau)$ if she is in coverage and served by cache tiers. If she is in coverage yet served by macro cells, the latency is controlled by the processing and queuing rate of the backhaul; thus the user is served with a lower delivery rate $r_b < \log(1+\tau)$. The value of r_b is jointly determined by factors such as the maximum bandwidth of the backhaul, the distribution of macro cells, etc.; we treat it as a known constant. As a result, we can write the **delivery rate utility**, U_3, from (2.2) as

$$U_3 = \mathbb{E}[\log(1+\tau)(C_v + C_s) + r_b C_m] \tag{2.12}$$

2.6 Optimal Caching Intensities

In this section, we will answer Section 2.4's research questions on the caching demand functions and economic viability by solving Problem 2.1, and on the value of competition by solving Problem 2.2, with Section 2.5's utility functions. With the cache hit utility U_2 in (2.8), both problems are convex, and can be solved in closed-form. However, the cache connectivity and delivery rate utilities U_1 and U_3 in (2.6) and (2.12) are not concave. We design an efficient fractional programming algorithm to find the optimal solution in these cases.

2.6.1 Solutions for Profit Maximization (Problem 2.1)

We first consider the cache hit utility (2.8) before giving an algorithm to solve Problem 2.1 with more general objectives.

Theorem 2.4 (Demand Functions under Cache Hit Utilities)**.** *Under the general placement policy (2.1), assume* $\mu_v \leq \mu_s$*. With utility function (2.8), the solution to Problem 2.1 is:*

$$\begin{aligned} L_s^*(\cdot,d) = &-\frac{1}{u_s}F_{ms}L_m(\cdot,d) + \\ &\sqrt{\frac{\gamma F_{ms}L_m(\cdot,d)\sum_t(Q_s(t,d)-\delta_v(t,d)Q_v(t,d))L_u(t,d)}{\left\|L_u\right\|_1 u_s\sum_t P_s(t,d)-\delta_v(t,d)\frac{u_s}{u_v}F_{sv}P_v(t,d)}} \end{aligned} \tag{2.13}$$

$$\begin{aligned} L_v^*(t,d) = &\sqrt{\frac{\gamma F_{mv}L_m(\cdot,d)Q_v(t,d)L_u(t,d)}{\left\|L_u\right\|_1 u_v P_v(t,d)}} \\ &-\delta_s(t,d)\frac{u_s}{u_v}F_{sv}L_s^*(\cdot,d)-\frac{1}{u_v}F_{mv}L_m(\cdot,d) \end{aligned} \tag{2.14}$$

Here $Q_k(t,d) = \sum_{i=1}^{\mu_k N} f_i(t,d)$*,* $\delta_k(t,d) = \mathbb{1}(L_k^*(t,d) > 0)$*,* $F_{kj} = (p_k/p_j)^{2/\alpha}$*. If the term inside the square root of (2.13) is negative, or if (2.13) or (2.14) is negative, we set the corresponding intensity to zero.*

Proof. The result follows from the KKT (Karush-Kuhn-Tucker) optimality conditions. □

Demand functions. Theorem 2.4 allows us to quantify how the optimal demands L_s^* and L_v^* vary with the user intensities L_u. Surprisingly, we observe that the caching intensity L_k^* for $k \in \{s, v\}$ is a *square root function of* $L_u/\left\|L_u\right\|_1$, and thus increases at a slower rate than the tier connectivity, which is linear in $L_u(t,d)/\left\|L_u\right\|_1$ (Theorem 2.1), as user requests become more concentrated in a single time and region (t,d) ($L_u(t,d)/\left\|L_u\right\|_1$ increases). The optimal small cell demand $L_s^*(t,d)$ depends on the aggregated user intensity $\sum_t L_u(t,d)$, as we would expect given small cells' lack of mobility, while the vehicle demand $L_v^*(t,d)$ only depends on the users at other times through the small cell demand $L_s^*(t,d)$. In fact, *small cells substitute for vehicles:* as $L_s^*(t,d)$ increases, the vehicular demand $L_v^*(t,d)$ decreases linearly.

The marginal rate of substitution, $\delta_s(t,d)\frac{u_s}{u_v}F_{sv}$, is independent of $L_u(t,d)/\left\|L_u\right\|_1$ but depends on the ratios of the small cell and vehicle transmit powers p_s/p_v and cache capacities u_s/u_v.

We next investigate the dependence on the prices P_k. The *optimal demand* L_k^* *decays on the order of* $\sqrt{P_k}$ *for both tiers* $k \in \{s, v\}$. As with the user intensity, the optimal small cell demand $L_s^*(\cdot, d)$ is a function of the aggregated price $\sum_t P_s(t,d)$ at each location d. However, the optimal vehicle demand $L_v^*(t,d)$ is only affected by the prices P_v at other times through the small cell demand $L_s^*(t,d)$. We can thus find the cross-time price sensitivity of this demand:

Corollary 2.1 (Cross-Time Sensitivity for Vehicles' Prices)**.** *For cache hit utilities, if the price* $P_v(t_1,d)$ *increases for some* t_1*, the optimal demands of vehicles in other time slots* $L_v^*(t_2,d) : t_2 \neq t_1$ *either decrease or stay unchanged.*

Intuitively, if $P_v(t_1,d)$ increases, small cells become more attractive than vehicles in region d. Thus, $L_s^*(t_1,d)$ either increases or stays the same. Since small cells are fixed over time, the number of small cells at (t_2,d) also increases or stays the same, prompting a decrease in $L_v^*(t_2,d)$.

Economic viability. We next use Theorem 4 to find conditions under which vehicular caching is viable, i.e., there is positive demand for some time slots and regions.

Corollary 2.2 (Economic Viability of Vehicular Caching)**.** *Assume* $\mu_v \leq \mu_s$*. If at the optimal point all intensities* L_v^* *and* L_s^* *are positive, the following relation holds*

$$\sum_t P_s(t,d) > \frac{u_s}{u_v}\left(\frac{p_s}{p_v}\right)^{\frac{2}{\alpha}} \sum_t P_v(t,d) \tag{2.15}$$

In words, the region-accumulative price $\sum_t P_s(t,d)$ of small cells is lower bounded by that of vehicles, scaled by the ratio of transmit power and caching capacities. We can thus interpret (2.15) as a lower bound on the small cell prices for which all intensities are positive, i.e., the operator uses both vehicles and small cells. If the small cells are sufficiently less expensive or have sufficiently higher capacity ($u_s \gg u_v$), the operator will not utilize the vehicular caches at all.

Profit maximization with general utility functions. We now turn to the general utility function U_0. In this case, we solve Problem 2.1 with a quadratic transform [85] as it does not have a closed-form solution. From Theorem 2.1, we can write

$$\mathbb{E}[C_{k,i}(t,d)] = \frac{W_{k,i}(t,d)}{Z_{k,i}(t,d)} \tag{2.16}$$

where $W_{k,i}$ and $Z_{k,i}$ are affine functions of L_v and L_s.

Thus, defining $\alpha_v = \alpha_s = a, \alpha_m = b$, (2.2) can be written as

$$U_0 = \sum_{t,d,i,k} \alpha_k \frac{L_u(t,d)}{\|L_u\|_1} f_i(t,d) \frac{W_{k,i}(t,d)}{Z_{k,i}(t,d)} \tag{2.17}$$

We then introduce the auxiliary utility function (2.17):

$$\begin{aligned} V(y, L_v, L_s) = \sum_{t,d,i,k} \alpha_k \frac{L_u(t,d)}{\|L_u\|_1} f_i(t,d) \times \\ \left(2y_{k,i}(t,d)\sqrt{W_{k,i}(t,d)} - y_{k,i}^2(t,d) Z_{k,i}(t,d)\right), \end{aligned} \tag{2.18}$$

Substituting (2.17) with (2.18) in Problem 2.1, we find the equivalent optimization problem:

Problem 2.3 (Transformed Profit Maximization)**.**

$$\begin{aligned} \underset{y,L_v,L_s}{\mathit{maximize}} \quad & \gamma V(y, L_v, L_s) - tr(P_v^T L_v) - tr(P_s^T L_s) \\ \mathit{subject\ to} \quad & L_v(t,d) \geq 0, L_s(t_1,d) = L_s(t_2,d) \geq 0 \end{aligned}$$

For general content placement policies, the dimension of y is in the order of $\Theta(TDN)$. However, if the value of $H_{k,i}$ is fixed for most contents (e.g. the general placement policy (2.1)), the summation over i can be greatly simplified, and the number of auxiliary variables y can be reduced to $\Theta(TD)$.

Theorem 2.5 (Equivalence of Problems 2.1 and 2.3)**.** *The variable (L_v^*, L_s^*) maximizes Problem 2.1 if and only if (L_v^*, L_s^*) together with some y^* maximizes Problem 2.3. Furthermore, Problems 2.1 and 2.3 have the same objective value at the optimal point.*

Proof. See Appendix A of [85]. □

The transformed Problem 2.3 is not necessarily a convex problem. However, it is easy to see that this problem is convex with respect to (L_v, L_s) when the value of y

is fixed. On the other hand, given (L_v, L_s), the optimal y can be written as

$$y_{k,i}(t,d) = \sqrt{W_{k,i}(t,d)}/Z_{k,i}(t,d) \tag{2.19}$$

This useful property allows the use of a block coordinate ascent algorithm to find the optimum by iteratively solving for the optimal (L_v, L_s) given y and the optimal y given (L_v, L_s). The convergence of this procedure to optimality follows from that of the block coordinate ascent method [92].

2.6.2 Solutions for Utility Maximization (Problem 2.2)

As in Theorem 2.4, we can also derive a closed-form solution to Problem 2.2 for the cache hit utility (2.8). Based on that we can conclude Corollary 2.3 and 2.4 below.

Given a limited budget, the maximum utility in the competitive market is bound to be no worse than that of single-product markets as the latter are special cases of the two-product optimization. We first consider the **value of competition** with infinite budgets, when the prices of the small cells and vehicles do not matter:

Corollary 2.3 (Upper Bound of Cache Hit Utilities)**.** *With infinite budget, the maximum utility for the vehicle only and the small cell only market using utility (2.8) can not exceed*

$$U \leq \sum_{t,d} \frac{L_u(t,d)}{\|L_u\|_1} Q_k(t,d), k \in \{v, s\} \tag{2.20}$$

Here $Q_k(t,d)$ is given in Theorem 2.4, and the upper bound for the competitive market is the same as the small cell market, as we assume $\mu_v \leq \mu_s$ in the definition of Q_k.

Corollary 2.3 suggests that *as the network operator's budget increases, the value of competition approaches zero*: the utilities under the competitive and small cell only markets have the same upper bounds. With infinite budget, either vehicles or small cells can achieve their optimal utilities as their costs are insignificant. We verify this result in Section 2.7: as the budget increases, the competitive market achieves nearly the same utility as the small cell only market. Corollary 2.3 also implies that the value of the competitive market is bounded away from 1, i.e., perfect cache hits, due to the limited capacities of the caches. However, despite this lack of competition, both products are purchased at large budgets:

Corollary 2.4 (Linear Increase of Optimal Demands)**.** *Under cache hit utilities*

(2.8), for a sufficiently large value of the budget $P_0 \gg 0$, the optimal intensities L_v^* and L_s^* increase linearly with respect to P_0 at every (t, d).

Thus, for sufficiently large budgets, there is effectively no change in the competition between vehicles and small cells: a budget increase yields constant marginal increases in the demands for both types of caching. This result is consistent with the linear substitution of small cells for vehicles from Theorem 2.4. Together with Corollary 2.3, it implies that the marginal value of additional caching intensities is eventually zero, which we verify in Section 2.7.

To solve Problem 2.2 for general utility functions, we can still apply the quadratic transform (2.18) and update rule (2.19) as for Problem 2.1. We can then use a block coordinate ascent algorithm to find the optimal solution.

2.7 Numerical Simulations

We next verify Section 2.6's results numerically. We first consider the optimal demands in Section 2.7.1 and the value of competition in Section 2.7.2. We then investigate the impact of variations in the system model in Section 2.7.3.

We use the SF311 database [22] to simulate the temporal and spatial dynamics of user requests. The database records the location and time for each mobile 311 service request in San Francisco, California. Figure 2.2 shows the hourly distribution of user requests for the 30 most popular neighborhoods, averaged over all weekdays in 2017. To facilitate the visualization of our results, we focus on two representative regions: the business/office region Financial District (referred to as R1), and the residential/leisure region Showplace Square (R2). Both regions have about 5000 daily requests and similar land area. We scale their distributions such that the highest hourly intensity equals 50 per squared kilometer (Figure 2.2). We define the business hour (T1) as 10:00 to 16:00, and the off hour (T2) as 18:00 to 24:00. For both regions, we average the intensities within each time slot, thus yielding four intensity bins (km^{-2}): 39.5 (T1+R1), 11.8 (T1+R2), 27.4 (T2+R1), 25.0 (T2+R2). For both regions, the intensity of macro cells is 0.3 km^{-2}. Unless otherwise noted, we set $P_s \equiv 18, P_v(1, \cdot) = 2, P_v(2, \cdot) = 5$.

We set the transmit power (Watt) as $p_m = 40$, $p_s = 6.3$, and $p_v = 1.0$ [28]. We let the SIR threshold $\tau = 1(0 \text{ dB})$, and the path-loss coefficient $\alpha = 4$. We consider

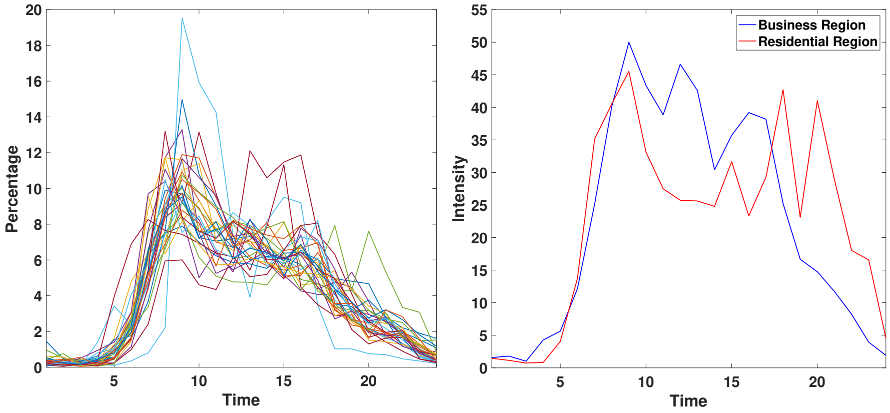


Figure 2.2: The distribution of user requests in a typical day for the top 30 neighborhoods and the typical "business" and "residential" regions

a content catalog with a total of $N = 10000$ chunks, and let $N_v = 30, N_s = 100$. For simplicity, we assume users' preferences for content chunks do not change for different time slots and regions, and follow the Zipf distribution with shape coefficient equal to 1.2 [17]. For the delivery rate model, let $r_b = 0.5 \log(1 + \tau)$, i.e. the delivery rate when the cache is missed is half that for the cache hit rate. The marginal utility value (i.e. γ) is set to be 500 for the connectivity utility U_1 in (2.6) and the cache hit utility U_2 in (2.8), and 1000 for the delivery rate utility U_3 in (2.12).

2.7.1 Demands of Caching

Hourly demand of vehicular caching. An advantage of vehicular caching over small cells is its ability to better respond to the change of user intensities: at any time, more vehicles may be dispatched to regions that currently have higher user intensities. Thus, the hourly dynamics of the optimal vehicle demands are expected to match the patterns of user requests shown in Figure 2.2.

In Figure 2.3, we show a heat map of the optimal vehicle intensities for the top 30 neighborhoods under the cache hit utility U_2. In Figure 2.4, we compare the vehicle intensities in the business and residential regions (R1 and R2) for all three utilities U_1, U_2 and U_3. Comparing Figures 2.2 and 2.3, more vehicles are used in the daytime due to higher request intensities, and the vehicle intensity also adapts to variation over different locations. Under all utility functions, the temporal patterns of vehicle demands in R1 and R2 (Figure 2.4) closely track those of the user requests

(Figure 2.2).

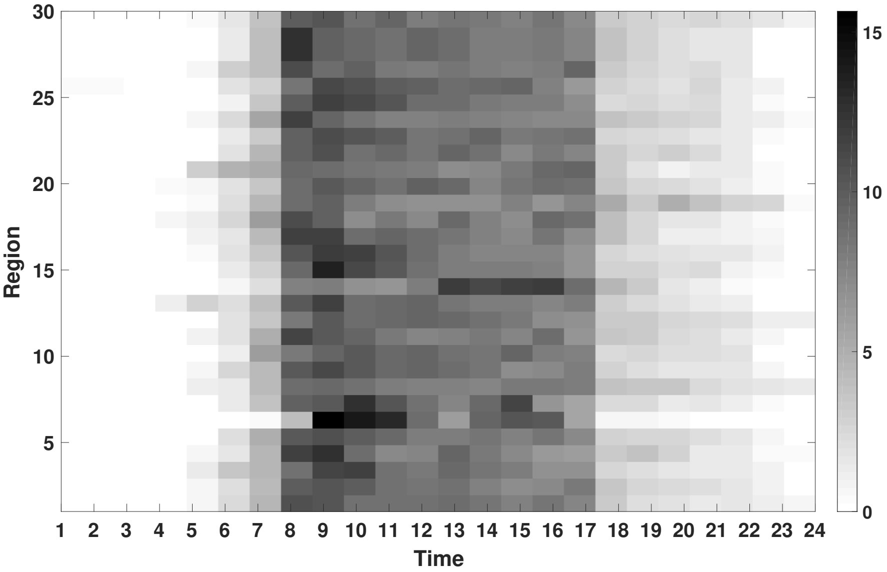


Figure 2.3: Heat map of the optimal vehicle demands with the cache hit utility U_2

Viability and accumulative demand. The accumulative demand for each caching tier is the sum of the temporally averaged demands across regions. In our setting, it reflects the operators' daily demands for both caching tiers. Thus, vehicular caching is economically viable only if its accumulative demand is positive. We consider the economic viability for two time slots (T1, T2) and two regions (R1, R2) for ease of visualization. For clarity we assume the prices do not change over time and regions (we explore price variation below), and set $P_s \equiv 18$. We then vary the price of vehicles P_v, and find the corresponding demands as per Theorems 2.4 and 2.5.

Figure 2.5 shows the accumulative demand curves for the three utility functions using the greedy content placement policy and the spectrum sharing scheme. Intensities are aggregated over regions and averaged across time. Vehicles exhibit zero demand when the price exceeds a certain threshold. The dash-dot line at 7.17 represents the lower bound of this threshold given by Corollary 2.2. The accumulative demand for vehicles decreases with a diminishing elasticity as the price goes up at approximately a rate of $(\sqrt{P_k})^{-1}$, as expected from Theorem 2.4. Each of the three utility functions considered exhibit qualitatively similar demand curves. For all three utility functions, the accumulative demand for vehicles drops to zero above

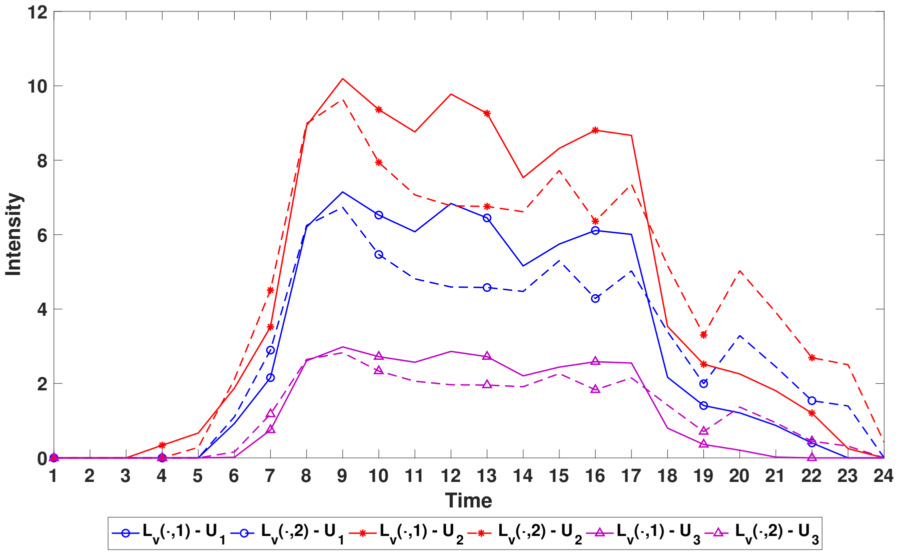


Figure 2.4: The optimal vehicle demands over 24 hours in the business region (R1) and the residential region (R2)

a threshold price around 8: vehicular caching is economically viable only if its price is below this threshold. Corollary 2.2 gives an lower bound of 7.17 on this threshold price, which is very close to the actual threshold.

Spillover effects on time and region. The demands of vehicles and small cells at different times are highly correlated. From Corollary 2.1, if the price of vehicles increases in some (t_1, d_1), operators will decrease their demands for vehicles $L_v(t_1, d_1)$ and subscribe to more small cells, increasing $L_s(t_1, d_1)$. Since the small cell intensities are fixed over time, $L_s(t_2, d_1)$ also increases, and the operators subscribe to fewer vehicles $L_v(t_2, d_1)$. To reveal this spillover effect, we fix $P_s \equiv 18, P_v(2, \cdot) = 5$, and vary vehicles' price in business hours $P_v(1, \cdot)$, assuming prices do not change across regions. Figure 2.6 shows the resulting change of demands with greedy placement and spectrum sharing. X-axes are intensities, Y-axes are price of vehicular caching in business hours $P_v(1, \cdot)$. As $P_v(1, \cdot)$ increases, demand for vehicles (small cells) decreases (increases) at all time slots and regions. The change of vehicle demand in off hours is consistent with Corollary 2.1: it first decreases, then stays unchanged. However, the decrease is most pronounced during business hours (T1), when the price itself changes.

Effect of user intensities. The dynamics of users' requests can significantly affect the economic performance of vehicular caching. Intuitively, vehicles are more

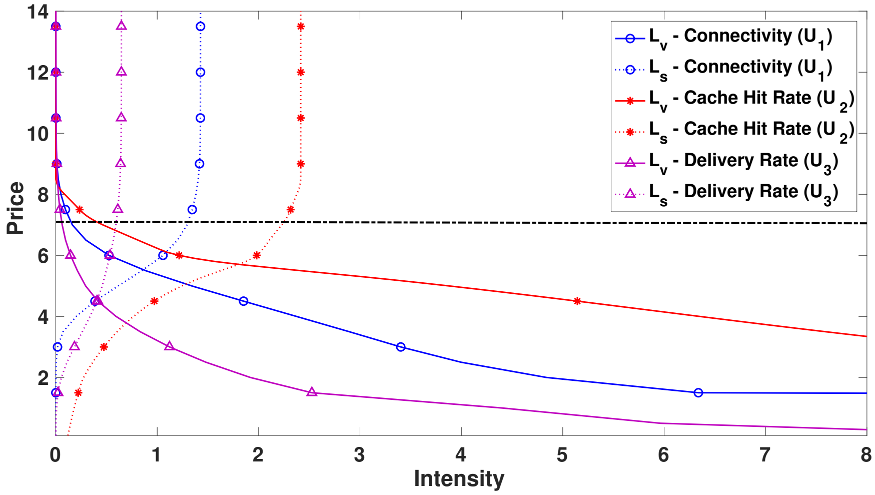


Figure 2.5: Accumulative demand curves for vehicular caching as the vehicle price $P_v(\cdot, \cdot)$ varies

attractive if user intensities are more divergent across time and regions. To illustrate this effect, we set the user intensities as $L_u(1,1) = L_u(2,2) = 30 + d_u$, $L_u(1,2) = L_u(2,1) = 30 - d_u$. As d_u grows higher, the distribution of users becomes more uneven. We use the cache hit utility U_2.

Figure 2.7 shows our results. The X axes represent d_u (user divergence), and Y axes represent caching demands. For both regions, as L_u diverges, $L_s^*(\cdot, d)$ first remains constant, then decreases. This is consistent with Theorem 2.4: since $\sum_t L_u(t,d) \equiv 60$ in our setting, L_s^* changes only when some vehicle demands L_v^* become zero. On the other hand, L_v^* changes significantly at a rate of $\sqrt{d_u}$. When $d_u = 30$, i.e., there are no users in the business (residential) region during off (business) hours, $L_v^*(1,2) = L_v^*(2,1) = 0$. When $d_u = 0$ and the user intensities are uniform, vehicle demand is still positive due to the lower prices P_v compared to small cell prices P_s.

2.7.2 Value of Competition

As discussed in Section 2.3, we evaluate the value of competition by comparing the solutions to Problem 2.2 for the competitive and single-product markets.

Figure 2.8 shows our achieved utilities with greedy placement policy and spectrum sharing. The X-axes represent budgets, and Y-axes represent utilities. Both plots follow the same pattern. At first, when the budget is small, vehicular caching obtains

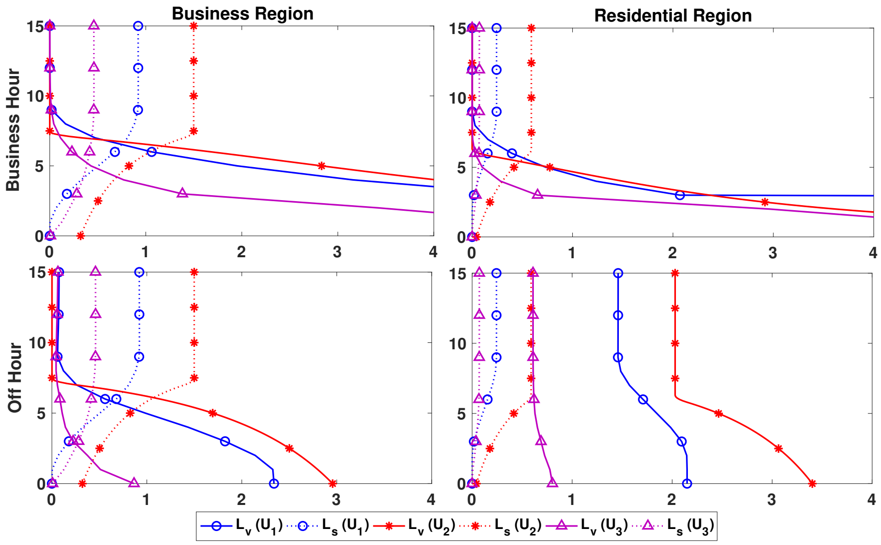


Figure 2.6: Spillover effect caused by change of vehicles' prices

a higher utility level than small cells, and the optimal choice in the competitive market is to subscribe only to vehicles. In this phase, the competitive curve overlaps with the vehicle curve; the value of competition is thus zero. As the budget increases, we start to observe a positive value of competition - the competitive market achieves a higher utility level than both single-product markets. When the budget is large enough, small cells start to outperform vehicles, and the vehicle curve and the small cell curves intersect. In the meantime, the utility gain from competition continues to rise. Finally, when the budget goes to infinity, the competitive curve converges to the small cell curve, i.e., the value of competition gradually drops to zero. For the cache hit utility, the utility bounds can be calculated by Corollary 2.3, which states that the curves will eventually converge at $Q_v = 0.639$ for the vehicle only market and $Q_s = 0.751$ for both the competitive market and the small cell only market.

These phase transitions can be explained as follows. When the budget is small, few caching points are deployed in the network. Thus, even requests for popular content chunks can not be served by the cache. In this situation, cached chunks in vehicles can be better utilized due to their mobility, making vehicular caching more valuable. When the budget increases, however, the increase of utility is constrained by the finite vehicle capacity N_v. As a result, the total utility is driven by the larger capacity N_s of the small cells.

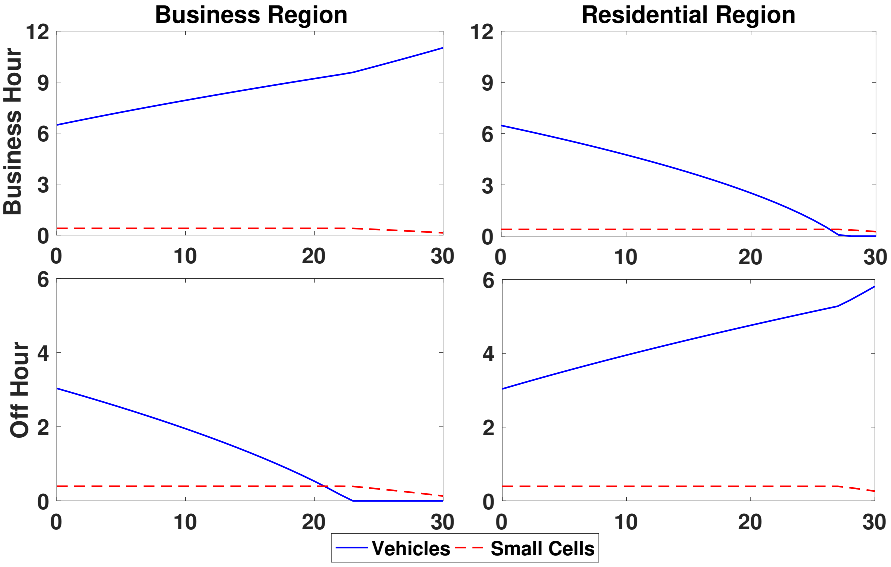


Figure 2.7: Evolution of caching demands as user intensities diverge

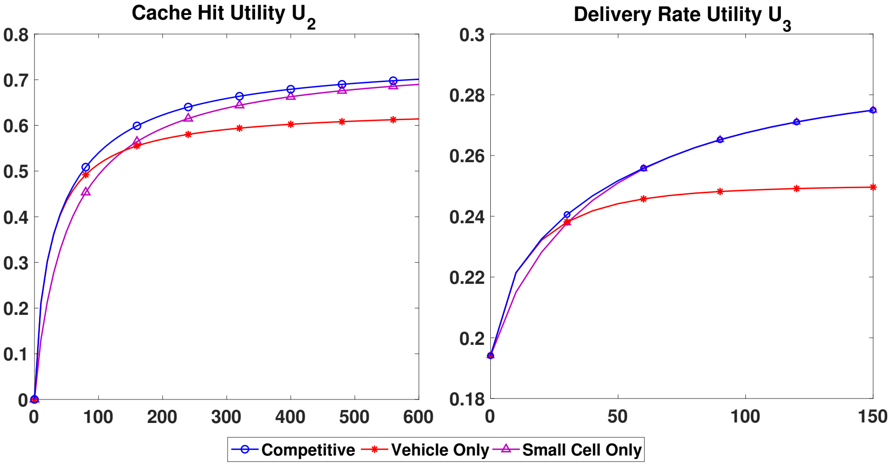


Figure 2.8: Comparison of maximum utilities with fixed budget

2.7.3 Variations in the System Model

We finally test the robustness of our results to inaccuracies in our cache intensity models. To do so, we add bias to the optimal intensities $L_k^*, k \in \{v, s\}$, and recalculate the resulting operator profit. The bias is drawn from a Gaussian distribution with zero mean, and the standard deviation equals $L_k^* \times \sigma$ for some σ. In the case an intensity value is negative, we set it to zero. We run a total of 500 trials, gradually increasing σ from 0 to 0.3. In Figure 2.9, the grey lines represent results of 500 trials.

The black lines are envelopes, showing little variation in the achieved profit. The profit drops by at most 10 (5.8%) for $\sigma < 0.2$, implying our model is robust to small intensity deviations.

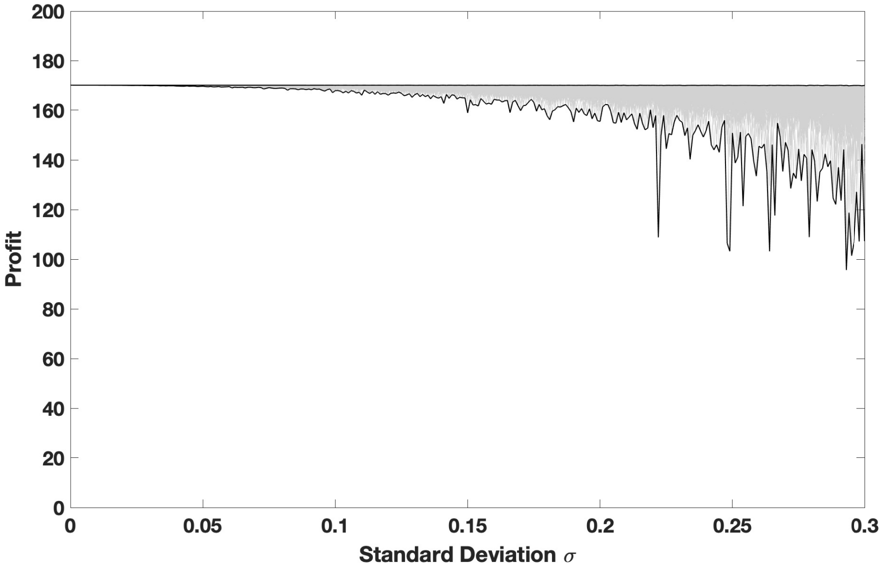


Figure 2.9: The change of profits under U_2 when the optimal intensity is offset by a zero mean bias term

2.8 Summary

In this chapter, we have proposed a framework to evaluate the economic value of vehicular caching. A system model is developed to capture the economic and physical dynamics of caching tiers. We then use this model to derive the utility functions for different network operators, including both CPs and ISPs. The value of vehicular caching is quantified through its viability, demands, and the gain of competition. These research questions are answered by solving two optimization problems, for which we have provided both analytical and algorithmic solutions. We find that vehicular caching does not add value if its price exceeds a finite threshold or the operator has infinite budget. Simulation results verify the economic value of vehicular caching, and illustrate the dynamics of the system.

Chapter 3

Topology-Aware Offloading of Machine Learning Data

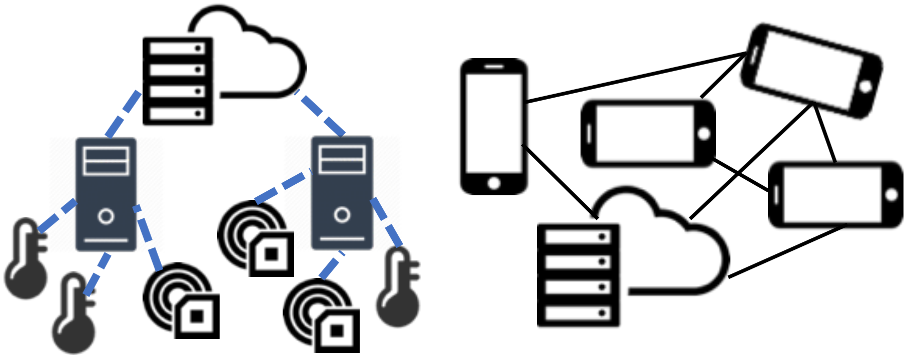
3.1 Introduction

Initial efforts in decentralizing ML have focused on decomposing model parameter updates over several nodes, typically managed by a centralized serving entity [61, 69]. Most of these methods, however, implicitly assume idealized network topologies where node and link properties are homogeneous. Edge environments, by contrast, are characterized by devices' heterogeneity both in available compute resources and in connectivity with each other, e.g., due to power constraints or mobility.

Figure 3.1 illustrates two example edge topologies. In the hierarchical case, less powerful devices are connected to more powerful ones, while for the social network, connections are denser and devices tend to be similar. A central question that arises, then, in adapting ML methodologies to these environments is: *How should each edge device contribute to the ML training and inference?* We answer this question by developing a methodology for *optimizing the distribution of processing across a network of edge devices.*

3.1.1 Machine Learning in Edge Environments

ML models are generally trained by iterating over a dataset to estimate parameter values (e.g., weights in a neural network) that best "fit" the empirical data. We face two major challenges in adapting such training to edge environments: *(i) heterogeneity*



(a) Hierarchical (b) Social network

Figure 3.1: Cartoon illustrations of two example topologies for edge computing

in devices' compute resources and *(ii) constraints on devices' abilities to communicate with each other*. We outline these characteristics in some key applications below:

Internet-connected vehicles can collaboratively learn about their environment [21], e.g., by combining their data with that of road sensors to infer current road or traffic conditions. Since sensors have less computing capabilities than vehicles, they will likely offload their data to vehicles or roadside units for processing. This offloading must adapt as vehicles move and their connectivity with (stationary) sensors changes.

Augmented reality (AR) uses ML algorithms for e.g., image recognition [14] to overlay digital content onto users' views of an environment. A network of AR-enabled devices can distributedly train ML models, but may exhibit significant heterogeneity: they can range from generic smartphones to AR-specific headsets, with different battery levels. As the users move, connectivity between devices will also change.

Industrial IoT. 5G networks will allow sensors that power control loops within factory production lines to communicate across the factory floor [4, 21], in turn enabling distributed ML algorithms to use this data for, e.g., predicting production delays. Our approach can determine which controllers should process data from which sensors: this depends on sensor-controller connectivities, which may vary with factory activity.

3.1.2 Outline and Summary of Contributions

We first differentiate our approach from related literature in Section 3.2. To the best of our knowledge, we are the first to optimize the distribution of ML data processing (i.e., training) tasks across edge nodes, leading to several contributions:

Formulating the task distribution problem (Section 3.3). In deciding which devices should process which datapoints, our formulation accounts for resource limitations and model accuracy. While ideally more of the data would be processed at devices with more computing resources, sending data samples to such devices may overburden the network. Moreover, processing too many data samples can incur large processing costs relative to the gain in model accuracy. We derive new bounds (Theorem 3.1) on the model accuracy when data can be moved between devices, and show that the optimal task distribution problem can be formulated as a convex optimization that can be solved rapidly even for large networks.

Characterizing the optimal task distribution (Section 3.4). Solving the optimization problem formulated in Section 3.3 requires specifying parameters that may not be known in advance, e.g., the number of datapoints that each device can process in a single timeslot. We analyze the expected deviations from our assumptions in Section 3.3 to derive guidelines on how these parameters should be set (Theorem 3.2). We then derive the optimal task distributions for typical edge network topologies (Theorems 3.3 and 3.4) and use them to estimate the value (i.e., reduction in processing costs) of allowing devices to move processing tasks to other devices (Theorems 3.5 and 3.6).

Experimental validation (Section 3.5). We train classification models on the MNIST dataset to validate our algorithms. We use data traces from a Raspberry PI testbed to emulate network delays and compute resource availability. Our proposed algorithm nearly halves the computing overhead yet achieves an accuracy comparable to centralized model training.

3.2 Related Work

We contextualize our approach within prior results on (i) federated learning algorithms and (ii) methods for offloading ML tasks from mobile devices to edge servers. In classical distributed learning, multiple "workers" each compute a gradient or parameter value on their own local data. These results are aggregated at a central server, and updated parameter values are sent back to the workers to begin another round of local computations. In the federated learning framework, devices instead perform a *series* of local updates between aggregations [31, 46, 61]. Such a framework preserves user privacy by keeping data at local devices [87] and reduces the communication between devices and the central server. However, these works do not optimally distribute parameter computations between devices, and do not consider the compute-communication tradeoffs inherent in edge scenarios.

Offloading computations from constrained mobile devices to nearby edge servers when there is a high-bandwidth connection between them will intuitively improve system performance, and has been shown to significantly accelerate training of a linear regression model [13] and inference on a neural network model [71]. Other works have considered splitting deep neural network layers between edge devices and an edge server for faster inference [40, 90]. We instead consider generic ML frameworks, and additionally provide theoretical performance bounds not found in these prior works.

3.3 Model and Optimization Formulation

In this section, we define models for edge networks (Section 3.3.1) and ML training (Section 3.3.2), and then formulate the ML task distribution optimization problem (Section 3.3.3).

3.3.1 Edge Computing System Model

Edge computing nodes. We consider a set V of n devices, an aggregation server s, and discrete time intervals $t = 1, \ldots, T$. Each device, e.g., a sensor or smartphone, can both collect data and process it to contribute to an ML task. The server s aggregates the results of each device's local analysis, as will be explained in Section 3.3.2. Both the length and number of time intervals may depend on the specific ML application. In each interval t, we suppose a subset of devices $V(t)$, indexed by i, is active (i.e., available to collect and/or process data). For simplicity of notation, we omit i's dependence on t.

Data collection and processing. We use $D_i(t)$ to denote the set of data collected by device $i \in V(t)$ at time t; $d \in D_i(t)$ denotes each datapoint. (We may have $D_i(t) = 0$ if a device does not collect data.) $G_i(t)$, by contrast, denotes the set of datapoints *processed* by each device at time t; our optimization in Section 3.3.3 relates

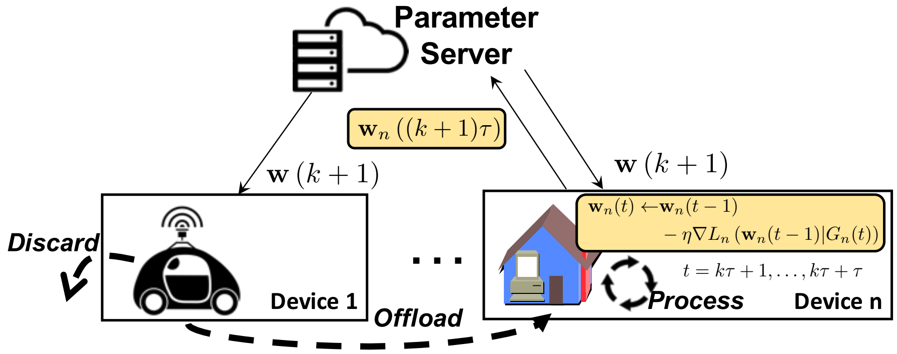

Parameter Server
$\mathbf{w}_n\left((k+1)\tau\right)$
$\mathbf{w}\left(k+1\right)$
$\mathbf{w}\left(k+1\right)$
$\mathbf{w}_n(t) \leftarrow \mathbf{w}_n(t-1) - \eta \nabla L_n\left(\mathbf{w}_n(t-1)|G_n(t)\right)$
Discard
Device 1
. . .
$t = k\tau+1,\ldots,k\tau+\tau$
Process
Device n
Offload

Figure 3.2: Federated learning updates between aggregations k and $k+1$

$G_i(t)$ to the datasets $D_i(t)$. In conventional learning frameworks, $D_i(t) = G_i(t)$, as all devices process the data they collect [99]; separating these variables is one of our main contributions. We suppose that each device i can process up to $C_i(t)$ datapoints at each time t, incurring a cost of $c_i(t)$ for each point. This cost and capacity may for instance represent the battery level; devices with low battery will have lower capacities $C_i(t)$ and higher costs $c_i(t)$.

Edge network connectivity. The devices V are connected to each other via a set E of directed links, with $(i, j) \in E$ denoting a link from device i to j, and $E(t) \subseteq E$ denoting the set of functioning links at time t. The overall system then can be described as a directed graph $(\{s, V\}, E)$ with vertices V representing the devices and edges E the links between them. We suppose that $(\{s, V(t)\}, E(t))$ is fully connected at each time t and that links between devices are single-hop, i.e., devices do not use each other as relays except possibly to the server. Note that the scenarios outlined in Section 3.1.1 each possess such an architecture: in smart factories, for example, a subset of the floor sensors connect to each controller. Each link $(i, j) \in E(t)$ is characterized by a capacity $C_{ij}(t)$, i.e., the maximum datapoints it can transfer, and a "cost of connectivity" $c_{ij}(t)$. This cost may reflect network conditions (e.g., signal strengths, congestion) or a desire for privacy, and will be higher if sending from i to j is less desirable at t.

Data structure. Each datapoint d can be represented as (x_d, y_d), where x_d is an attribute/feature vector and y_d is an associated label for model learning. We use $D_V = \cup_{i,t} D_i(t)$ to denote the full set of datapoints collected by all devices over all time. For simplicity, we follow prior work [104, 113] and model the data collection at device i as points being selected uniformly at random from a (usually unknown) distribution $\mathcal{D}_i$. In practice the $\mathcal{D}_i$ can evolve over time, but we assume this evolution is slow compared to the horizon T. We use $\mathcal{D} = \cup_i \mathcal{D}_i$ to denote the global distribution induced by these $\mathcal{D}_i$. Note this assumption implies the relationship between x_d and y_d is temporally invariant, which is common in ML applications, e.g., image recognition from cameras at fixed locations or AR users with random mobility patterns. We will use such an image dataset for evaluation in Section 3.5.

3.3.2 Machine Learning Model

Our goal is to learn a parameterized model that outputs y_d given the input feature vector x_d. We use the vector w to denote the set of model parameters, whose values are chosen so as to minimize a loss function $L(w|\mathcal{D})$ that depends on the ML model (e.g., squared error for linear regression, cross-entropy loss for multi-class classifiers [64]). Since the overall distributions $\mathcal{D}_i$ are unknown, instead of minimizing $L(w|\mathcal{D})$ we minimize the empirical loss function, as commonly done:

$$\underset{w}{\text{minimize}} \;\; L(w|D_V) = \frac{\sum_{t=1}^{T} \sum_{i \in V(t)} \sum_{d \in G_i(t)} l(w, x_d, y_d)}{|D_V|} \tag{3.1}$$

where $l(w, x_d, y_d)$ is the error for datapoint d, and $|D_V|$ is the number of datapoints. Note that the function l may include regularization terms that aim to prevent model overfitting [61].

Edge computing allows (3.1) to be solved distributedly: instead of computing the solution at the server s, we can use computations at each device i. Below, we follow the commonly used federated averaging framework [99] in specifying these local computations and global aggregation, illustrated by device n in Figure 3.2. Device 1 discards all of its data or offloads it to device n, which computes τ gradient updates on its local data. The final parameter values are averaged at the parameter server, with the result sent back to the devices to begin a new iteration. To avoid excessive re-optimization at each device, we suppose that they execute the same local updating algorithm regardless of $G_i(t)$. We adjust the server averaging to account for the amount of data each device processes.

Local loss minimization. In order to solve (3.1) in a distributed manner, we first decompose the empirical loss function into a weighted sum of local loss functions

$$L_i(w_i|G_i) = \frac{\sum_{t=1}^{T} \sum_{d \in G_i(t)} l(w, x_d, y_d)}{|G_i|} \tag{3.2}$$

where $G_i \equiv \cup_{t \leq T} G_i(t)$ denotes the set of datapoints processed by device i over all times. The global loss in (3.1) is then equal to $L(w|D_V) = \sum_i L_i(w|G_i)\,|G_i|\,/\,|D_V|$ if $\cup_i G_i = D_V$, i.e., if all datapoints $d \in D_V$ are eventually processed at some device.

Due to the inherent complexity of most ML models, loss functions such as (3.2) are typically minimized using gradient descent techniques [61]. Specifically, the devices update their local parameter estimates at t according to

$$w_i(t) = w_i(t-1) - \eta(t)\nabla L_i(w_i(t-1)|G_i(t)) \tag{3.3}$$

where $\eta(t) > 0$ is the step size, which often decreases with t, and $\nabla L_i(w_i(t-1)|G_i(t)) = \sum_{d \in G_i(t)} \nabla l(w_i(t-1), x_d, y_d)/|G_i(t)|$ is the gradient with respect to w of the average loss of points in the current dataset $G_i(t)$ at the parameter value $w_i(t-1)$. We define the loss only on the current dataset $G_i(t)$ since future data in G_i has not yet been revealed; since we assume each node's data is IID over time, we can view $L_i(w_i(t-1)|G_i(t))$ as approximating the local loss $L_i(w_i|G_i)$. One can then interpret the computational cost $c_i(t)$ of processing datapoint d as the cost of computing the gradient $\nabla l(w_i(t-1), x_d, y_d)$. If the local data distributions $\mathcal{D}_i$ are all the same, then all datapoints across devices are IID samples of this distribution, and this process is similar to stochastic gradient descent with batch size $|G_i(t)|$.

Aggregation and synchronization. The aggregation server s will periodically average the local estimates $w_i(t)$ from the devices and synchronize the devices with a global update. Formally, the kth aggregation is computed as

$$w(k) = \frac{\sum_i H_i(k\tau) \cdot w_i(k\tau)}{\sum_i H_i(k\tau)} \tag{3.4}$$

where τ is the fixed aggregation period and $H_i(k\tau) = \sum_{t=(k-1)\tau+1}^{k\tau} |G_i(t)|$ is the number of datapoints node i processed since the last aggregation. Thus, the update is a weighted average factoring in the sample size H_i on which each $w_i(t)$ is based. Once this is computed, each device's local estimate is synchronized, i.e., $w_i(t) \leftarrow w(t/\tau)$. A lower value of τ will result in faster convergence of w, while a higher value requires less network resources. Prior work [99] has considered how to optimize τ, so we assume it is pre-determined in our formulation, analyzing its effect experimentally in Section 3.5.

3.3.3 Optimization Model for Data Processing Tasks

We now consider the choice of $G_i(t)$, which implicitly defines the ML tasks to be executed by device i at time t, i.e., processing all datapoints in $G_i(t)$. There are two possible reasons $G_i(t) \neq D_i(t)$: first, device i may *offload* some of its collected data to another device j or vice versa, e.g., if i does not have enough capacity ($D_i(t) \geq C_i(t)$) or possibly if j has lower computing costs ($c_j(t) \leq c_i(t)$). Second, device i may *discard* data if processing it does not reduce the empirical loss (3.1) by much. In Figure 3.2, device 1 offloads or discards all of its data. We collectively refer to discarding and offloading as *data movement*. We do not include the cost of communicating parameter updates to/from the server in our model; unless a device processes no data, the number of updates stays constant.

Data movement model. We define $s_{ij}(t) \in [0,1]$ as the fraction of data collected at device i that is offloaded to device $j \neq i$ at time t. Thus, at time t, device i offloads $D_i(t)s_{ij}(t)$ amount of data to j.[1] Similarly, $s_{ii}(t)$ will denote the fraction of data collected at time t that device i also processes at time t. We suppose that as long as $D_i(t)s_{ij}(t) \leq C_{ij}(t)$, the capacity of the link between i and $j \neq i$, then all offloaded data will reach j within one time interval and be processed at device j in time interval $t+1$. Since devices must have a link between them to offload data, $s_{ij}(t) = 0$ if $(i,j) \notin E(t)$. We also define $r_i(t) \in [0,1]$ as the fraction of data collected by device i at time t that will be discarded. In doing so, we assume that device j will not discard data that has been offloaded to it by others, since that has already incurred an offloading cost $D_i(t)s_{ij}(t)c_{ij}(t)$. The amount of data collected by device i at time t and discarded is then $D_i(t)r_i(t)$, and the amount of data processed by each device i at time t is

$$G_i(t) = s_{ii}(t)D_i(t) + \sum_{j \neq i} s_{ji}(t-1)D_j(t-1).$$

---

[1] For notational convenience, $D_i(t)$ here refers to the length $|D_i(t)|$, and similarly for $G_i(t)$ in this section. The context makes the distinction clear.

In defining the variables $s_{ij}(t)$ and $r_i(t)$, we have implicitly specified the constraint $r_i(t) + \sum_j s_{ij}(t) = 1$: all data collected by device i at time t must either be processed by device i at this time, offloaded to another device j, or discarded. We assume that devices will not store data for future processing, which would add another cost component to the model.

Data movement optimization. We formulate the following cost minimization problem for determining the data movement variables $s_{ij}(t)$ and $r_i(t)$ over the time period:

$$\begin{aligned}
\underset{s_{ij}(t), r_i(t)}{\text{minimize}} \quad & \sum_{t=1}^{T} \Bigg(\sum_i G_i(t) c_i(t) + \sum_{(i,j)\in E(t)} D_i(t) s_{ij}(t) c_{ij}(t) \\
& \qquad + \sum_i f_i(t) L\left(w_i(t) | D_V\right) \Bigg) && (3.5) \\
\text{subject to} \quad & G_i(t) = s_{ii}(t) D_i(t) + \sum_{j \neq i} s_{ji}(t-1) D_j(t-1) && (3.6) \\
& s_{ij}(t) = 0, \ \ (i,j) \notin E(t), j \neq i && (3.7) \\
& r_i(t) + \sum_j s_{ij}(t) = 1, \ \ s_{ij}(t), r_i(t) \geq 0 && (3.8) \\
& G_i(t) \leq C_i(t), \ \ s_{ij}(t) D_i(t) \leq C_{ij}(t) && (3.9)
\end{aligned}$$

Constraints (3.6–3.8) were introduced above and ensure that the solution is feasible. The capacity constraints in (3.9) ensure that the amounts of data transferred and processed are within link and node capacities, respectively. The three terms in the objective (3.5) correspond to the processing, offloading, and error costs, respectively, as we detail below.

(i) Processing, $G_i(t)c_i(t)$**:** This is the computing cost associated with processing $G_i(t)$ of data at node i at time t.

(ii) Offloading, $D_i(t)s_{ij}(t)c_{ij}(t)$**:** This is the communication cost incurred from node i offloading data to j.

(iii) Error, $f_i(t)L\left(w_i(t)|D_V\right)$**:** This cost quantifies the impact of the data movement on the error of the model at each device i; note that since $w_i(t)$ is computed as in (3.3), it is an implicit function of $G_i(t)$, the data processed at device i. We include the error from *each* device i's local model at each time t, instead of simply considering the error of the final model, since devices may need to make use of their local models as they are updated (e.g., if aggregations are infrequent due to resource constraints [99]). Discarding data clearly increases the loss, since less data is used to train the ML model; offloading may also skew the local model if it is updated over a small number of samples $G_i(t)$. We can, however, upper bound the loss function $L(w_i(t))$ regardless of the data movement:

Theorem 3.1 (Upper bound on the local loss). *If $L_i(w)$ is convex, ρ-Lipschitz, and β-smooth, if $\eta \leq \frac{1}{\beta}$, and if $L(w(T)) - L(w^\star) \geq \epsilon$ for some $\epsilon > 0$, then after K aggregations with a period τ and defining the constant $\delta_i \geq ||\nabla L_i(w) - \nabla L(w)||$,*

$$L(w_i(t)) - L(w^\star) \leq \epsilon_0 + \rho g_i(t - K\tau), \tag{3.10}$$

where $g_i(x) = \frac{\delta_i}{\beta}((\eta\beta + 1)^x - 1)$ implying $g_i(t - K\tau)$ is decreasing in K, and ϵ_0 is given by

$$\frac{1}{t\omega\eta(2-\beta\eta)} + \sqrt{\frac{1}{t^2\omega^2\eta^2(2-\beta\eta)^2} + \frac{Kh(\tau) + g_i(t-K\tau)}{t\omega\eta(1-\beta\eta/2)}}.$$

Proof. Define $v_k(t)$ for $t \in \{(k-1)\tau, ..., k\tau\}$ as the parameters under centralized gradient descent updates, $\theta_k(t) = L(v_k(t)) - L(w^\star)$, $K = \lfloor t/\tau \rfloor$, and assume $\theta_k(k\tau) \geq \epsilon$ [99]. After lower-bounding $\frac{1}{\theta_{K+1}(t)} - \frac{1}{\theta_1(0)}$ and $\frac{1}{L(w_i(t)) - L(w^\star)} - \frac{1}{\theta_{K+1}(t)}$, we can upper-bound $L(w_i(t)) - L(w^\star)$ as

$$\left(t\omega\eta\big(1 - \frac{\beta\eta}{2}\big) - \frac{\rho}{\epsilon^2}\big(Kh(\tau) + g_i(t - K\tau)\big)\right)^{-1} = y(\epsilon).$$

Let ϵ_0 be the positive root of $y(\epsilon) = \epsilon$, which is easy to check exists. The result follows since either $\min_{k \leq K} L(v_k(k\tau)) - L(w^\star) \leq \epsilon_0$ or $L(w_i(t)) - L(w^\star) \leq \epsilon_0$; both imply (3.10). □

In Section 3.4, we will consider how to use Theorem 3.1's result to find tractable forms of the loss expression that allow us to solve (3.5–3.9) efficiently and accurately. Indeed, without perfect information on the device costs, capacities, and error statistics over the time period T, it is not possible to solve (3.5–3.9) exactly. We will experimentally validate our proposed methods for estimating these parameters in Section 3.5.

3.4 Optimization Model Analysis

We turn now to a theoretical analysis of the data movement optimization problem (3.5–3.9). We discuss the choice of error and capacity parameters under various assumptions (Section 3.4.1), and then characterize the optimal solution for the ML use cases outlined in Section 3.1 (Section 3.4.2).

3.4.1 Choosing Cost Parameters and Capacities

We may not be able to reliably estimate the cost parameters $c_{ij}(t)$, $c_i(t)$, and $f_i(t)$ or capacities $C_i(t)$ and $C_{ij}(t)$ in real time. Mis-estimations are likely in highly dynamic scenarios of mobile devices, since the costs $c_{ij}(t)$ of offloading data depend on network conditions at the current device locations. Mobile devices are also prone to occasional processing delays called "straggler effects" [66], which can be modeled as variations in their capacities. The error cost, on the other hand, will change over time as the model parameters move towards convergence. Here, we propose and analyze parameter selection methods.

Choosing capacities. Over-estimating the device processing capacities will force some data processing to be deferred until future time periods, which may cause a cascade of processing delays. Under- or over-estimations of the link capacities will have similar effects. Here, we formalize guidelines for choosing the capacities in (3.9)'s constraints so as to limit delays due to over-estimation. As commonly done [32], we assume that processing times on device stragglers follow an exponential distribution $\exp(\mu)$ for parameter μ.

For device capacities, we obtain the following result:

Theorem 3.2 (Data processing time with compute stragglers)**.** *Suppose that the service time of processing a datapoint at node i follows $\exp(\mu_i)$, and that $c_{ij}(t)$, $c_i(t)$, $C_i(t)$ are time invariant. We can ensure the average waiting time of a datapoint to be processed is lower than a given threshold σ by setting the device capacity parameter C_i such that $\phi(C_i) = \sigma\mu_i/(1+\sigma\mu_i)$, where $\phi(C_i)$ is the smallest solution to the equation $\phi = \exp\left(-\mu_i(1-\phi)/C_i\right)$, which is an increasing function of C_i.*

Proof. The processing at node i follows a D/M/1 queue with an arrival rate $G_i(t) \leq C_i$, and the result follows from the average waiting time in such a queue. □

For instance, $\sigma = 1$ guarantees an average processing time of less than one time slot, as assumed in Section 3.3's model. Thus, Theorem 3.2 shows that we can still (probabilistically) bound the data processing time when stragglers are present.

Network link congestion in transferring data may also delay its processing. Such delays can be handled by carefully choosing the network capacity analogously to Theorem 3.2's method.

Choosing error expressions. As shown in Theorem 3.1, we can bound the local loss at time t in terms of a gradient divergence constant $\delta_i \geq \|\nabla L_i(w) - \nabla L(w)\|$. The following in turn provides an upper bound for δ_i in terms of $G_i(t)$:

Lemma 3.1 (IID error convergence)**.** *Suppose that the distributions $\mathcal{D}_i$ are identical so that $\mathcal{D}_i = \mathcal{D}$, and that $\mathcal{D}$ has finite second and third moments. Then there exists a constant $\gamma > 0$ that does not depend on the value of $G_i(t)$ such that*

$$\delta_i \equiv \|\nabla L_i\left(w|G_i(t)\right) - \nabla L(w)\| \leq \frac{\gamma}{\sqrt{G_i(t)}} \tag{3.11}$$

Proof. The result follows immediately from the central limit theorem upon viewing each $\nabla l(w, x_d, y_d)$ as a sample from the distribution induced by $\nabla l(w|\mathcal{D})$. □

Assuming IID data distributions is reasonable for many ML applications: for instance, AR users might follow statistically similar mobility patterns throughout an area, and sensors on a factory floor might monitor machines with similar failure patterns. Combining the result in Lemma 3.1 with Theorem 3.1, we find that $L(w_i(t)) - L(w^\star) \propto \sqrt{G_i(t)^{-1}}$. Thus, it is possible to take the error cost $f_i(t)L(w_i(t)|D_V)$ in (3.5) as $f_i(t)\sqrt{G_i(t)^{-1}}$ with $f_i(t)$ scaling the error importance; $f_i(t)$ may decrease over time as the model approaches convergence.

Since $\sqrt{G_i(t)^{-1}}$ is a convex function of $G_i(t)$, with this choice of error cost, (3.5–3.9) becomes a convex optimization problem and can be solved relatively easily in theory. When the number of variables is large, however – e.g., if the number of devices $n > 100$ with $T > 100$ time periods – standard interior point solvers will be prohibitively slow [100]. In such cases, we may wish to approximate the error term with a linear function and leverage faster linear optimization techniques, i.e., to take the error cost as $f_i(t)G_i(t)$ but with $f_i(t) < 0$ since the error decreases when $G_i(t)$ increases. If we neglect the offloaded data $s_{ij}(t)$ for $j \neq i$, we can rewrite this cost as $f_i(t)D_i(t)[1 - r_i(t)]$, which is equivalent to minimizing $-f_i(t)r_i(t)$. The error costs from the offloaded data can then be folded into the communication costs $c_{ij}(t)$, and we can approximate the error cost as $-f_i(t)D_i(t)r_i(t)$. Intuitively, discarding data implies a less accurate model.

3.4.2 Optimal Task Distributions

Given a set of cost and capacity parameters for the optimization (3.5–3.9), we now characterize the optimal solutions in a range of practical cases. In particular, when we consider a linear error term $f_i(t)r_i(t)D_i(t)$, we have the following result:

Theorem 3.3 (Unconstrained resource solution)**.** *Suppose that* $C_i(t) \geq D_i(t) + \sum_{j\in\mathcal{N}_i(t-1)} D_j(t-1)$ *for each device* i*, i.e., its compute capacity always exceeds the data it collects as well as any data offloaded to it by its neighbors* $\mathcal{N}_i(t-1) = \{j : (j,i) \in E(t-1)\}$*. Then if the error cost is linearly approximated as* $f_i(t)D_i(t)r_i(t)$*, the optimal* $s^*_{ij}(t)$ *and* $r^*_i(t)$ *will each be* 0 *or* 1*, with the conditions for* 1 *at node* i *being:*

$$\begin{cases} s^*_{ik}(t) = 1 & \textit{if } c_{ik}(t) + c_k(t+1) \leq \min\{f_i(t), c_i(t)\} \\ s^*_{ii}(t) = 1 & \textit{if } c_i(t) \leq \min\{f_i(t), c_{ik}(t) + c_k(t+1)\} \\ r^*_i(t) = 1 & \textit{if } f_i(t) \leq \min\{c_i(t), c_{ik}(t) + c_k(t+1)\} \end{cases} \tag{3.12}$$

where $k = \underset{j\neq i,(i,j)\in E(t)}{\arg\min} \{c_{ij}(t) + c_j(t+1)\}$.

Proof. Since $r_i(t) + \sum_j s_{ij}(t) = 1$ in (3.8), each datapoint in $D_i(t)$ is either discarded, offloaded, or processed at i. It is optimal to choose the option with least marginal cost. □

This theorem implies that in the absence of resource constraints, all data a device generates will either be processed, offloaded to the lowest cost neighbor, or discarded. Below, we examine implications of this result for typical edge topologies.

Edge use cases. Table 3.1 summarizes the topologies of the edge applications from Section 3.1. Networks in smart factories have fairly static topologies, since they are deployed in controlled indoor settings. They also exhibit a hierarchical structure, with less powerful devices connected to more powerful ones in a tree-like manner, as shown in Figure 3.1. Connected vehicles have a similar hierarchical structure, with sensors and vehicles connected to more powerful edge servers, but their architectures are more dynamic as vehicles are moving. Similarly, AR applications

Table 3.1: Dominant characteristics of the four use cases

Use case	Topology	Dynamics
Smart factories [21]	Hierarchical	Fairly static
Connected vehicles [21]	Hierarchical	Rapid changes
Augmented reality [14]	Hierarchical, heterogeneous	Rapid changes possible
Privacy-sensitive [66, 73]	Social network	Fairly static

feature (possibly heterogeneous) mobile AR headsets connected to powerful edge servers. Applications that involve privacy-sensitive data may have very different, non-hierarchical topologies as the links between devices are based on trust, i.e., comfort in sharing private information. Since social relationships generally change slowly compared to ML model training, these topologies are relatively static.

Hierarchical topologies. In hierarchical scenarios, more powerful edge servers will likely always have sufficient capacity $C_i(t)$ to handle all offloaded data (satisfying the assumptions in Theorem 3.3), and they will likely experience lower computing costs $c_i(t)$ as well compared to other devices. Theorem 3.3 indicates that, with a linear error cost, sensors would then offload their data to the edge servers, unless the costs of offloading the data exceed the difference in computing costs. In Section 3.5, we will see on our Raspberry PI testbed that the network cost does indeed sometimes exceed the gain in computing cost from offloading to more powerful devices.

When the cost of discarding data is nonlinear, the optimal solution is less intuitive: it may be optimal to discard fractions of data if the reduction in error is not worth the additional cost of processing. Formally, in the case of a hierarchical topology, we have the following result:

Theorem 3.4 (Data movement with nonlinear error costs)**.** *Suppose that* n *devices with identical, constant processing costs* $c_j(t) = c$ *and data generation rates* $D_j(t) = D$ *can offload their data to a single edge server, indexed as* $n+1$*. Further assume that there are no resource constraints,* $c > c_{n+1}$*, the costs* $c_{ij}(t) = c_t$ *of transmitting to the server are identical and constant, and the discard cost is given by* $f_i(t)L(w_i(t)) = \gamma/\sqrt{G_i(t)}$ *as in Lemma 3.1. Then, letting* s *denote the fraction of data offloaded, for* D *sufficiently large, the optimal amount of data discarded as a function of* s *is*

$$r^*(s) = 1 - \frac{1}{D}\left(\frac{\gamma}{2c}\right)^{\frac{2}{3}} - s. \tag{3.13}$$

Given the optimal r^*, the optimal s^* is given by

$$s^* = \frac{1}{nD}\left(\frac{\gamma}{2(c_{n+1}+c_t)}\right)^{\frac{2}{3}} \tag{3.14}$$

Proof. In the hierarchical scenario, the cost objective (3.5) can be rewritten as

$$n(1-r-s)Dc + nsD(c_{n+1}+c_t) + \frac{n\gamma}{\sqrt{(1-r-s)D}} + \frac{\gamma}{\sqrt{snD}}.$$

Taking the partial derivatives with respect to r and s, and noting that a large D forces $r, s \in [0,1]$ gives the result. □

Intuitively, increases in the costs c_{n+1}, c_t, data D, or devices n should cause the amount of data offloaded to decrease and the amount discarded to increase. D has the strongest effect: with more data at each node, the fraction needed for processing at the server and devices to manage the discard cost decreases inversely. The only cost that impacts r but not s is c, as it is a device parameter that does not involve the network.

Social network topologies. When device networks are larger and have more complex topologies, we can extrapolate from Theorem 3.3's characterization of individual device behavior to understand data movement in the network as a whole. Consider, for instance, a social network of users in which edges are defined by willingness to share data (Figure 3.1b). We can find the probability that a given device offloads data, which allows us to determine the cost savings from offloading:

Theorem 3.5 (Value of offloading)**.** *Suppose the fraction of devices with k neighbors equals $N(k)$, e.g., in a scale-free network $N(k) = \Gamma k^{1-\gamma}$ for some constant Γ and $\gamma \in (2,3)$. Suppose $c_i \sim U(0,C)$ and $c_{ij} = 0$, where $U(a,b)$ is the uniform distribution between a and b, with no discarding. Then the average cost savings, compared to no offloading, equals*

$$\sum_{k=1}^{n} N(k)\left(\frac{C}{2} - \frac{C(-1)^k}{k+2} - \sum_{l=0}^{k-1}\binom{k}{l}\frac{C(-1)^l(k+3)}{(l+2)(l+3)}\right). \tag{3.15}$$

The processing cost model may for instance represent device battery levels drawn uniformly at random from 0 (full charge) to C (low charge). This result establishes that the reduction in cost from enabling device offloading in such scenarios is approximately linear in C: as the range of computing costs increases in a scale-free topology, there is greater benefit from offloading, since devices are more likely to find a neighbor with lower cost. The expected reduction, however, may be less than the average computing cost $C/2$ even in the absence of link costs, as offloading data to another device does not entirely eliminate the computing cost.

We finally consider the case in which resource constraints are present, e.g., for less powerful edge devices. We can find the expected number of devices with tight resource constraints:

Theorem 3.6 (Probability of resource constraint violation)**.** *Let $N(k)$ be the number of devices with k neighbors, and for each device i with k neighbors, let $p_k(n)$ be the probability that its neighbor j has n neighbors. Also let $\tilde{C}$ denote the distribution of resource capacities, assumed to be IID across devices, and let $D_i(t) = D$ be constant. Then if devices offload as in Theorem 3.3, the expected number of devices whose capacity constraints are violated is*

$$\int_{\tilde{C}(x)} \left(\sum_{k=1}^{N} N(k) \mathbb{P} \left[1 - P_o(k) + k \sum_{n=1}^{N} \left(\frac{P_o(n) p_k(n)}{n} \right) \geq \frac{x}{D} \right] \right), \tag{3.16}$$

with $P_o(k)$ defined as the probability a device with k neighbors offloads its data.

Proof. This follows from Theorem 3.3 and determining the expected amount of data that will be processed at a node with k neighbors when offloading is enabled. □

Theorem 3.6 allows us to quantify the complexity of solving the data movement optimization problem when resource constraints are in effect. We observe that it depends on not just the resource constraints, but also on the distribution of computing costs (through $P_o(k)$), since these costs influence the probability devices will want to offload in the first place.

3.5 Experimental Evaluation

In this section, we experimentally evaluate our methodology in several scenarios. We discuss the general setup in Section 3.5.1, and present our results in Sections 3.5.2 to 3.5.4.

Table 3.2: Our method, centralized learning, and federated learning show comparable accuracies on the test dataset

Method	MLP	CNN
Centralized learning	92.58%	98.39%
Federated learning	90.33%	96.81%
Network-aware learning	90.34%	96.49%

3.5.1 Experimental Setup

Data samples and ML models. We consider the MNIST dataset [49], which contains 70K images of hand-written digits. We use 60K of them as the training dataset D_V, and the remainder as our test set. Each node i is randomly allocated $|D_i(t)| \sim U(0, |D_V|/(nT))$ datapoints from D_V, without replacement. We train a multilayer perceptron (MLP) and a convolutional neural network (CNN) for image recognition on MNIST, with cross entropy loss [47] as the loss function $L(w|D_V)$, and constant learning rate $\eta(t) = 0.01$ in (3.3). Unless otherwise stated, results are reported for CNN, an aggregation period $\tau = 10$ in (3.4), and $T = 100$ time intervals.

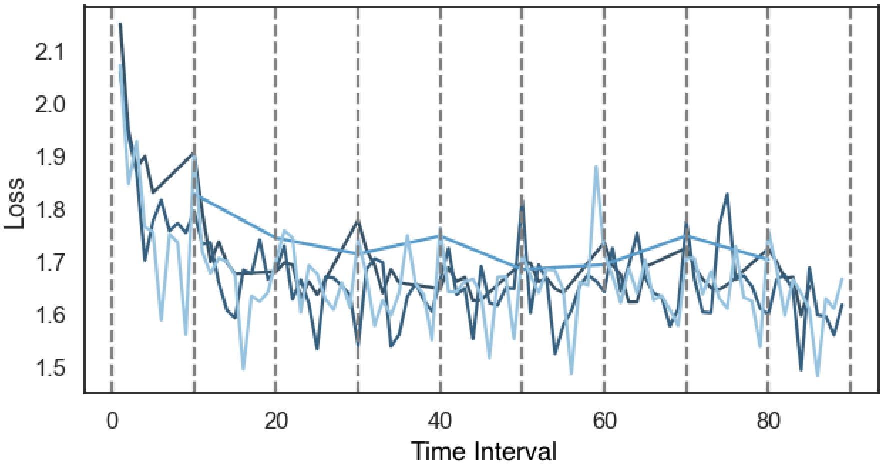


Figure 3.3: Training loss over time for each device observed with our method

Cost and capacity parameters. In the default case, we simulate $n = 10$ edge devices and one server. When imposed, the capacity constraints $C_i(t)$ and $C_{ij}(t)$ are drawn from $U(0, 4 \max D_i(t))$ and $U(0, 4 \max D_i(t)/n)$ respectively. Due to randomization, results are averaged over several iterations.

We consider both real and synthetic cost parameters. For the synthetic parameters,

Table 3.3: Network costs and model accuracies obtained in four different settings

Setting	**Accuracy**	**Cost**				
		Process	**Transfer**	**Discard**	**Total**	**Unit**
A	82.89%	2078	0	0	2078	0.48
B	90.24%	367	620	166	1153	0.26
C	89.24%	326	606	67	999	0.23
D	82.10%	225	835	14	1074	0.24

$c_{ij}(t) \sim U(0, 1/n)$ and $c_i(t) \sim U(0, 1)$. The real-world measurements come from our testbed consisting of six Raspberry PIs using AWS DynamoDB as a cloud-based parameter server. Three PIs each collect data and transmit it over bluetooth to another "gateway" PI. The three gateway nodes receive this data and can either perform a local gradient update (processing time recorded as $c_i(t)$) or upload the data to DynamoDB (communication time recorded as $c_{ij}(t)$) to be processed there. We collect 100 sets of processing and communication measurements while training a two-layer fully connected neural network, with devices communicating over 2.4 GHz WiFi or LTE cellular.

Centralized and federated learning. To see whether our method compromises learning accuracy in considering network costs as additional objectives, we compare against a baseline of centralized ML training where all data is processed at a single device (server). Additionally, we consider the standard implementation of federated learning where there is no data offloading or discarding (i.e., $G_i(t) = D_i(t)$) and aggregations occur after every time interval (i.e., $\tau = 1$) [99].

Perfect information vs. estimation. As discussed in Section 3.4.1, solving (3.5-3.9) in practice requires estimating the cost and capacity parameters over the time horizon T. To do this, we divide T into L intervals $T_1, ..., T_L$, and in each interval l, we use the time-averaged observations of $D_i(t)$, $p_i(t)$, $c_{ij}(t)$, and $C_i(t)$ over T_{l-1} to compute the optimal data movement. The resulting $s^\star_{ij}(t)$ and $r^\star_i(t)$ for $t \in T_l$ are then be used by device i to transfer data in T_l. This "imperfect information" scheme will be compared with the ideal case of perfect information.

3.5.2 Efficacy of Network-Aware Learning

Our first experiments seek to establish the overall efficacy of our method. Here, we use the synthetic cost parameters with a fully connected topology $E(t) = \{(i, j) : i \neq j\}$; qualitatively similar results were observed with other configurations.

Model accuracy. Table 3.2 compares the average accuracy on the testing datasets obtained by the MLP and CNN models trained with centralized, federated, and network-aware learning, where the centralized and federated cases are run until convergence. Our method does well: it matches federated learning's accuracy, and does only 2% worse than centralized learning. We also plot the training loss $L_i(w_i(t))$ across devices over time in Figure 3.3, confirming that while some devices have higher local losses, all tend to decrease over time.

Cost reduction with imperfect information. Table 3.3 compares the costs incurred and model accuracy for four settings: (A) offloading and discarding disabled, (B) network-aware learning with perfect information and no capacity constraints, (C) network-aware learning with imperfect information and no capacity constraints, and (D) network-aware learning with imperfect information and capacity constraints. Each cost is aggregated over nodes/links and time periods. The unit cost normalizes the total cost over the amount of data generated in that setting, to account for the $D_i(t)$ varying randomly.

Comparing (A) and (B), we see that allowing data transfers substantially reduces the unit cost– by 46%. Surprisingly, the accuracy also improves from (A) to (B) despite some datapoints being discarded: when offloading without capacity constraints, nodes with lower processing costs tend to receive significantly more data, giving them a larger sample size $G_i(t)$ for gradient updates, and thus more accurate parameter estimates that are more heavily weighted in the aggregations. Even with imperfect information on the parameters in (C), we observe only minor changes in cost or accuracy, highlighting the robustness of the model to estimation errors similar to our observations from the analytics results in Section 3.4.1. The result for (D) furthers the point on solution accuracy: when devices have strict capacity constraints, their gradient updates are based on fewer samples, and each node's $L_i(w_i(t))$ will tend to have larger errors. However, the total cost of (D) is still significantly lower than (A) with a comparable accuracy.

3.5.3 Effect of Network System Parameters

Our next experiments on synthetic cost data assess the impact of n, the number of nodes, and the aggregation period, τ, on a fully connected topology. Then, we consider the effect of connectivity when nodes are connected in a random graph with

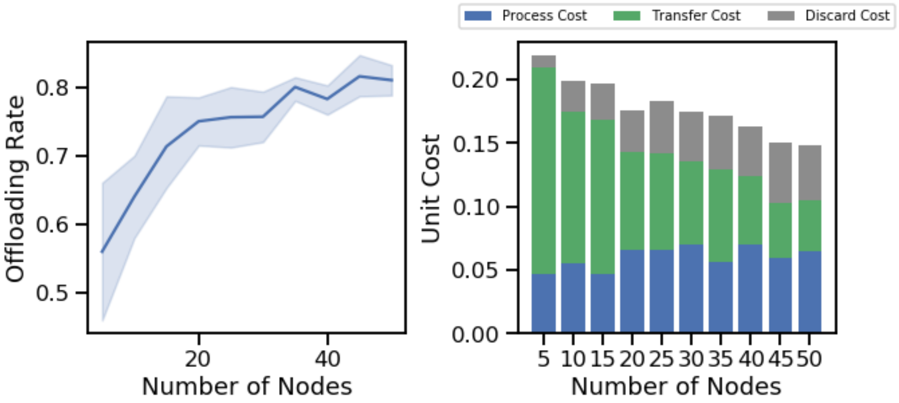


Figure 3.4: Impact of the number of nodes n on (a) the average offloading rate and (b) the different cost components

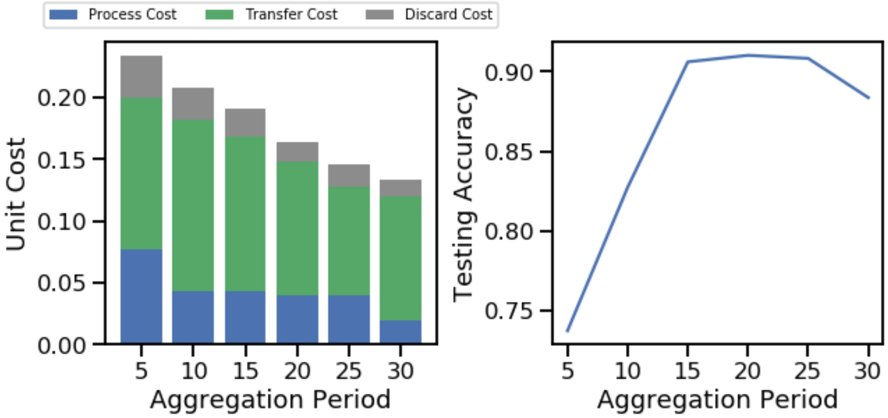


Figure 3.5: Impact of the global aggregation period τ on (a) the cost components and (b) the learning accuracy

probability ρ, i.e., $P[(i,j) \in E(t), j \neq i] = \rho$.

Varying number of nodes n. Figure 3.4 shows the (a) offloading rate and (b) unit costs as the network size changes. Overall, we see that our method scales well with the number of nodes, as the unit (i.e., per datapoint) cost steadily decreases with n. The transferring (i.e., offloading) cost drives this improvement, as the processing and discarding costs actually increase: with more nodes, the network leverages lower cost opportunities for offloading – hence the increase in offloading rate to over 80% when $n = 50$ – as long as it is less than any increase in processing cost, consistent with Theorem 3.3. The learning accuracy increases slightly from 88 to 92% as n increases.

Varying aggregation period τ. Figure 3.5 shows the (a) unit costs and (b) learning accuracy as the period of global aggregation is varied. A larger value of τ yields smaller total unit cost, but only improves learning accuracy until roughly $\tau = 20$: below this, the nodes are not processing enough datapoints in-between aggregations for the local parameters to become steady, and above it, the local models are not synchronized frequently enough. We also note that the cost breakdown exhibits a different trend than in Figure 3.4, as the transmission cost stays relatively constant while the processing and discarding costs decrease: with a longer τ, data can be discarded with a lower cost towards the end of each period.

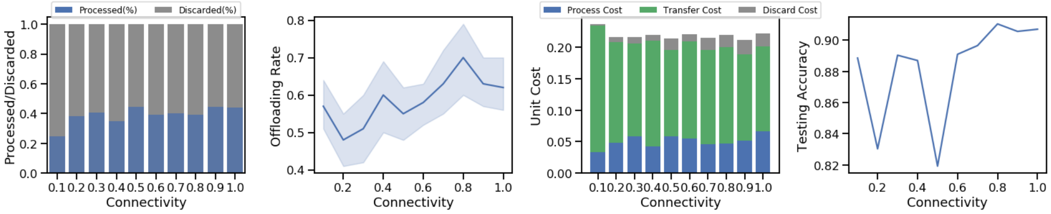


Figure 3.6: Impact of the network connectivity ρ on data movement and costs in network-aware learning

Varying network connectivity ρ. In Figure 3.6, we plot the (a) fraction of processed data, (b) offloading rate, (c) costs, and (d) learning accuracy as the network connectivity changes. We see that our network-aware learning methodology is reasonably robust to ρ in terms of cost (for $\rho > 0.1$): when connectivity drops, there are less low-cost opportunities for offloading, so nodes do not transfer as much data. The learning accuracy, by contrast, tends to benefit from higher connectivity (for $\rho > 0.5$), since low cost nodes can process more of the data, similar to setting (B) in Table 3.3. Interestingly, while the percentage of data discarded increases slightly with lower ρ, the discard cost has the opposite trend: when nodes do not have the option of low cost transfers, their $G_i(t)$ become larger for smaller t, causing the discard cost to drop more rapidly as in Lemma 3.1.

3.5.4 Effect of Edge Topology

Finally, we evaluate our network-aware learning methodology on different edge computing topologies. We consider three different graph structures: hierarchical and social network topologies as studied in Section 3.4.2, and a fully connected topology

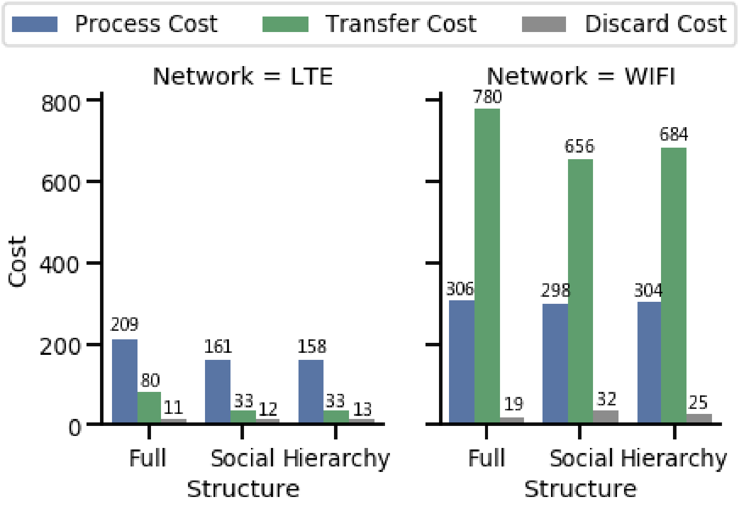


Figure 3.7: Cost components for different topologies running network-aware learning on (a) LTE and (b) WiFi network media

for completeness. The social network is generated as a Watts-Strogatz small world graph [19] with each node connected to $n/5$ of its neighbors, and the hierarchical network connects each of the $n/3$ nodes with lowest processing costs to two of the $2n/3$ remaining nodes as leaves. We use the cost parameters collected from our Raspberry PI testbed, which provides two different wireless network media: LTE and WiFi.

The resulting costs are shown in Figure 3.7. For LTE, we see that processing dominates the cost distribution for all three topologies, while for WiFi, the transfer costs are the largest: WiFi has less interference mitigation techniques than cellular, so in the presence of several devices we expect its links to exhibit larger delays. This is also likely the reason why the total cost is substantially higher under WiFi, since devices have less lower cost options for offloading. This point is further consistent with the fact that in the case of LTE, the social and hierarchical topologies exhibit virtually the same costs: despite guaranteed connections to higher powered nodes up the hierarchy, the social network likely contains low cost links to these nodes anyway. In the case of WiFi, by contrast, the hierarchical topology has a noticeably higher offloading cost, and lower discarding cost, than the social topology, since the transmissions to high-power nodes occurs over high cost links.

3.6 Summary

We consider distributing ML training tasks over devices in a edge computing network. We develop a framework to optimize the distribution of training tasks, taking into account both physical computing and communication costs and the error achieved by the models at each device. We derive new error bounds when devices can transfer their local data processing to each other, and theoretically bound the impact of these transfers on the cost and accuracy of the model training. Through experimentation with a popular machine learning task, we show that our network-aware scheme significantly reduces the cost of model training while achieving comparable accuracy.

Our framework and analysis point to several possible extensions. First, while we do not observe significant heterogeneity in compute times on our wireless testbed, in general edge devices may experience compute straggling and failures, which might benefit from more sophisticated offloading mechanisms. Second, predicting devices' mobility patterns and the resulting network connectivity can likely further optimize the data offloading. Finally, for some applications, one might wish to learn individual models for each device, which would introduce new performance tradeoffs in offloading data processing.

Chapter 4

Enhancing Federated Learning with Intelligent Client Recruitment

4.1 Introduction

Federated learning allows users to contribute the power of their data to train machine learning models without sharing any raw records. However, ensuring the good performance of federated learning requires overcoming additional challenges that do not present in centralized learning. Specifically, the challenges originate in the heterogeneous distributions of data at different users (statistical challenge), and in the resource-constrained nature of the edge computing system (system challenge) [89]. Much recent work aims to address them by optimizing the learning algorithm given a set of participating clients, i.e., clients that use their data to contribute model updates to the training process. However, these works neglect a complementary question: Before running the federated learning algorithm, *how should the operator recruit participating clients so as to optimize the performance of its federated learning algorithm?* In this chapter, we show that a good client recruitment is essential to overcoming federated learning's statistical and system challenges, complementing algorithmic innovations like carefully selecting or scheduling model updates from a given set of clients.

Client recruitment formalizes the relationship between the two market players in typical **commercial applications** of federated learning: the operators and the users. Operators are typically companies who hope to create or improve their AI products utilizing their users' data. For example, Google has utilized data from Android users to train a query suggestion model for their keyboard application [105]. Federated learning operators are responsible for setting up a coordinator that collects iterative updates from the participating clients. However, most federated learning algorithms require upfront commitments from users to compute local updates, which may consume limited battery, and send them to the coordinator, which may reveal private information, to the training on demand. Such upfront commitments are generally required to ensure convergence of the training algorithm [61]. To compensate for these commitments, recruited users may need incentives from the operator to participate in the training, as proposed in [50] to compensate privacy losses. Such compensation, however, introduces a new challenge not commonly considered in federated learning: limiting the recruitment cost.

We define **client recruitment** as the preliminary step of federated learning, in which the coordinator determines the set of candidate clients with which it will train a model. When the recruitment is finalized, we will have determined the quality and quantity of the training data, the number and types of local devices, and the associated cost of compensating users. A good client recruitment is fundamental to the successful execution of federated learning and complements the more commonly considered *client selection* [67], in which the coordinator chooses which of the recruited clients will be asked to provide updates in each training iteration. Since federated learning requires upfront client commitments as discussed, recruiting clients is necessary to ensure the success of subsequent client selection. Indeed, careful recruitment will reduce the number of clients required to make training commitments (by almost 5x in our experiments), improving federated learning's overall efficiency and impact on user privacy. However, client recruitment raises **new challenges** compared to client selection: unlike client selection algorithms that utilize information revealed during the training, we must base recruitment decisions on information known before training begins, which requires more detailed statistical analysis of the anticipated learning accuracy. Complicating this problem further, client recruitment additionally decides the model's *generalizability* and *representativeness*, which client selection cannot control.

Our contributions are: 1) We construct a comprehensive system model to *quantify the quality measures of federated learning*, including not only the training loss, but also the model's generalizability, the reliability and completion time of training, and the operating expense (Section 4.4); 2) We formulate an *optimization framework* to capture the complex tradeoffs in client recruitment (Section 4.5); 3) We introduce *approximation methods for our quality metrics* that can be computed in practice even when clients' data distributions are unknown (Section 4.5.1); 4) We exploit the structure of this NP-hard optimization problem to provide a *provably optimal, tractable* solution (Section 4.6); and finally, 5) We demonstrate our approach's *practical feasibility* by learning models with higher accuracy and fewer clients compared to heuristic recruitment methods, on synthetic and real datasets (Section 4.7).

4.2 Related Work

To the best of our knowledge, we are the first to study the client recruitment problem for federated learning. Yet there exists a fairly similar concept named *client selection*, which studies the scheduling of client participation in each global round of federated learning. E.g., [67] proposes an adaptive selection algorithm to maximize the number of participating clients in each round while subject to resource restrictions. Similar topics are discussed in [103], which assumes clients follow a specific scheduling policy for global aggregations. Client recruitment complements these selection policies by ensuring that a suitable group of clients is available to be selected in the first place: without a good recruitment, generic client selection algorithms cannot guarantee the convergence of federated learning to a globally optimal solution [20]. Client selection also requires all clients to stay active and ready to be summoned anytime, even if they are not always selected, which without a good recruitment process is unrealistic due to the system challenges to be discussed in Section 4.3.

In practical settings, client recruitment can thus limit the cost of federated learning since it pre-excludes disqualified clients before any training steps are taken or incentives offered, while client selection still requires the operator to pay recruited clients, who have committed to being available for training even if they are never ultimately selected. The client recruitment method discussed in this chapter is independent of the remaining training details, and can thus be coupled with any learning algorithm and selection strategies.

4.3 Federated Learning Background And Notations

Federated learning trains a single model by attempting to minimize the model's empirical risk, i.e., the training loss, over data from multiple clients. Let $\mathcal{U}$ denote the set of K candidate clients, each with a dataset $\mathcal{D}_k = \{(u_i, v_i)\}_i$, where u_i, v_i are a feature vector and the corresponding label. Let $l(w; u, v)$ be a loss function with weight vector w and a data record (u, v). The local empirical risk of client k is:

$$\tilde{R}_k(w; \mathcal{D}_k) = \frac{1}{|\mathcal{D}_k|} \sum\nolimits_i l(w; u_i, v_i) \tag{4.1}$$

The ultimate goal of training is thus to find w that minimizes the empirical risk over the global dataset $\mathcal{D}_x = \cup_k \mathcal{D}_k$:

$$\min_w \tilde{R}_x(w; \mathcal{D}_x) = \sum\nolimits_{k=1}^{K} \frac{n_k}{n_x} \tilde{R}_k(w; \mathcal{D}_k) \tag{4.2}$$

Here $n_k = |\mathcal{D}_k|$, and $n_x = \sum_k n_k$, representing the total samples of all recruited clients determined by the recruitment decision x, which we formally define in Section 4.4. To minimize $\tilde{R}_x$, the distributed stochastic gradient descent (SGD) paradigm is utilized. A central coordinator maintains a global weight w, and each client maintains a local weight w_k. The training repeats the following 3 steps for $t = 1, ..., T$.

1) *Synchronization*: The coordinator broadcasts the latest global weight $w^{\tau t}$ to the clients through the network. Clients then update their local weights $w_k^{\tau t}$ with $w^{\tau t}$.

2) *Local optimization*: Each client k runs SGD in parallel for τ steps to minimize its local risk $\tilde{R}_k$, getting $w_k^{(t+1)\tau}$.

3) *Aggregation*: The coordinator aggregates clients' local weights $\{w_k^{(t+1)\tau}\}_k$ by setting the next global weight as their weighted average: $w^{(t+1)\tau} = \sum_k \frac{n_k}{n_x} w_k^{(t+1)\tau}$.

Client selection chooses which clients perform these steps in each iteration, while client recruitment chooses the set of eligible clients $\mathcal{U}_x$. This chapter assumes all algorithm parameters are given, including τ and T. Our results are thus robust to different algorithm settings and complement optimizations of these parameters. The challenges of federated learning are:

Statistical Challenge: Non-IID datasets. Unlike in the data center where data is assumed to be identically and independently distributed among workers, clients' data distributions in federated learning may well be non-IID. Thus, we suppose that the samples in every local dataset $\mathcal{D}_k$ are independently drawn from a distinct distribution $\mathcal{P}_k$. Non-IID data can greatly decelerate the training. The training loss can be bounded by a monotonically increasing function of the average difference in local and global distributions $|\tilde{P}_k - \tilde{P}_x|$ [99]:

$$\text{training loss} \propto \sum\nolimits_k \frac{n_k}{n_x} |\tilde{\mathcal{P}}_k - \tilde{\mathcal{P}}_x| \tag{4.3}$$

System Challenge: Stragglers and failures. User devices have relatively constrained computation resources (e.g. CPU, memory), which furthermore must be shared among many apps. These make stragglers (devices that take a long time to run local iterations) and even occasional device failures, e.g., due to lost power or network connectivity, more likely to appear in federated learning. These must be treated carefully to ensure the success of the learning algorithm.

In this chapter our analysis is largely based on the vanilla **FedAvg** algorithm. Incorporating client recruitment with other variant algorithms such as asynchronous federated learning, model personalization, and dynamic client selection will be an interesting future research direction.

4.4 System Modeling

Formally speaking, given a list of candidate clients $\mathcal{U} = \{u_k\}_k$, the goal of client recruitment is to choose the optimal subset of clients $\mathcal{U}_x \subseteq \mathcal{U}$ that will run the learning algorithm to optimize the overall performance of federated learning. In this section, we first model the local/global/population data distributions, which we then use to propose formal performance metrics for both the learning accuracy and the efficiency.

4.4.1 Data Distributions

Since federated learning trains one model for all clients, we assume all data is generated in an IID manner from a *population distribution* $\mathcal{P}$. On the other hand, each client's data individually forms a *local distribution* $\mathcal{P}_k$, which can differ from other local distributions and from $\mathcal{P}$. When we compare two local datasets, we assume the data is not identically distributed between them. In contrast, when we discuss the union of all local datasets, we treat each sample as IID distributed in $\mathcal{P}$. E.g., suppose we are training a model to predict temperature from features such as the amount of sunlight and rainfall. Then $\mathcal{P}$ represents the joint distribution of world-wide temperature with these features. Since a client can only collect temperature data in a small region, its local distribution $\mathcal{P}_k$ only reflects the regional climate characteristics. As a result, clients in different regions possess divergent local distributions. In the meanwhile, all these data points are essentially generated within the same Earth climate system. Thus when forged together, they do follow the world-wide distribution $\mathcal{P}$ in an IID manner.

A local dataset $\mathcal{D}_k$ may not describe its local distribution $\mathcal{P}_k$ well if insufficient data points were collected. In practice, the operator estimates $\mathcal{P}_k$ by indirectly evaluating its *empirical distribution* $\tilde{\mathcal{P}}_k$, which converges to the real $\mathcal{P}_k$ when $\mathcal{D}_k$ grows larger. Similarly, we define the global dataset $\mathcal{D}_x = \cup_k \mathcal{D}_k$ as the union of all recruited datasets. Data in $\mathcal{D}_x$ forms the *global empirical distribution* $\tilde{\mathcal{P}}_x$. Likewise, $\tilde{\mathcal{P}}_x$ is a weighted average of local empirical distributions: $\tilde{\mathcal{P}}_x = \sum_k \frac{n_k}{n_x} \tilde{\mathcal{P}}_k$. Since all data in $\mathcal{D}_x$ is independently drawn from $\mathcal{P}$, $\tilde{\mathcal{P}}_x$ can be regarded as an empirical estimation of $\mathcal{P}$ when a reasonably large number of clients are recruited. In the climate data example, if we have recruited clients from all climatic zones in the world, the union of this data $\tilde{\mathcal{P}}_x$ becomes a good representative of the whole Earth climate system $\mathcal{P}$.

We suppose the operator can estimate $\mathcal{P}$ with a small benchmark dataset (e.g., as in [94]) $\tilde{D} \sim \mathcal{P}$, and we denote its empirical distribution by $\tilde{P}$. Since $\tilde{D}$ is small in size, it cannot be directly used for training. Instead, this educated guess of the population allows the operator to gauge the clients' quality and representativeness. E.g., the operator may estimate the distribution of rainfall from historical climate data. We show how to estimate data quality with $\tilde{P}$ in Sections 4.5 and 4.7.

4.4.2 Performance Metrics

We will consider two categories of performance measures. The first category is the accuracy of the output model for the distribution $\mathcal{P}$, which includes not only the training loss, but also the model's generalizability and its representativeness. The second measures the training efficiency, which includes the time to complete the training, and the cost incurred.

Reduce training loss with high-quality data: A dataset $\mathcal{D}_k$ is considered of good quality if its distribution $\tilde{\mathcal{P}}_k$ resembles the population $\mathcal{P}$. From (4.3), if $\tilde{\mathcal{P}}_x$ resembles $\mathcal{P}$ (see "representativeness" below), the quality of the local datasets directly determines the training loss. A dataset $\mathcal{D}_k$ with a small distribution divergence $|\tilde{\mathcal{P}}_k - \mathcal{P}|$ yields small training loss.

Reduce generalization error with more data: Given a loss function l and dataset $\mathcal{D}$, the generalization error $|\tilde{R} - R|$ is the divergence between the empirical risk $\tilde{R}(w; \mathcal{D}) = (\sum l(w; u, v))/|\mathcal{D}|$ and the real risk $R(w) = \mathbb{E}_{\mathcal{P}}[l] = \int l(w; u, v) d\mathcal{P}$. While the training loss gauges the model's performance on the training data, the generalization error reflects its accuracy when applied to *new samples drawn from the recruited distributions*. If a client has insufficient data, its local empirical distribution $\tilde{\mathcal{P}}_k$ may poorly approximate $\mathcal{P}_k$, which implies a large generalization error. Existing works generally omit the generalization error as they take the training data as given. For us, however, client recruitment determines the size of the dataset, affecting the generalization error.

Choose for population representativeness: For the trained model to be applicable to *unrecruited datasets*, the recruited clients, when forged together, must be representative of the population $\mathcal{P}$. Indeed, if the clients do not cover portions of the population space, we will perform poorly in those areas. E.g., including polar region data complicates the training of models that predict worldwide temperatures, but failing to do so can degrade the model's performance in this region.

Control the completion time: Federated learning is useless if the training process does not complete in reasonable time. We define the completion time as the expected time for the coordinator to finish all T rounds of aggregations.

Control the cost: Since the size of an individual local dataset is usually small, a typical execution of federated learning may need thousands of recruited clients. The operator should thus make sure the resulting expense is affordable.

4.5 Problem Formulation

We formulate client recruitment as the following optimization problem: Given a set of candidate clients $\mathcal{U} = \{U_k\}_{k=1}^{K}$, let $x \in \{0, 1\}^K$ be a binary vector denoting the recruitment decision for each client. The operator picks an optimal subset $\mathcal{U}_x = \{U_j | x_j = 1\}$ to minimize an objective function f, subject to a given maximum completion time I_t and cost I_c.

Problem 4.1. *Client Recruitment*

$$\begin{aligned} \min_{x \in \{0,1\}^K} \quad & f(x) = \gamma_{tl} f_{tl}(x) + \gamma_{ge} f_{ge}(x) + \gamma_{rp} f_{rp}(x) \\ s.t. \quad & g_t(x) \leq I_t, g_c(x) \leq I_c \end{aligned}$$

Here f consists of 3 terms that determine the accuracy of the trained model: f_{tl}, f_{ge}, f_{rp}, which respectively upper bound the training loss, the average generalizability, and the representativeness. $f(x)$ determines the goodness of the trained model when applied to existing or future data points generated by both recruited and unrecruited clients. The coefficients $\gamma_{tl}, \gamma_{ge}, \gamma_{rp}$ determine the relative importance of these terms, and g_t, g_c are respectively the completion time and cost.

4.5.1 Quantifying Accuracy Metrics

We first consider the training loss. From (4.3), the training loss is determined by the divergence between local and global empirical distributions $\sum_k \frac{n_k}{n} |\tilde{\mathcal{P}}_k - \tilde{\mathcal{P}}_x|$. However, since the global distribution can only be determined after the recruitment process, it is hard to optimize the divergence directly. We thus use the fact that $\tilde{\mathcal{P}}_x$ resembles $\mathcal{P}$ (Lemma 4.2) to define:

$$f_{tl}(x) = \sum\nolimits_k \frac{x_k n_k}{n_x} |\tilde{\mathcal{P}}_k - \tilde{\mathcal{P}}| \tag{4.4}$$

Sharing this metric preserves user privacy since it does not actually require the individual empirical distributions $\tilde{\mathcal{P}}_k$'s. Instead, the required information from the clients $|\tilde{\mathcal{P}}_k - \tilde{\mathcal{P}}|$ only encodes the distance of local distributions to the population. Below we provide tractable methods and formula to approximate f_{tl}, **without the need to know $\tilde{\mathcal{P}}_k$ or $\mathcal{P}_k$**. We verify the effectiveness of these methods in Section 4.7.

- *Counting classes:* Consider a classification problem with L classes, and suppose $\tilde{\mathcal{P}}_k$ and $\tilde{\mathcal{P}}$ have densities $\tilde{p}_k$ and $\tilde{p}$. We can write $|\tilde{\mathcal{P}}_k - \tilde{\mathcal{P}}| = \int |\tilde{p}_k(u,v) - \tilde{p}(u,v)| du dv = \sum_{i \in [L]} \int |\tilde{p}_k(u|v)\tilde{p}_k(v=i) - \tilde{p}(u|v)\tilde{p}(v=i)| du$.

Assume $\tilde{p}_k(u|v) = \tilde{p}(u|v)$, i.e., local features have the same distribution as the population given the label. Thus, $\int |\tilde{p}_k - \tilde{p}| \propto \sum_{i\in[L]} |\tilde{p}_k(v = i) - \tilde{p}(v = i)|$, where $\tilde{p}$ is known a prior. Denoting by C_i^k the number of data points with label i in $\mathcal{D}_k$, $\tilde{p}_k(v = i) = C_i^k / \sum_{i=1}^{L} C_i^k$. Thus, the whole $\sum |\tilde{p}_k(v = i) - \tilde{p}(v = i)|$ can be easily computed by simply counting the number of labels each client sees. Estimating $\tilde{P}$ from $\tilde{D}$ entails the same simple counting process.

- *Gaussian graphic model approximation:* For general supervised learning with continuous labels, we can formulate the features and the label as a Gaussian graphic model. A local empirical distribution is then fully specified by the mean and covariance $(\tilde{\mu}_k, \tilde{\Sigma}_k)$. The quality measure then becomes the divergence between two Gaussian distributions, which can be quantified by the Kullback–Leibler divergence: $|\tilde{\mathcal{P}}_k - \tilde{\mathcal{P}}| \propto D_{KL}\left(\mathcal{N}(\tilde{\mu}_k, \tilde{\Sigma}_k), \mathcal{N}(\tilde{\mu}, \tilde{\Sigma})\right)$. Estimating $\tilde{P}$ from $\tilde{D}$ entails computing $\tilde{\mu}, \tilde{\Sigma}$ and inferring the graph connectivity (e.g. from the covariance), which is tractable.

Next, we model the average generalizability. Since the training objective of federated learning $\tilde{R}_x$ is an average of local empirical risks as in (4.2), we similarly quantify the average generalization error of local datasets by $\sum_k \frac{x_k n_k}{n_x} |\tilde{R}_k - R_k|$. To formulate it, we rely on Lemma 4.1 as follows:

Lemma 4.1. *There exists a class of convex learning problems (e.g. linear regression), for which we can obtain the following generalization error bound for all clients k:*

$$|\tilde{R}_k - R_k| = O\left(n_k^{-0.5}\right) \tag{4.5}$$

For example, [72] proves this bound for the linear regression model. A tighter convergence bound taking the form $O(n_k^{-\beta})$ with $\beta > 0.5$ is also possible using more sophisticated statistical tools. For simplification and to accommodate non-convex models that may have looser risk generalization bounds, we assume a relatively big $\beta = 0.5$. However, our analysis can be easily extended to any $\beta < 1$. We thus define:

$$f_{ge}(x) = \sum\nolimits_k \frac{x_k n_k}{n_x} n_k^{-0.5} \tag{4.6}$$

We then model the representativeness. To make sure the chosen distributions can represent basic characteristics of the population distribution, we seek to minimize the divergence between $\tilde{\mathcal{P}}_x$ and $\mathcal{P}$. As is discussed in Section 4.4.1 where we assume

$\tilde{\mathcal{P}}_x$ is an empirical distribution of $\mathcal{P}$, and using the central limit theorem, we have the following uniform bound:

Lemma 4.2. *$\tilde{\mathcal{P}}_x - \mathcal{P}$ converges in distribution to the Gaussian distribution with 0 mean at the rate of $O(n_x^{-0.5})$.*

Therefore, statistically, when n_x grows larger, $\tilde{\mathcal{P}}_x$ becomes a more accurate representative of $\mathcal{P}$. We thus define:

$$f_{rp}(x) = n_x^{-0.5} \tag{4.7}$$

Combining (4.4), (4.6), and (4.7), the objective f can be expressed as in (4.8). Since the coefficient γ_{rp} is a positive constant independent of x, we normalize it to 1. The s_k value here is a weighted sum of the client's quality representation $|\tilde{\mathcal{P}}_k - \tilde{\mathcal{P}}|$ and quantity representation $n_k^{-0.5}$. It thus enables us to quantify the quality-quantity tradeoff when choosing user datasets.

$$\begin{aligned} f(x) &= \gamma_{rp}\left(\frac{\sum x_k n_k s_k}{n_x} + n_x^{-0.5}\right) \\ s_k &= \frac{\gamma_{tl}}{\gamma_{rp}}|\tilde{\mathcal{P}}_k - \tilde{\mathcal{P}}| + \frac{\gamma_{ge}}{\gamma_{rp}} n_k^{-0.5} \end{aligned} \tag{4.8}$$

4.5.2 Quantifying System Metrics

Now we analyze the completion time. We assume for each round of the training, the coordinator will wait up to a predetermined duration E_0. The global weight will then be calculated based on the weights received before the deadline.

We model the client failure as a Markov chain. An active client crashes with probability q_f, and a failed client recovers with probability q_r. Here $q_r, q_f > 0$. Suppose there are m recruited clients. Letting A^t be the number of active clients at iteration t, which has the following properties.

Proposition 4.1. *A^t is an ergodic Markov chain. In the steady state, the probability that there are i active clients equals*

$$\pi_i \triangleq \mathbb{P}(A^\infty = i) = \frac{\binom{m}{i}(q_r/q_f)^i}{(1 + q_r/q_f)^m} \tag{4.9}$$

Proof. The transition probability is given by

$$\begin{aligned} P_{ij} &= \mathbb{P}(A^{t+1} = j | A^t = i) \\ &= \sum_k \binom{i}{k} (1-q_f)^k q_f^{i-k} \binom{m-i}{j-k} q_r^{j-k} (1-q_r)^{m-i-j+k} \end{aligned} \tag{4.10}$$

Here the summation is taken from $k = \max\{0, i+j-m\}$ to $k = \min\{i, j\}$. This conditional probability P_{ij} does not depend on t, so A^t is a homogeneous Markov chain. Furthermore, it is easy to check that $\max\{0, i+j-m\} \leq \min\{i, j\}$ given that $0 \leq i, j \leq m$. Therefore, A^t is positive recurrent since $P_{ij} > 0$ for all pairs of states (i, j). In addition, A^t is aperiodic since there is a self-loop for every state i: $P_{ii} > 0$. As a result, A^t is ergodic.

Besides, it is easy to verify that for all (i, j), we have

$$\frac{P_{ij}}{P_{ji}} = \frac{P_{i0} P_{0j}}{P_{j0} P_{0i}} \tag{4.11}$$

Let $\pi = \{\mathbb{P}(A^\infty = i)\}_i$ be the steady state vector, and $\boldsymbol{P} = [P_{ij}]_{ij}$ be the transition matrix. It can be shown that with (4.11), the solution of the steady state equations $\pi \boldsymbol{P} = \pi$ satisfies:

$$\pi_i = \frac{P_{0i}}{P_{i0}} \pi_0 \tag{4.12}$$

Combing (4.10,4.12), and using the condition that $\sum_{i=0}^m \pi_i = 1$, we can get (4.9). $\square$

Since ergodic Markov chains converge exponentially fast, we only consider the steady state. The probability that no clients fail is thus $(\frac{q_r}{q_f + q_r})^m$. To model the system heterogeneity of clients, we partition clients into N groups according to their devices, network qualities, battery levels etc. Suppose each group has m_z clients for $z = 1, ..., N$, and all the clients inside a group z have the same failure rate q_f^z and recovery rate q_r^z. Since clients are running independently, the probability that all clients in all groups are active is then $\prod_{z=1}^N (\frac{q_r^z}{q_f^z + q_r^z})^{m_z}$.

If a client k in group z is active, we model its per-iteration runtime as a random variable $Y_k^z \sim \exp(\lambda^z)$. The expected full iteration runtime when all clients are active is: $\Gamma(m_1, ...m_N) = \mathbb{E}[\min\{\max_z \max_k Y_k^z, E_0\}]$. The completion time is as follows, where m_z depends on the recruitment.

$$\begin{aligned} g_t(x) &= g_t(m_1, ..., m_N) \\ &= T\left(\Gamma \prod_{z=1}^{N} (\frac{q_r^z}{q_f^z + q_r^z})^{m_z} + E_0(1 - \prod_{z=1}^{N} (\frac{q_r^z}{q_f^z + q_r^z})^{m_z})\right) \end{aligned} \tag{4.13}$$

Intuitively, the completion time increases when we recruit more clients. This is summarized in Proposition 4.2.

Proposition 4.2. *The completion time* $g_t(m_1, ...m_N)$ *increases when any* $m_z, z = 1, ..., N$ *increases.*

Proof. Let $Y = \min\{\max_z \max_k Y_k^z, E_0\}$. The cumulative distribution function of Y is

$$F_Y(Y \leq y) = \mathbf{1}_{y \leq E_0} \prod_{z=1}^{N} (1 - e^{-\lambda_z y})^{m_z} + \mathbf{1}_{y > E_0} \tag{4.14}$$

Since $Y \geq 0$, we have

$$\begin{aligned} \Gamma(m_1, ..., m_N) = \mathbb{E}[Y] &= \int_0^{\infty} 1 - F_y(y) dy \\ &= \int_0^{E_0} 1 - \prod_{z=1}^{N} (1 - e^{-\lambda_z y})^{m_z} dy \end{aligned} \tag{4.15}$$

Because F_y is continuous w.r.t. $(y, m_1, ..., m_N)$, we get

$$\frac{\partial \Gamma}{\partial m_z} = -\int_0^{E_0} \log(1 - e^{-\lambda_z y}) \prod_{z=1}^{N} (1 - e^{-\lambda_z y})^{m_z} dy > 0 \tag{4.16}$$

Thus,

$$\begin{aligned} \frac{\partial g_t}{T \partial m_z} =& \frac{\partial \Gamma}{\partial m_z} \prod_{z=1}^{N} (\frac{q_r^z}{q_f^z + q_r^z})^{m_z} + \\ & (\Gamma - E_0) \prod_{z=1}^{N} (\frac{q_r^z}{q_f^z + q_r^z})^{m_z} \log(\frac{q_r^z}{q_f^z + q_r^z}) > 0 \end{aligned} \tag{4.17}$$

Therefore, g_t increases when any m_z increases. □

Finally, we consider the cost. The cost depends on specific payment mechanisms (e.g. [50]) adopted. Here we assume a generic case where each client k has an exogenous price c_k:

$$g_c(x) = \sum_k^K x_k c_k \leq I_c \tag{4.18}$$

4.6 The Optimal Client Recruitment

From Section 4.5, each client $U_k \in \mathcal{U}, k = 1, .., K$ can be characterized by a tuple $(|\tilde{\mathcal{P}}_k - \tilde{\mathcal{P}}|, n_k, \mathcal{Z}(k), c_k)$, representing respectively the distribution divergence, the local dataset size, the group number, and the ask price. Clients in a group z have failure rate q_f^z, recovery rate q_r^z and processing rate λ^z. As discussed above, the client can readily compute this information and send it to the operator without significant privacy loss at the start of the recruitment.

Combining (4.8), (4.13), and (4.18), Problem 4.1 becomes:

Problem 4.2. *Client Recruitment*

$$\begin{aligned}
\min_{x \in \{0,1\}^K} \quad & f(x) = \frac{1}{n_x} \sum_k x_k n_k s_k + n_x^{-0.5} \\
s.t. \quad & g_t(x) = (\Gamma - E_0) \prod_{z=1}^{N} \left(\frac{q_r^z}{q_f^z + q_r^z} \right)^{m_z} + E_0 \leq \frac{I_t}{T} \\
& g_c(x) = \sum_k^K x_k c_k \leq I_c
\end{aligned}$$

Proposition 4.3. *Problem 4.2 is NP-Hard.*

Proof. Let $I_t = \infty, s_k = 0$, then $\min f \Leftrightarrow \max n_x$. We thus reduce Problem 4.2 to the NP-Hard Knapsack problem. □

4.6.1 Unconstrained Optimization

We first consider the unconstrained version of Problem 4.2, i.e., when all the limits I_t, I_c approach infinity. This is useful when the operator has gained complete right of usage of the clients (so that they can be used for free without time limit). This unconstrained optimization can be solved in polynomial time using the following proposition:

Proposition 4.4. *(Unconstrained Client Recruitment) Suppose clients are sorted by their s values, i.e. $s_1 \leq ... \leq s_K$. The solution to problem Problem 4.2 without* *constraints must be of the form:* $x^* = (1, 1, ..., 1, 0, 0, ...0)$, *i.e., if a client* j *is recruited, all the clients* $k < j$ *must also be recruited.*

Proposition 4.4 indicates that recruiting more clients does not always help improve the accuracy. Intuitively, when more client participate, the overall dataset grows larger and the representativeness should thus improve. However, a chosen dataset $\mathcal{D}_k$ itself may be small in size, making its data biased from $\mathcal{P}_k$. This will enlarge its generalization error. Worse still, if $\mathcal{P}_k$ is also biased from the population $\mathcal{P}$, the training loss will increase as well due to the increased divergences in local distributions. Based on the proposition, we can solve the unconstrained client recruitment problem by simply sorting the devices by their s values, then comparing the objective values for all the K possible choices of x^*. The **time complexity** is dominated by the sorting step, which is $O(K \log K)$.

To prove the proposition, we use the following lemma.

Lemma 4.3. *Consider two recruitments* x^0 *and* x^j *that contain the same set of clients, except that the latter includes client* j *while the former does not. If* $f(x^j) \leq f(x^0)$, *then* $f(x^j)$ *decreases as we increase* n_j, *the number of data points in* j.

Proof. For convenience we rewrite $f(x^j) = f^j(n_j)$. Note that

$$f^j(n_j) = \frac{\sum_k x_k^0 n_k s_k + n_j s_j}{\sum_k x_k^0 n_k + n_j} + \big(\sum_k x_k^0 n_k + n_j\big)^{-0.5} \tag{4.19}$$

$$\frac{df^j}{dn_j} = \frac{\sum_k x_k^0 n_k (s_j - s_k)}{(\sum_k x_k^0 n_k + n_j)^2} - \frac{0.5}{(\sum_k x_k^0 n_k + n_j)^{1.5}} \tag{4.20}$$

As n_j increases, (4.20) is either i) strictly negative, or ii) first positive then negative. Thus, (4.19) will either i) strictly decrease, or ii) first increase then decrease. Using the condition $f^j(n_j) = f(x^j) \leq f(x^0) = f^j(0)$, the value of n_j must fall into the decreasing interval. Therefore, further increasing n_j will only cause the objective f to decrease. $\square$

We then prove Proposition 4.4:

Proof. (Proposition 4.4) We prove by contradiction. Assume the clients are already sorted by the s value. Suppose the optimal recruitment $x^* = x^j$, where client j is the last recruited client, and there exists at least one unrecruited client U_i, such that $i < j, x_i^j = 0$. Denote by $f(x|U(s, n))$ the objective value for recruiting clients in x, plus an additional client U who has parameters s and n. Let x^0 be a copy of x^j, except that client j is not recruited. We thus have $f(x^*) = f(x^j) \leq f(x^0)$, and:

$$\begin{aligned} &f(x^j|U(s_i, n_i)) \leq f(x^j|U(s_j, n_i)) \\ =&f(x^0|U(s_j, n_i + n_j)) < f(x^0|U(s_j, n_j)) = f(x^j) \end{aligned} \tag{4.21}$$

The first inequality is due to the condition $s_i \leq s_j$. The second follows from Lemma 4.3 with the fact that $f(x^j) \leq f(x^0)$. Therefore, adding the unchosen client i to the recruitment x^j results in a smaller objective value $f(x^j|U(s_i, n_i))$, contradicting that $x^* = x^j$. □

4.6.2 Constrained Optimization

As we would expect from our NP-hardness result (Proposition 4.3), Proposition 4.4 does not hold when incorporating the constraints. To solve the constrained optimization Problem 4.2, we first relax the completion time constraint $g_t \leq I_t$ by N linear constraints $\mathcal{G}_t(m_1, ..., m_N) = \{m_z \leq M_t^z\}_{z=1}^N$ on m_z.

$$\begin{aligned} M_t^z = \min\Big\{ &\sum\nolimits_{k=1}^{K} \mathbf{1}(\mathcal{Z}(k) = z), \\ &\operatorname{argmax}_{m_z}\{g_t(0, ..., 0, m_z, 0, ..., 0) \leq I_t\}\Big\} \end{aligned} \tag{4.22}$$

According to Proposition 4.2, if $(m_1, ..., m_N)$ satisfies the original completion time constraint $g_t(m_1, ...m_N) \leq I_t$, it also satisfies the relaxed constraint $\mathcal{G}_t(m_1, ..., m_N)$. We then construct a new optimization Problem 4.3. Here we define $s'_k = n_k s_k, I_s = \sum_k^K s'_k$. Problem 4.3 maximizes a linear objective, subject to $N+2$ linear constraints. This is a multi-dimensional Knapsack problem, and can be solved by the dynamic programming (DP) algorithm [60].

Problem 4.3. *Data Quantity Maximization*

$$\begin{aligned} \max_{x \in \{0,1\}^K} \quad & n_x = \sum\nolimits_k x_k n_k \\ s.t. \quad & m_z = \sum\nolimits_k \mathbf{1}(\mathcal{Z}(k) = z) x_k \leq M_t^z, z = 1, ..., N \\ & g_c(x) = \sum\nolimits_k x_k c_k \leq I_c, \;\; g_s(x) = \sum\nolimits_k x_k s'_k \leq I_s \end{aligned}$$

As in conventional DP procedures, we construct a $N+3$ dimensional table $\phi(k, m_1, ..., m_N, c, s)$ to keep track of the algorithm states. $\phi(k, m_1, ..., m_N, c, s)$ represents the maximum value of n_x we can get, under the conditions that: 1) we only pick from the first k clients (the order of clients does not matter); 2) we recruit at most m_z clients for each group z; 3) the cost is less than or equal to c; and 4) the sum $\sum s'_k \leq s$. Conditions 2) to 4) correspond to the three constraints in Problem 4.3. The DP algorithm gradually increments the recruitment boundary k. For each k, the following recursive relation guarantees the consistency of ϕ:

$$\begin{aligned} \phi(k, m_1, ..., m_N, c, s) = \max\{&\phi(k-1, ..., m_{\mathcal{Z}(k)} - 1, \\ &..., c - c_k, s - s'_k) + n_k, \phi(k-1, m_1, ..., m_N, c, s)\} \end{aligned} \tag{4.23}$$

In practice, c and s may be float numbers, but we can easily normalize them to integers. The correctness of the algorithm is obvious by induction. The **time complexity** is bounded by the size of the DP table, which is $O(KI_cI_s \prod_{z=1}^{N} M_t^z)$.

Algorithm 4.1 *DP and Revisit.* Solving Problem 4.2.

```
procedure OPTIMIZE
    $\phi \leftarrow$ (solve Problem 4.3 with DP), $f^* \leftarrow \infty$
    if $\phi(K, M_1^t, ..., M_N^t, I_c, I_s) \leq 0$ then
        // Infeasible
        return $\infty$
    for $s = 0$ to $I_s$ do
        for $m_1 = 0$ to $M_t^1$ do
            ......
            for $m_N = 0$ to $M_t^N$ do
                if $g_t(m_1, ..., m_N) \leq I_t$ then
                    $f^* \leftarrow \min(f^*$, Equation (4.24))
    return $f^*$
```

Now we go back to the original Problem 4.2. We can observe that when the value of s is fixed, minimizing the objective f is equivalent to maximizing the number of samples n_x. Since the ϕ table records a one to one mapping of s to the maximum n_x, we can utilize ϕ to reconstruct the original objective f.

Formally speaking, given $(m_1, ..., m_N, s)$, we define

$$f' = \frac{s}{\phi(K, m_1, ..., m_N, I_c, s)} \\ + (\phi(K, m_1, ..., m_N, I_c, s))^{-0.5} \tag{4.24}$$

Intuitively, for a solver x^* of Problem 4.2, if its corresponding s^* and m_z^* are recorded during the DP iteration, then f' should be "related" to the optimal objective value $f(x^*)$. We thus propose Algorithm 4.1 to solve the constrained client recruitment. Its correctness is shown below. The **time complexity** is dominated by the DP step as $O(KI_cI_s\prod_{z=1}^{N} M_t^z)$.

Proposition 4.5. *Algorithm 4.1 solves Problem 4.2.*

Proof. Let x^* be a solver of Problem 4.2, with $n_x^* = \sum x_k n_k, s^* = \sum x_k^* n_k s_k, m_z^* = \sum \mathbf{1}(\mathcal{Z}(k) = z)x_k^*$, then

$$\phi(K, m_1^*, ..., m_N^*, I_c, s^*) = n_x^* \tag{4.25}$$

Otherwise, if the left hand side is smaller, the DP algorithm yields a smaller objective n_x^0 for some recruitment x^0. Both x^0 and x^* satisfy the four conditions in the definition of ϕ at $(K, m_1^*, ..., m_N^*, I_c, s^*)$. But replacing x^0 with x^* yields a greater objective $n_x^* > n_x^0$. This contradicts the correctness of DP. In addition, if the left hand side is greater, the DP algorithm finds a recruitment x^0 that has $s^0 = \sum x_k^0 n_k s_k \leq s^*$, $n_x^0 = \sum x_k^0 n_k > n^*$, and satisfies all the constraints in Problem 4.2. Thus, by recruiting x^0, we have $f(x^0) = \frac{s^0}{n_x^0} + (n_x^0)^{-0.5} < \frac{s^*}{n_x^*} + (n_x^*)^{-0.5} = f(x^*)$. This shows x^0 is a better recruitment than x^*, which contradicts the assumption that x^* is an optimal. Thus, since Algorithm 4.1 iterates through all the feasible elements, we must at some point visit $(K, m_1^*, ..., m_N^*, I_c, s^*)$. □

4.7 Performance Evaluation

We finally evaluate the performance of our client recruitment strategy with a classification problem and a regression problem. We set the aggregation deadline $E_0 = 30$. Unless otherwise noted, we assume clients have the default specification: Group I $= (q_f^1 = 0.001, q_r^1 = 0.6, \lambda^1 = 0.1)$. If a client fails upon the aggregation, we replace its $w_k^{t\tau}$ with the previous global weight $w^{t\tau}$. Throughout this section, we uniformly at random set the cost of each client in the range of 1 to 9. We consider three baseline recruitment strategies:

- **All participation:** recruiting all clients. Comparisons with this baseline show the value of intelligent client recruitment.
- **Greedy recruiting by quantity:** greedily choosing clients with the most data samples until any constraint is active.
- **Greedy recruiting by quality:** greedily choosing clients with best quality until any constraint is active.

In the case of unconstrained optimization, we force the greedy baselines to choose the same number of clients as the optimal recruitment. By comparing to the latter two baselines, we present the value of considering both quantity and quality of the data. Since we take the training parameters as fixed as discussed in Section 4.3, we will not fine-tune them in the simulation. We pick these values such that all model training in all experiments are fully converged. We then only need to determine the relative weights of the training accuracy (γ_{tl}) and generalizability (γ_{ge}). In practice, they can be tuned by optimizing the unconstrained recruitment through grid search.

4.7.1 Image Classification

We first consider the MNIST digit recognition problem. We use the same 2NN model as [61]. All clients are equipped with the Adam optimizer and use the same set of training parameters. We use a batch size of 10 for local iterations. The initial learning rate is set to 3e-4, and decays by half every 200 steps. The local epochs $\tau = 30$ and global epochs $T = 50$.

Dataset and clients. To construct the non-IID distributions of local datasets, we assign each client a set of class labels (digits). Clients then randomly sample training images corresponding to the assigned labels. We limit each client to sample 10 to 40 images. For j from 1 to 10, we assign j label(s) to 30 new clients, resulting in total 300 candidate clients. The default MNIST test dataset is used.

Approximation of divergence. We use the "counting classes" method described in Section 4.5.1 to approximate the probability divergence. Here we have $L = 10$ classes, thus $\int |\tilde{p}_k - \tilde{p}| = \sum_{i=0}^{9} |\tilde{p}_k(v = i) - \tilde{p}(v = i)|$. For the population distribution, all classes appear with the same probability, so $\tilde{p}(v=i) \equiv 0.1$. If a client k has j labels, $\tilde{p}_k(v=i) = 1/j$ if label i was assigned, or $\tilde{p}_k(v=i) = 0$ otherwise. Thus, $\sum |\tilde{p}_k(v=i) - \tilde{p}(v=i)| = j(\frac{1}{j} - \frac{1}{10}) + (1-j)\frac{1}{10}$, which clients can easily compute knowing only the number of labels that they see. By tuning the unconstrained recruitment, we choose $\gamma_{tl} = 0.015, \gamma_{ge} = 1$ for all experiments.

Unconstrained recruitment. The left plot of Figure 4.1 shows the convergence progress of the four recruitment strategies. The left plot is the classification problem. The X axis is global epochs, and the Y axis is test accuracy. The optimal strategy's model yields higher test accuracy than other baselines after 10 epochs. The right plot is the regression problem. The X axis is global epochs, and the Y axis is the normalized MSE on the test dataset. The untrained model has MSE=1, and the closed-form solution has MSE=0. 64 clients are recruited by the optimal strategy. The optimal recruitment converges the fastest and obtains the highest test accuracy on the fully trained models. Notably, the optimal strategy can increase the test accuracy by 5.0% compared to simply recruiting all clients, which is a big improvement for most classification problems. Figure 4.2 shows the distribution of recruited clients w.r.t. the number of classes assigned to them. The left plot is the counts of recruited clients. The right plot is the sizes of local datasets for recruited clients, where each point represents a client. Compared to the greedy-by-quantity strategy, the optimal recruitment chooses fewer low-quality datasets, but more high-quality ones. Also, most clients recruited by the optimal strategy contain more than 30 samples, but the greedy-by-quality recruitment includes lots of small-sized datasets.

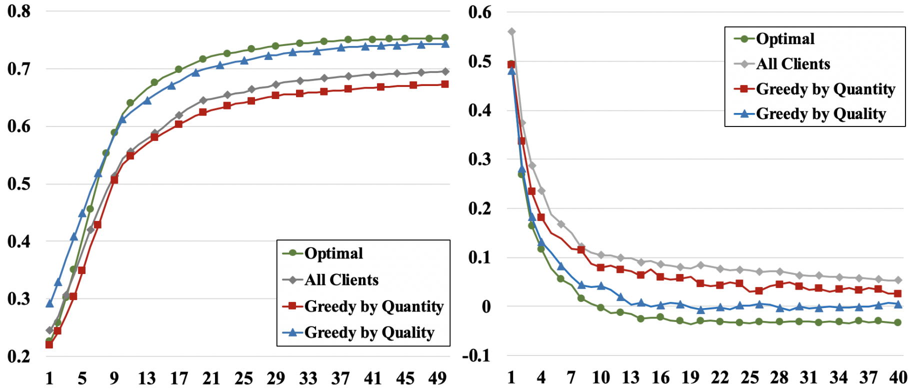


Figure 4.1: Convergence curves for the unconstrained recruitments

Constrained recruitment. The left plot of Figure 4.3 shows the change of test

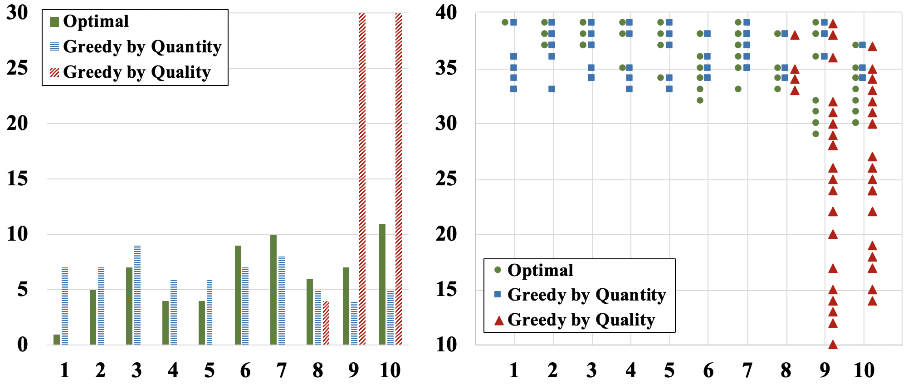


Figure 4.2: The distribution of recruited clients w.r.t. the number of assigned classes

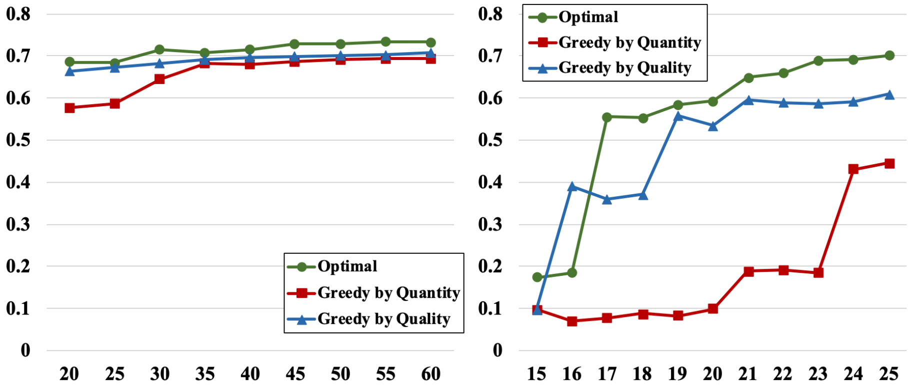


Figure 4.3: Test accuracy when varying the budget I_c and the per round completion time I_t/T

accuracy when we increase the budget I_c from 20 to 60 and take I_t to be infinity. The optimal strategy obtains the highest accuracy for all the budgets I_c. In the right plot of Figure 4.3 we drop the I_c constraints and vary the completion time constraint I_t from $15T$ to $25T$. Apart from Group I, we also create a relatively lower-end specification Group II $= (q_f^2 = 0.01, q_r^2 = 0.5, \lambda^2 = 0.05)$. We randomly pick one third of the clients and assign them to Group II. When I_t/T is down to around half of $E_0 = 30$, only 1 or 2 clients are recruited, so the models do not appear to be trained at all. The optimal strategy exhibits the best performance when I_t is reasonably large, improving the accuracy by 10% to 20%.

4.7.2 Climate Data Regression

We now evaluate client recruitment with a 5-dimensional linear regression model, simulating a climate prediction task. All clients use the Adam optimizer with the initial learning rate set to 1e-3, and decay by 0.8 every 200 steps. We set the batch size as 20, $\tau = 10$, and $T = 40$.

Dataset and clients. We use the U.S. Historical Climatology Network (HCN) dataset [62], which contains climate records for climate stations in the 48 contiguous United States. The local datasets of these stations are by their nature non-IID, allowing us to evaluate how well our recruitment algorithm performs on realistic data distributions. For simplicity, we only use the data on the first day of December from 1960 to 2019, and we randomly pick 1-3 stations from each state, resulting in 117 stations. Each record contains 5 features: station latitude, station longitude, lowest temperature of the day, highest temperature of the day, and precipitation of the day. Our goal is to predict the snowfall of the day. To reflect the uneven sizes of local datasets, we randomly drop some data so that each client has 30 to 69 samples. We test the learned models on a holdout dataset, which is generated by randomly picking 2 unused stations from each state.

Approximation of divergence. We use the second approximation method described in Section 4.5.1 by assuming the 5 features and the snowfall form a fully connected Gaussian graphic model $\mathcal{N}(\mu, \Sigma)$. Thus, each local distribution can be parameterized by the sample mean and the sample covariance $\mathcal{N}(\tilde{\mu}_k, \tilde{\Sigma}_k)$. Similarly, we approximate the population distribution $\mathcal{N}(\tilde{\mu}, \tilde{\Sigma})$ utilizing the unused (neither training nor testing) data. Thus, we only need to compute the divergence between the local Gaussian $\mathcal{N}(\tilde{\mu}_k, \tilde{\Sigma}_k)$ and population Gaussian $\mathcal{N}(\tilde{\mu}, \tilde{\Sigma})$. We normalize the divergences to the range of 0 and 10, and we choose the coefficients $\gamma_{tl} = 0.01$, $\gamma_{ge} = 1$.

Unconstrained recruitment. The right plot in Figure 4.1 shows the mean-squared error (MSE) on the holdout dataset, which includes 1-2 stations from each state, for different strategies. 37 clients are chosen by the optimal recruitment, allowing us to drop most clients as in Section 5.1. Since linear regression is a convex problem, we can easily calculate the closed-form optimal model over the full dataset. For ease of comparison, we normalize the MSE values so that the untrained model has MSE equal 1, and the closed-form solution has MSE equal 0. As in Figure 4.1, the optimal recruitment yields a lower MSE even than the closed-form solution, which illustrates the value of incorporating generalizability and representativeness metrics. Compared to other strategies, the optimal recruitment can decrease the MSE up to 10%.

Similar to Figure 4.2, Figure 4.4 shows the distribution of recruited clients. Here we divide the clients based on their local-population distribution divergences into 10 bins. The X axes are quantile ranges. Left bins correspond to small divergence (i.e. good quality).

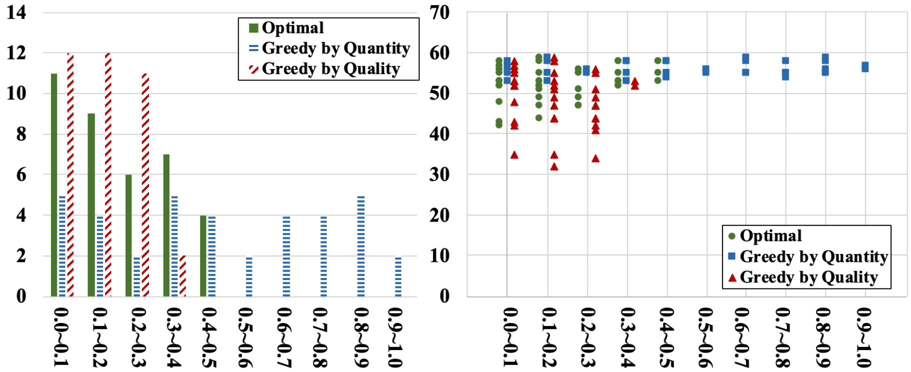


Figure 4.4: The distribution (left: client count; right: dataset size per client) of recruited clients w.r.t. the distribution divergence

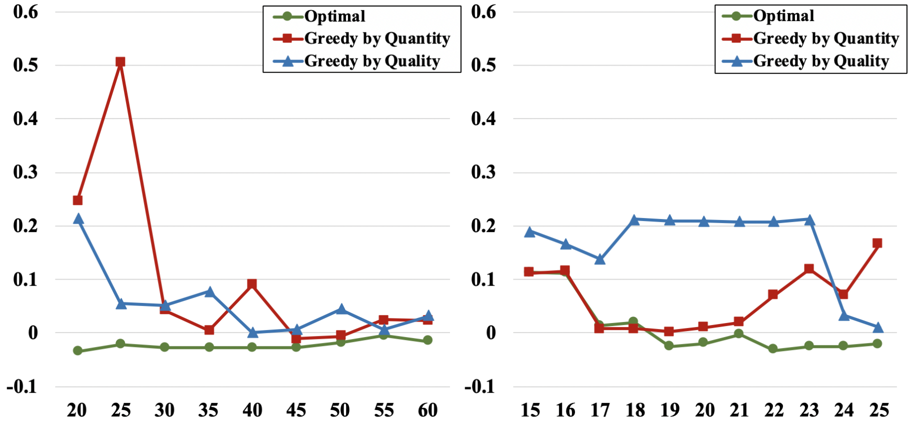


Figure 4.5: Left: Test MSE when varying the budget I_c and I_t/T

Constrained recruitment. Figure 4.5 shows the change of MSE when varying the cost and time limits, on the same setup as in Section 4.7.1. The optimal recruitment obtains the lowest MSE and much smaller variance in most cases.

4.8 Summary

This chapter studies the client recruitment problem in federated learning. We first introduce and quantify five performance metrics that cover both the model's accuracy (training loss, generalization error, representativeness) and the training efficiency (completion time, cost). We then formulate the client recruitment as an NP-Hard optimization problem, and provide an optimal solution algorithm. Finally, we verify our theoretical results with experiments using both synthetic and real-world data. Our results show that recruiting more clients does not always improve the model, and intelligent client recruitment can greatly improve the accuracy of the trained model in constrained execution environments.

Chapter 5

Enabling Flexible Device Participation in Federated Learning

5.1 Introduction

In Chapter 4 we have illustrated how the recruitment strategy can affect the accuracy and efficiency of federated learning. In this chapter, we will show the forms of client partition can also have prominent impact on the learning process. Participating entities of federated learning are mostly mobile devices such as smart phones and tablets. These devices generally have limited computing and communication resources, e.g., due to battery limitations, and have different training data distributions, i.e., data is not independently and identically distributed (non-IID) among devices [53]. To relieve the computation and communication burden, in the last step of the training procedure, the federated learning coordinator may only aggregate a subset of local models. However, only a few device selection policies ensure convergence in the non-IID setting, and the selection must be independent of the hardware status of devices [53]. In other words, for the training to converge successfully, all selected devices must be able to train their local models and upload the results whenever they are selected. This is why the traditional federated learning paradigm requires participating devices to be dedicated to the training during the entire federated learning period, e.g., the popular **FedAvg** algorithm assumes mobile users will participate only when their phones are currently plugged-in, and have unlimited WiFi access [61].

Considering that federated learning typically takes thousands of communication rounds to converge, it is difficult to ensure that all devices will be available during the entire training in practice. Moreover, there are typically multiple apps running simultaneously on user devices, competing for already highly constrained hardware resources. As such, it cannot be guaranteed that devices will complete their assigned training tasks in every training round as expected. A similar challenge also arises in cloud based distributed learning due to the increasingly popular usage of preemptive cloud services, where the user process can be interrupted unexpectedly [110].

While many methods have been proposed to mitigate the workload of individual devices, such as weight compression and federated dropout [12][46], they cannot completely remove the possibility that devices are unable to fulfill their training responsibilities, e.g., due to poor wireless connectivity. Thus, in large scale federated learning, many resource-constrained devices have to be excluded from joining federated learning in the first place, which restricts the potential availability of training datasets, and weakens the applicability of federated learning. Furthermore, existing work does not specify how to react when confronting unexpected device behaviors, and also does not analyze the (negative) effects of such behaviors on the training progress.

In this chapter, we relax these restrictions and allow devices to follow more flexible participation patterns. Specifically, the chapter incorporates four situations that are not yet well discussed in the literature: 1) *In-completeness*: devices might submit only partially completed work in a round. 2) *Inactivity*: furthermore, devices might not complete any updates, or respond to the coordinator at all. 3) *Early departures*: in the extreme case, existing devices might quit the training without finishing all training rounds. 4) *Late arrivals*: apart from existing devices, new devices might join after the training has already started.

The difference between inactivity and departure is that inactive devices will temporarily disconnect with the coordinator, but are expected to come back in the near future. In contrast, departing devices will inform the coordinator that they do not plan to rejoin the training. For example, if a user quits the app running federated learning, a message can be sent to the coordinator; the coordinator thus knows who is departing. In the meanwhile, although devices' arriving and departing seem symmetric, they affect the model training differently, and thus require distinct treatments. The key difference is that arriving devices offer extra information about the data distribution, which can be utilized to accelerate the training, while departing devices reduce our available knowledge, thus degrading the applicability of the trained model.

Our approach to improve the flexibility of device participation comprises the following components that supplement the existing **FedAvg** algorithm and handle the challenges brought by flexible device participation.

- **Debiasing for partial model updates**. **FedAvg** aggregates device updates as a weighted sum, with weights that are proportional to the sizes of the local datasets. This choice of aggregation coefficients yields an unbiased gradient as in the centralized setting only when all data points from all devices are equally likely to join the learning [53]. However, it in general fails to guarantee convergence to the globally optimal point in the presence of partial aggregation from incomplete and inactive devices. We show that by *adapting the aggregation coefficients*, the bias can be reduced and the convergence to a global optimum can still be established. Furthermore, our analysis shows the bias originates from the heterogeneity in device participation, as well as from the degree to which local datasets are not IID.
- **Fast-rebooting for device arrivals**. Arriving devices interrupt the training by forcing the model to re-orient to the new device's data, thus slowing the convergence process. In this chapter, we propose to rapidly reboot the training in the case of device arrivals by applying *extra updates* from the new devices. Intuitively, since an arriving device misses all previous epochs, the model training should emphasize more on its updates to compensate. We will rigorously prove this method indeed expedites learning convergence under certain conditions.
- **Redefining model applicability for device departures**. A model successfully trained by federated learning is expected to be applicable to the data from all participating devices. However, when a device withdraws itself from the learning, due to the lack of its future updates, we may no longer require the trained model to perform well on its data. It is then important to redefine the model's applicability. Namely, one can either keep the departing device as a part of the global learning objective, or exclude it to focus only on the remaining devices. The decision depends on which definition yields smaller

training loss. We will show the key to this determination lies in the *remaining training time*.

In Section 5.2, we review relevant literature. In Section 5.3, we give a convergence analysis that incorporates flexible device participation. Based on this analysis, we detail our contributions, as outlined above, in Section 5.4, and we experimentally verify our theoretical results in Section 5.5. Finally we conclude in Section 5.6. The proof of theorems and collieries is shown in Section 5.7.

5.2 Related Work

The **FedAvg** algorithm with non-IID data across devices has been modified in specific edge computing scenarios to reduce the communication overhead [55][83][8] or maintain a good training convergence under a resource budget constraint [99]. However, these works do not consider the possibility that the edge devices can be unavailable during the training process or join at different times, which are the main challenges that we will solve. An online learning framework [16][38][26] is a possible way to enable flexible device participation in the federated learning scenario. For instance, [16] proposes an asynchronous federated learning algorithm to handle unbalanced data that arrives in an online fashion onto different devices. Although the asynchronous aggregation in their proposed algorithm can be naturally applied to randomly inactive devices, the authors do not analyze how their algorithm's convergence is affected by the device inactivity or incompleteness and the data heterogeneity.

In recent years, some attempts have been made to relax the strict training requirements on the participating devices. For example, [102] incorporates heterogeneity of devices into the design of the learning systems; [67] proposes a client selection policy that adapts to the change of devices' hardware status. However, these works do not show how the variations in the devices could affect the convergence of training, nor do they incorporate the heterogeneity of user data into the algorithm design.

In [76] and [97], the authors reveal that incomplete devices can block the convergence, but they consider neither other dynamic participation patterns such as inactivity, arrivals and departures, nor probabilistic models for uncertain device participation. To relieve the impact of incomplete devices, these works propose similar strategies as our method by reweighting the contribution of local models. However, they focus mostly on removing the additional bias term originating from heterogeneous device updates, without looking into how this bias is related to the participation frequency of devices and the divergence among them. They also do not compare the proposed methods with alternative extensions of **FedAvg**. In this chapter, we model the device participation as random variables and incorporate them into the convergence analysis, and we compare the convergence rates for three reasonable aggregation schemes.

5.3 Convergence Analysis

In this section, we establish a convergence bound for federated learning with flexible device participation patterns. Our analysis generalizes the standard **FedAvg** to incorporate arbitrary aggregation coefficients. In the aggregation step, all devices are counted even if they cannot finish all local epochs. The analysis considers a non-IID data distribution and heterogeneous devices, i.e., some devices can be more stable than the others. We first derive the convergence bound with incomplete and inactive devices in Sections 5.3.1 to 5.3.2, and then discuss arrivals and departures in Section 5.3.3.

5.3.1 Algorithm Description

Suppose there are N devices, where each device k defines a local objective function $F_k(w)$. Here w represents the parameters of the machine learning model to be optimized, and $F_k(w)$ may be defined as the average empirical loss over all data points at device k, as in typical federated learning frameworks [61]. The global objective is to minimize $F(w) = \sum_{k=1}^{N} p^k F_k(w)$, where $p^k = \frac{n_k}{n}$, n_k is the number of data points device k owns, and $n = \sum_{k=1}^{N} n_k$. Let w^* be the minimizer of F, and denote by F_k^* the minimum value of F_k. We quantify the degree to which data at each device k is distributed differently than that at other devices as $\Gamma_k = F_k(w^*) - F_k^*$ to capture that data distributions at different devices are non-IID, and let $\Gamma = \sum_{k=1}^{N} p^k \Gamma_k$ as in [53].

We consider discrete time steps $t = 0, 1, \ldots$. Model weights are synchronized when t is a multiple of E, i.e., each round consists of E time steps. Assume there are at most T rounds. For each round (say the τth round), the following three steps are executed:

- Synchronization: the coordinator broadcasts the latest global weight $w_{\tau E}^{\mathcal{G}}$ to all devices. Each device updates its local weight so that: $w_{\tau E}^{k} = w_{\tau E}^{\mathcal{G}}$
- Local updates: each device runs stochastic gradient descent (SGD) on F_k for $i = 0, \ldots, s_\tau^k - 1$:[1]

  $$w_{\tau E+i+1}^{k} = w_{\tau E+i}^{k} - \eta_\tau g_{\tau E+i}^{k} \tag{5.1}$$

  Here η_τ is a staircase learning rate that decays with τ, $0 \leq s_\tau^k \leq E$ represents the number of local updates this device completes in this round, $g_t^k = \nabla F_k(w_t^k, \xi_t^k)$ is the stochastic gradient at device k, and ξ_t^k is a mini-batch sampled from device k's local dataset. We also define $\bar{g}_t^k = \nabla F_k(w_t^k)$ as the full batch gradient at device k, hence $\bar{g}_t^k = \mathbb{E}_{\xi_t^k}[g_t^k]$.
- Aggregation: the coordinator aggregates the gradients and generates the next global weight as

  $$\begin{aligned} w_{(\tau+1)E}^{\mathcal{G}} &= w_{\tau E}^{\mathcal{G}} + \sum\nolimits_{k=1}^{N} p_\tau^k (w_{\tau E + s_\tau^k} - w_{\tau E}^{\mathcal{G}}) \\ &= w_{\tau E}^{\mathcal{G}} - \sum\nolimits_{k=1}^{N} p_\tau^k \sum\nolimits_{i=0}^{s_\tau^k} \eta_\tau g_{\tau E+i}^{k} \end{aligned} \tag{5.2}$$

We define that a device k is *inactive* in round τ if $s_\tau^k = 0$ (i.e., it completes no local updates), and say it is *incomplete* if $0 < s_\tau^k < E$. We treat each s_τ^k as a random variable that can follow an arbitrary distribution. Devices are *heterogeneous* if they have different distributions of s_τ^k, and otherwise they are *homogeneous*. We allow the aggregation coefficients p_τ^k to vary with τ. In Section 5.4, we will discuss different schemes of choosing p_τ^k and their impacts on the convergence.

As a special case, traditional **FedAvg** assumes all selected devices can complete all E local epochs, so that $s_\tau^k \equiv E$. Also, **FedAvg** with full device participation uses fixed aggregation coefficients $p_\tau^k \equiv p^k$, so that the right hand side of (5.2) can be written as $\sum_{k=1}^{N} p^k w_{\tau E}^k$, i.e., aggregating gradients is equivalent to aggregating the model parameters directly.

---

[1] While some papers define local epochs and local updates separately, we use them interchangeably in this chapter. Both refer to the times (5.1) is conducted in a global round.

5.3.2 General Convergence Bound

The analysis relies on the following five assumptions. The first four are standard [53]. The last assumption ensures bounded aggregation coefficients and is satisfied by all schemes discussed in Section 5.4. In Section 5.5, we experimentally show that our proposed learning algorithm performs well even when some assumptions (like strong convexity) are violated.

Assumption 5.1. *$F_1, \dots, F_N$ are all L-smooth, so that F is also L-smooth.*

Assumption 5.2. *$F_1, \dots, F_N$ are all μ-strongly convex, so that F is also μ-strongly convex.*

Assumption 5.3. *The variance of the stochastic gradients is bounded: $\mathbb{E}_\xi \|g_t^k - \bar{g}_t^k\|^2 \leq \sigma_k^2$, $\forall k, t$.*

Assumption 5.4. *The expected squared norm of the stochastic gradients at each local device is uniformly bounded: $\mathbb{E}_\xi \|g_t^k\|^2 \leq G^2$ for all k and t.*

Assumption 5.5. *There exists an upper bound $\theta > 0$ for the aggregation coefficient: $p_\tau^k / p^k \leq \theta, \forall k$.*

Assume the following expectations exist and do not vary with time: $\mathbb{E}[p_\tau^k]$, $\mathbb{E}[p_\tau^k s_\tau^k]$, $\mathbb{E}[(p_\tau^k)^2 s_\tau^k]$, $\mathbb{E}[(\sum_{k=1}^N p_\tau^k - 2)_+ (\sum_{k=1}^N p_\tau^k s_\tau^k)]$ for all rounds τ and devices k, and assume $\mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k] \neq 0$. Intuitively, this last assumption ensures that some updates are aggregated in each round, otherwise this round can be simply omitted. Generally, p_τ^k's are functions of s_τ^k, and these expectations can be estimated from device histories. Let $z_\tau \in \{0, 1\}$ indicate the event that the ratio $\mathbb{E}[p_\tau^k s_\tau^k]/p^k$ does not take the same value for all k. We can obtain the following convergence bound for general p_τ^k:

Theorem 5.1. *By choosing the learning rate $\eta_\tau = \frac{16E}{\mu \mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k]} \frac{1}{\tau E + \gamma}$, we can obtain*

$$\mathbb{E}\|w_{\tau E}^{\mathcal{G}} - w^*\|^2 \leq \frac{M_\tau D + V}{\tau E + \gamma} \tag{5.3}$$

Here we define $\gamma = \max\left\{\frac{32E(1+\theta)L}{\mu \mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k]}, \frac{4E^2\theta}{\mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k]}\right\}$, $M_\tau = \sum_{t=0}^{\tau-1} \mathbb{E}[z_t]$, $D = \frac{64E \sum_{k=1}^N \mathbb{E}[p_\tau^k s_\tau^k]\Gamma_k}{\mu \mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k]}$, $V = \max\left\{\gamma^2 \mathbb{E}\|w_0^{\mathcal{G}} - w^*\|^2, \left(\frac{16E}{\mu \mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k]}\right)^2 \frac{\mathbb{E}[B_\tau]}{E}\right\}$, $B_\tau = 2(2 + \theta)L \sum_{k=1}^N p_\tau^k s_\tau^k \Gamma_k + \left(2 + \frac{\mu}{2(1+\theta)L}\right) E(E-1)G^2 \left(\sum_{k=1}^N p_\tau^k s_\tau^k + \theta(\sum_{k=1}^N p_\tau^k - 2)_+ \sum_{k=1}^N p_\tau^k s_\tau^k\right) + 2EG^2 \sum_{k=1}^N \frac{(p_\tau^k)^2}{p^k} s_\tau^k + \sum_{k=1}^N (p_\tau^k)^2 s_\tau^k \sigma_k^2$

Theorem 5.1 shows that the convergence rate is affected by the aggregation coefficients p_τ^k's as they determine M_τ, D, and V. From (5.3), $w_{\tau E}^{\mathcal{G}}$ will eventually converge to a globally optimal solution only if M_τ increases sub-linearly with τ. In the original full-participation **FedAvg**, $p_\tau^k s_\tau^k \equiv p^k E$, so $z_\tau \equiv 0$ and $M_\tau \equiv 0$ as per the definitions. Thus, full-participation **FedAvg** converges according to (5.3), which is consistent with [53]. However, when considering flexible device participation, M_τ may increase with τ, which can cause **FedAvg** to converge to an arbitrary suboptimal point. The magnitude of M_τ is determined by the degree of heterogeneity in the device participation, and D is bounded by the non-IID metric Γ_k of local datasets. If M_τ increases linearly with τ (e.g., due to device departures), the model will converge to a suboptimal point with the loss bounded by $\frac{D}{E}$. As we will see in Section 5.4.1, by smartly choosing the aggregation coefficients p_τ^k, the increase of M_τ can be controlled and a convergence to the global optimum can still be established.

While we only show results for s_τ^k whose distributions are static with time, Theorem 5.1 can be easily extended to time-varying distributed s_τ^k by replacing the corresponding expectations of $p_\tau^k s_\tau^k$ and $(p_\tau^k)^2 s_\tau^k$ with their minimum or maximum expectations over τ.

5.3.3 Shifts in the Global Objective

Recall the global objective is $F(w) = \sum_{k \in \mathcal{C}} p^k F_k(w)$, i.e., an average of local objectives for participating devices $\mathcal{C}$. A well trained model w^* is expected to perform well on all data points generated by devices in $\mathcal{C}$. In the presence of departing and arriving devices, $\mathcal{C}$ may shrink or expand dynamically during the training. The global objective thus varies accordingly. For example, after admitting an incoming device l with n_l data points: $\tilde{\mathcal{C}} \leftarrow \mathcal{C} + \{l\}$, the global objective becomes $\tilde{F}(w) = \tilde{p}^l F_l(w) + \sum_{k \in \mathcal{C}} \tilde{p}^k F_k(w)$, where $\tilde{p}^k = \frac{n_k}{n_{\mathcal{C}} + n_l}$. The model $\tilde{w}^*$ fully trained with this objective is then applicable to the new data from device l. We formally define *objective shift* as the process of changing the global objective, and the applicability of the trained model, by adding or removing devices from $\mathcal{C}$.

The following theorem bounds the offset between the global optima due to the objective shift. As we can intuitively expect, the difference reduces when the data becomes more IID ($\Gamma_l \to 0$), and when the departing/arriving device owns fewer data points ($n_l \to 0$):

Theorem 5.2. *Suppose a device l arrives/departs, and let n be the total number* *of data points originally. Consider the objective shift* $F \to \tilde{F}$*,* $w^* \to \tilde{w}^*$*. Let* $\tilde{\Gamma}_k = F_k(\tilde{w}^*) - F_k^*$ *quantify the degree of non-IID with respect to the new objective. Then in the arrival case*

$$\|w^* - \tilde{w}^*\| \leq \frac{2\sqrt{2L}}{\mu} \frac{n_l}{n + n_l} \sqrt{\Gamma_l} \tag{5.4}$$

and in the departure case

$$\|w^* - \tilde{w}^*\| \leq \frac{2\sqrt{2L}}{\mu} \frac{n_l}{n} \sqrt{\tilde{\Gamma}_l} \tag{5.5}$$

Objective shift is mandatory when a new device (say device l) arrives: Unless $F_l \equiv F$ (which is highly unlikely), incorporating updates from l will always move $F(w)$ away from F^*. The best strategy without objective shift is then not to aggregate updates from l, and thus not to admit l into the learning process in the first place. In contrast, objective shift is optional when devices depart: we can keep the original objective F even if we will no longer receive updates from a departing device, if doing so yields smaller training loss.

Suppose an objective shift occurs at τ_0. The remainder of the training is then equivalent to starting over from $w^{\mathcal{G}}_{\tau_0 E}$ but converging towards the new objective $\tilde{w}^*$. Combining Theorems 5.1 and 5.2, we can obtain the following convergence bound after the objective shifts:

Corollary 5.1. *Assume the objective shifts at* τ_0 *with* $\mathbb{E}\|w^{\mathcal{G}}_{\tau_0 E} - w^*\|^2 \leq \Delta_{\tau_0}$*. By increasing the learning rate back to* $\eta_\tau = \frac{16E}{\mu \mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k]} \frac{1}{(\tau - \tau_0)E + \gamma}$ *for* $\tau > \tau_0$*, the convergence to the new objective can be bounded by*

$$\mathbb{E}\|w^{\mathcal{G}}_{\tau E} - \tilde{w}^*\|^2 \leq \frac{\tilde{M}_\tau \tilde{D} + \tilde{V}}{(\tau - \tau_0)E + \tilde{\gamma}} \tag{5.6}$$

Here $\tilde{M}_\tau, \tilde{D}, \tilde{V}, \tilde{\gamma}$ *are defined analogously to* M_τ*,* D*,* V*,* γ *but they respectively include/exclude the arriving/departing device. The first term in* $\tilde{V}$ *equals* $\tilde{\gamma}^2(\sqrt{\Delta_{\tau_0}} + \|w^* - \tilde{w}^*\|)^2 = O\left(\frac{V}{\tau_0 E + \gamma} + \Gamma_l\right)$*.*

The increase of the learning rate after the objective shift is necessary. Intuitively, if the shift happens at a large time τ_0 when $w^{\mathcal{G}}_{\tau_0 E}$ is close to the old optimal w^* and η_{τ_0} is close to zero, the learning rate used in Theorem 5.1 will be too small to steer the model to the new optimum, since $\|w^{\mathcal{G}}_{\tau_0 E} - \tilde{w}^*\| \approx \|w^* - \tilde{w}^*\|$.

Comparing (5.3) and (5.6), an objective shift yields an one-time increase in the loss, which forces us to take actions when confronting departures and arrivals. In the case of device departure, it is possible that retaining the old objective can result in a smaller training loss compared to doing a shift. In this situation, the trained model is still applicable to data of the departing device. In the arrival case, though objective shift is mandatory, we can still accelerate the training by a "fast-reboot", applying extra gradient updates from the arriving device.

We will discuss in Section 5.4.2 the fast-reboot method for the arrival case, and in Section 5.4.3 the decision of model applicability for the departure case.

5.4 Main Results

Based on the convergence analysis in Section 5.3, in this section, we present corollaries that can guide operators in reacting to flexible device participation.

5.4.1 Debiasing on Incomplete Aggregation

According to Theorem 5.1, the convergence bound is controlled by the expectation of p^k_τ and its functions. Below we discuss three plausible schemes of choosing p^k_τ, and compare their convergence rates in Table 5.1. The bound for Scheme A assumes there is at least one complete device ($K_\tau \neq 0$), and those for Schemes B, C assume s^k_τ is not trivially zero ($\mathbb{E}[s^k_\tau] \neq 0$). While the three schemes have similar performance in the homogeneous setting, Schemes A and B fail to converge to the global optimum even assuming all devices are active. Scheme C works if inactive devices do not occur in every round ($\sum_t I_t < O(\tau)$).

- **Scheme A**: Only aggregate parameters from devices that complete all E local epochs, with aggregation coefficient $p^k_\tau = \frac{Np^k}{K_\tau} q^k_\tau$, where K_τ is the number of complete devices, $q^k_\tau \in \{0, 1\}$ denotes if client k is complete. If $K_\tau = 0$, this round is discarded.
- **Scheme B**: Allow clients to upload incomplete work (with $s^k_\tau < E$ updates), with fixed aggregation coefficient $p^k_\tau = p^k$.

- **Scheme C**: Accept incomplete works as in Scheme B, with adaptive $p_\tau^k = \frac{E}{s_\tau^k} p^k$, or $p_\tau^k = 0$ if $s_\tau^k = 0$.

Schemes A and B are natural extensions of **FedAvg**. Scheme C assigns a greater aggregation coefficient to devices that complete fewer local epochs. Though this idea seems counter-intuitive, as fewer local updates might lead to less optimal parameters (cf. Table 5.1), it turns out to be the only scheme that guarantees convergence when device participation is heterogeneous.

Corollary 5.2. *Let K_τ be the number of devices that run all E epochs, I_τ indicate the appearance of any inactive devices in round τ, and write $\bar{\sigma}_N^2 \equiv \sum_k^N (p^k \sigma_k)^2$. Table 5.1 gives the convergence rates of Schemes A, B, C when device updates may be incomplete and inactive.*

Table 5.1: Convergence rates with incomplete and inactive devices

	Homogeneous	**Heterogeneous**
A	$O\left(\frac{\mathbb{E}[\frac{N^2}{K_\tau}]+\bar{\sigma}_N^2+\Gamma}{\tau}\right)$	$\leq \frac{D}{E}$
B	$O\left(\frac{\bar{\sigma}_N^2+\Gamma}{\tau\mathbb{E}[s_\tau]}\right)$	$\leq \frac{D}{E}$
C	$O\left(\frac{\bar{\sigma}_N^2+\Gamma}{\tau(\mathbb{E}[1/s_\tau])^{-1}}\right)$	$O\left(\frac{\sum_{t=0}^{\tau-1} I_t D+\sum_k^N (p^k\sigma_k)^2\mathbb{E}\left[\frac{1}{s_\tau^k}\right]+\Gamma}{\tau}\right)$

The reason for enlarging the aggregation coefficients in Scheme C can be understood by observing from (5.2) that increasing p_τ^k is equivalent to increasing the learning rate of device k. Thus, by assigning devices that run fewer epochs a greater aggregation coefficient, these devices effectively run further in each local step, compensating for the additional epochs other devices completed. Figure 5.1 is a snapshot of a typical aggregation round. The bottom two devices completed all $E = 5$ local epochs, while the top two completed only 3 and 4 epochs. Scheme C enlarges the incomplete gradients by respectively 5/3 and 5/4 and produces unbiased aggregation results. Scheme C ensures an unbiased gradient after aggregation, while Schemes A and B will favor devices that run more epochs. Ideally, allowing devices to adapt learning rates by themselves would effectively lead to the same result. However, when a device is running local updates, it may not yet know or be able to estimate the number of local epochs it will complete. In contrast, centralized intervention can make accurate adjustments a posterior.

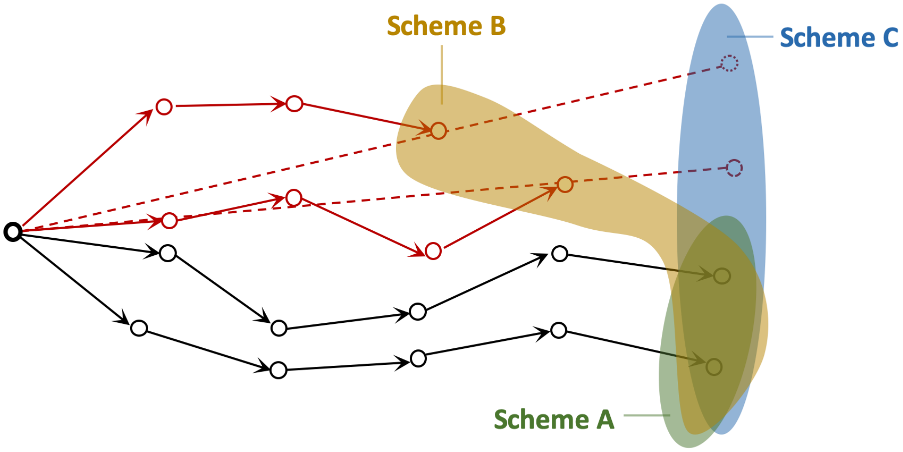


Figure 5.1: Snapshot of one aggregation round

Table 5.1 also reveals how the following system and statistical factors affect the convergence asymptotically:

- The non-IID metric Γ is the major obstacle of convergence in the homogeneous case. In the heterogeneous setting, the D term (which grows with Γ) dominates the training loss. It controls the maximum non-diminishing loss D/E of Scheme A and B, and decelerates the training of Scheme C in the presence of inactive devices.
- Devices' activeness s_τ^k and K_τ contribute inversely to the training loss: The more devices participate, the faster the loss decays. When inactivity occurs frequently, Scheme C cannot converge either. E.g., if a device never responds to the coordinator (so $I_t \equiv 1$), its training loss can never converge to zero.
- The variance $\bar{\sigma}_N, \sigma_k$ in the stochastic gradient descent algorithm slows down the training as expected.

5.4.2 Fast-rebooting on Arrivals

Intuitively, when a device l arrives, $\tilde{w}^*$ will be "dragged" towards its local optimum w_l^*. The gradients from device l may thus encode more information about the new optimum $\tilde{w}^*$ compared to those from the other devices. Thus, by adding an extra update $-\delta^l \nabla F_l(w^{\mathcal{G}}), \delta^l > 0$ to the gradient aggregation, it is likely that w can move closer to $\tilde{w}^*$, allowing the training to fast-reboot from the point of arrivals. However, as shown in Figure 5.2, this intuition may not hold: it is also possible that

$-\delta^l \nabla F_l(w^{\mathcal{G}})$ ends up driving $w^{\mathcal{G}}$ away from $\tilde{w}^*$. In fact, the success of this method is determined by the distance to the old optimum $b = \|w^{\mathcal{G}} - w^*\|$. When b is small, applying an extra update to $w^{\mathcal{G}}$ following the direction $-\nabla F_l(w^{\mathcal{G}})$ moves it closer to $\tilde{w}^*$ ($d_2 < d_1$). However, for a large b, the extra update may on the contrary enlarge this distance ($d_2 > d_1$). We formalize this statement in Corollary 5.3.

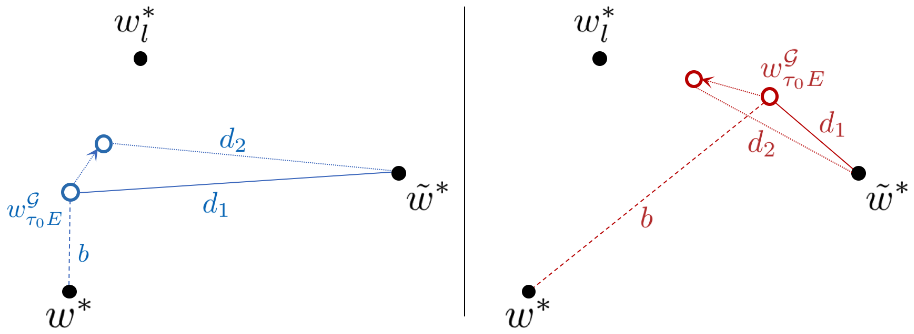


Figure 5.2: The effect of an extra update on the trained model

Corollary 5.3. *Assume* $\nabla F(w)$ *is continuous, and* $0 < \|\nabla F(w)\|_2, \|\nabla^2 F(w)\|_2 \leq W$ *for any* w *(The latter is the induced* l_2 *norm for matrices). Let* $w' = w - \delta^l \nabla F_l(w)$*, then there exists a* $\delta^l > 0$ *such that* $\|w' - \tilde{w}^*\| < \|w - \tilde{w}^*\|$ *if* w *satisfies*

$$\|w - w^*\| < \frac{\tilde{F}(w^*) - \tilde{F}(\tilde{w}^*)}{\left(\frac{2\sqrt{2L}}{\mu}\tilde{p}^l\sqrt{\Gamma_l} + 1\right)\tilde{p}^l W} \tag{5.7}$$

(5.7) defines a sphere around the original global optimum w^* within which the extra update helps fast-reboot. The radius of the sphere depends on the divergence between the new (arriving) and old data points. Generally, the longer the training has elapsed, the closer the global model is to w^*. Thus, the extra updating works best for devices that arrive late in the training.

When applied in practice, the extra updating can be conducted on-the-fly, by augmenting the aggregation coefficient of the arriving device so that $p^l_\tau = p^l + \delta^l$. Furthermore, the distance b can be estimated by the gradient norm with respect to the original objective.

As the name suggests, fast-reboot only accelerates the training for a certain duration after the device arrives. In fact, if there are no future interrupts, models with or without fast-rebooting eventually converge to the same global optimum. Nevertheless, fast-reboot is still beneficial if there is insufficient training time remaining (e.g., a device arrives near the end of the training).

5.4.3 Redefining Applicability on Departures

As is discussed in Section 5.3.3, when a device leaves, we need to redefine the applicability of the trained model. Namely, one can decide to either exclude this departing device and shift the objective, or keep including it and stick to the old objective. The decision depends on the time at which the device leaves. When including the device as a part of the global objective, from (5.3), since $M_\tau = \tau - \tau_0$ from then on, the training loss will always exceed a structural bias D/E. In contrast, if the device is excluded and the model is trained with a shifted global objective, there will be an immediate increase in the convergence bound as in Theorem 5.2. But afterwards, the bound will decrease and eventually the parameters will converge to the new global optimum.

Assume a device leaves at $\tau_0 < T$ and there are no subsequent arrivals/departures. Let $f_0(\tau)$ be the convergence bound if we include the device, and $f_1(\tau)$ be the bound if it is excluded. We can obtain $f_0(\tau) = \frac{(\tau-\tau_0)D+V}{\tau E+\gamma}$, $f_1(\tau) = \frac{\tilde{V}}{(\tau-\tau_0)E+\tilde{\gamma}}$. Here $\tilde{M}_\tau, \tilde{V}_\tau, \tilde{\gamma}$ are defined analogously to M_τ, V_τ, γ but they exclude the departing device. A device is excluded if by doing so, a smaller training loss can be obtained at the deadline T, which is summarized in the following corollary:

Corollary 5.4. *Excluding a device that departs at τ_0 leads to smaller training loss if*

$$\min_{\tau \geq \tau_0} f_0(\tau) \geq f_1(T) \tag{5.8}$$

Further assume $\tilde{\gamma} = \gamma$, and $\tilde{V}$ is dominated by its first term so that $\tilde{V} = \frac{V}{\tau_0 E+\gamma} + \Gamma_l$. (5.8) then becomes

$$T - \tau_0 \geq O\left(\sqrt{\Gamma_l \tau_0}\right) \tag{5.9}$$

From (5.9), when the remaining training time $T - \tau_0$ is at least $O(\sqrt{\Gamma_l \tau_0})$, applying the trained model to the departing device becomes less promising. It is thus better to exclude it and shift the objective. As we can expect, the bound grows with Γ_l, since the non-IID contribution from the departing device increases the initial $\tilde{V}$. As τ_0 increases, the learning rate without shift gets smaller, mitigating the increase of the training loss from departing devices.

5.5 Experiments

In this section, we experimentally evaluate Section 5.4's results. Due to the limitations on hardware resources, the training process is performed in computer simulations. To ensure the simulation is consistent with the real learning environment, we use real-world traces to represent the participation patterns of simulated devices. We present our experiment setup in Section 5.5.1, and verify our theory results in Sections 5.5.2 - 5.5.4.

5.5.1 Experiment Setup

We create various data traces to represent the heterogeneous participation patterns of local devices. We set up a simple federated learning experiment with five Raspberry PIs as workers, and a desktop server as the coordinator. Each PI has a training process that runs the original **FedAvg** algorithm, and a competitor process doing CPU-intensive work simultaneously. We manually tune the workload of the competitor process so that it takes up 0%, 30%, 50%, 70%, 90% of the PI's CPU resources, simulating different device configurations in federated learning. Under the five settings, for each round, we record the percentage of required epochs the PI ends up submitting before a preset, fixed deadline. Due to the default load-balancing behavior of the operating system's CPU scheduler, these traces do not contain zero epochs (i.e. inactive cases). To generate inactive device participation patterns, we create another set of three traces with respectively low, medium and high bandwidth. Devices can thus be inactive due to weak transmission. Table 5.2 shows the mean and standard deviation of the percentage of epochs completed for each trace. The first five traces do not contain inactive cases. In the following experiments, each simulated device is randomly assigned a trace. For each aggregation round τ, it randomly samples from its trace to obtain the number of local epochs s_τ^k.

Table 5.2: The means and standard deviations for the percentage of required local epochs actually submitted to the coordinator during the training

Name	$\mathcal{T}_0$	$\mathcal{T}_{30}$	$\mathcal{T}_{50}$	$\mathcal{T}_{70}$	$\mathcal{T}_{90}$	$\mathcal{T}_{hi}$	$\mathcal{T}_{mi}$	$\mathcal{T}_{lo}$
Mean	100	75.3	67.2	57.2	56.3	82.5	74.1	51.2
Stdev	0	14.8	11.3	11.7	14.8	23.3	22.3	18.3

Three datasets are used in the experiments: MNIST [48], EMNIST [23] and SYNTHETIC(α, β) [52]. We build a two-layer MLP model and a two-convolution-layer CNN model respectively for MNIST and EMNIST, both models are defined by [61]. For SYNTHETIC(α, β), we use an ordinary logistic regression model. All models use the vanilla SGD as local optimizers, with batch sizes of 10 for MNIST and EMNIST, and 20 for SYNTHETIC. When generating non-IID data, we sort the MNIST and EMNIST data by labels so that each device is assigned data from one label chosen uniformly at random. For SYNTHETIC(α, β), we vary the parameters α, β from 0 to 1. The larger α, β are, the less IID the dataset becomes. We use the staircase learning rate $\eta_\tau = \eta_0/\tau$ as adopted in our convergence analysis. The initial η_0 is 2e-3 for MNIST, 5e-4 for EMNIST, and 1 for SYNTHETIC(α, β). Unless otherwise noted, the number of samples at each device follows the Type-I Pareto distribution with the Pareto index of 0.5.

5.5.2 Comparison of Aggregation Schemes

We first examine the effects of the device heterogeneity and the non-IID data distributions on the convergence for each aggregation scheme. We conduct eight sets of experiments where we incrementally increase the number of participation traces to reflect the increasing heterogeneity in device participation. For SYNTHETIC, we use $\alpha = \beta = 0$ for the IID case, and $\alpha = \beta = 1$ for the non-IID case. We train on 100 devices for MNIST, 62 devices for EMNIST (by merge), and 50 devices for SYNTHETIC(α, β). Table 5.3 records the differences in the test accuracies between different aggregation schemes after 200 global epochs, where $|\mathcal{T}| = j$ represents using the first j traces in Table 5.2. The typical convergence process is depicted in Figure 5.3. Plots from left to right correspond to $|\mathcal{T}| = 1, 3, 5, 8$. (increasing device heterogeneity)

As we can see, Scheme C yields the best test accuracy on average. Compared to Schemes A and B, it achieves higher accuracy when devices get more heterogeneous and less IID. This is consistent with our loss bounds in Table 5.1, since Schemes A and B fail to converge to the global optimum in the heterogeneous case with non-IID data. On the other hand, Scheme A performs extremely badly with large $|\mathcal{T}|$. This is because the last few traces contain very few complete rounds, significantly increasing $\mathbb{E}[1/K_\tau]$. Noteworthily, Scheme C is no different from, or even worse than Scheme B

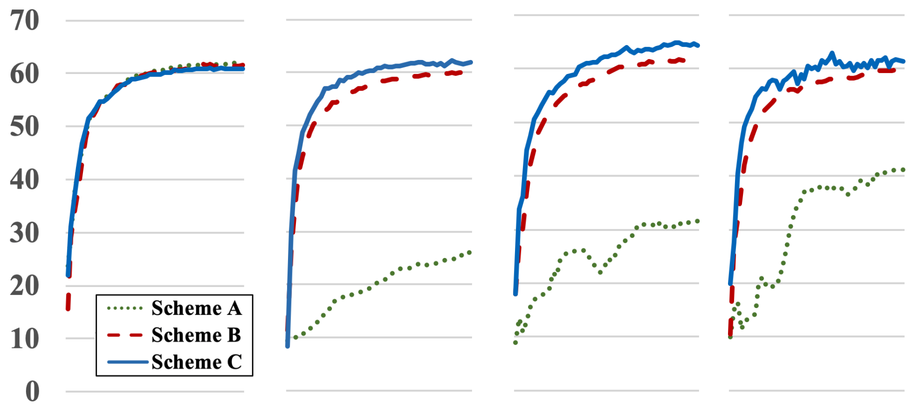


Figure 5.3: Test accuracy for non-IID EMNIST

in more homogeneous settings, this is consistent with Table 5.1 since $\frac{1}{\mathbb{E}[s_\tau]} \leq \mathbb{E}[\frac{1}{s_\tau}]$. When the traces contain inactive devices ($|\mathcal{T}| \geq 6$), Scheme C becomes less stable due to the variance introduced by I_t in Corollary 5.2.

5.5.3 Effectiveness of Fast-Reboot

We now investigate the effectiveness of the fast-reboot method described in Section 5.4.2. The experiments involve $N-1$ existing devices, and the arriving device joins at τ_0. As is discussed in Section 5.4.2, the method makes no difference when data distribution is IID. We thus only consider non-IID cases. We set $N = 10$ for MNIST and EMNIST (balanced) and $N = 30$ for SYNTHETIC$(1,1)$. To avoid the interference brought by inactive devices, for this experiment we only use the first five traces in Table 5.2, and we adopt Scheme C as the aggregation method. All devices are given the same number of samples for fair comparison.

When the device arrives, we increase the learning rate to $\eta_0/(\tau - \tau_0)$. The aggregation coefficient of the arriving device l is boosted to $p^l_\tau = 3p^l$ initially, and decays to p^l by $O(\tau^{-2})$. Table 5.4 records the number of global epochs it takes to recover to the accuracy level before the arrival. The left cells are for fast reboot and the right for vanilla reboot. Fast-reboot consistently achieves faster rebound, and works better for late arrivals as we expect. EMNIST-CNN enjoys less improvement from fast-reboot because CNN models converge more slowly than MLP and logistic regression models. Thus, at the moment new devices arrive, EMNIST models have not fully converged to the old optima, degrading the effectiveness of fast-reboot

Table 5.3: The % improvement in the test accuracies of Scheme B w.r.t. Schemes A(left numbers) and Scheme C w.r.t. Scheme B (right numbers)

(a) MNIST Data

$\|\mathcal{T}\|$	1	2	3	4
IID	-0.6 | 0.3	1.9 | 0.1	5.6 | 0.1	8.3 | 0.7
NIID	0.2 |-0.3	9.5 | 1.8	19.3 | 1.6	33.8 | 3.3
$\|\mathcal{T}\|$	5	6	7	8
IID	10.4 | 2.6	14.0 | 2.2	5.8 | 1.9	11.8 | 2.4
NIID	33.2 | 3.2	28.7 | 6.2	36.0 | 3.6	43.4 | 6.9

(b) EMNIST Data

$\|\mathcal{T}\|$	1	2	3	4
IID	0.7 |-0.6	0.9 | 0.1	4.2 | 0.7	4.8 | 1.0
NIID	-0.1 |-0.7	17.0 |-2.0	34.2 | 1.8	37.9 | 4.8
$\|\mathcal{T}\|$	5	6	7	8
IID	6.9 | 1.1	6.6 | 1.2	4.0 | 1.5	7.6 | 1.2
NIID	30.2 | 2.5	22.5 | 3.0	25.3 | 2.2	18.6 | 1.8

(c) SYNTHETIC Data

$\|\mathcal{T}\|$	1	2	3	4
IID	-0.6 | 0.5	2.1 | 0.1	6.6 | 0.0	9.0 | 0.7
NIID	0.1 |-0.4	9.6 | 1.5	22.2 | 1.8	38.2 | 3.2
$\|\mathcal{T}\|$	5	6	7	8
IID	11.6 | 3.0	16.4 | 2.5	6.3 | 1.9	14.4 | 2.8
NIID	33.3 | 3.9	30.5 | 7.9	37.9 | 4.5	41.6 | 8.0

as per Corollary 5.3. The typical fast-reboot process is shown in Figure 5.4. The dashed vertical lines indicate the arriving (departing) time τ_0. After τ_0, except for the "include" option, models are tested with new datasets that include (exclude) holdout data from the arriving (departing) device.

Next we study the situation when multiple devices arrive in a row. Figure 5.5 shows the training process for MNIST data. The test dataset is updated every time a new device arrives to include its holdout data. The vertical dashed lines indicate the time the device arrives. Every time a device arrives, we increase the learning rate as per Corollary 5.1. Initially, seven devices are in the training. After 100 global epochs, the remaining three devices arrive at 50 epoch intervals, without waiting for

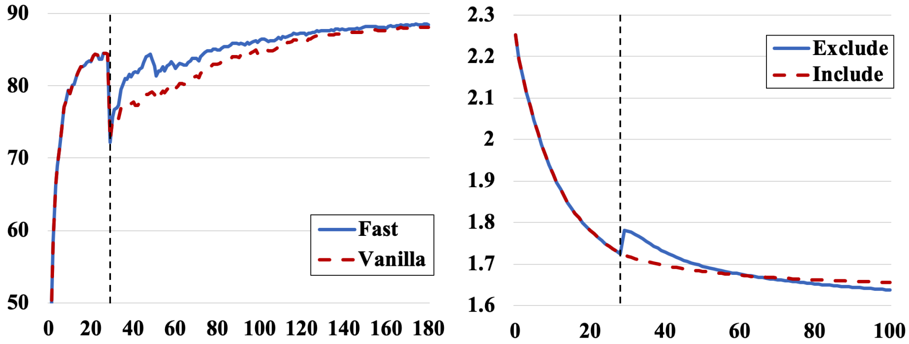


Figure 5.4: Evolution of the test accuracy (left) and loss (right) under device arrival (left, MNIST) and departure (right, SYNTHETIC) cases

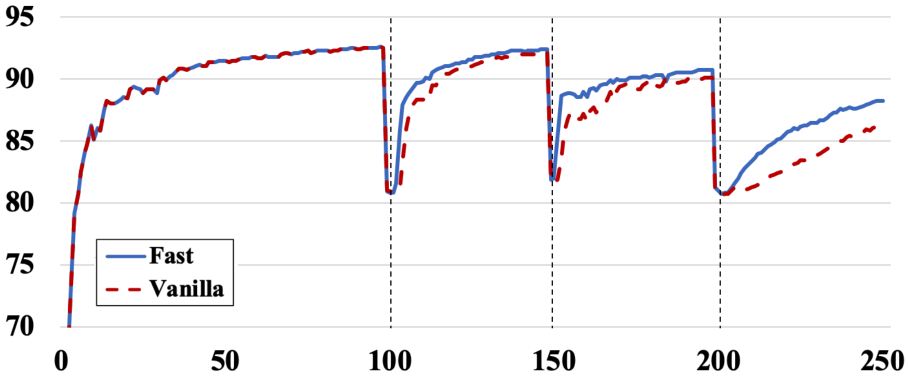


Figure 5.5: Test accuracy with and without fast-reboot for multiple arrivals for non-IID MNIST

the model to fully converge. From Figure 5.5, the fast-reboot trick accelerates the convergence for every device arrival.

5.5.4 Model Applicability upon Departures

The right plot in Figure 5.4 shows the typical change of the test loss after the device departs. We use the same setting as in Section 5.5.3. As is predicted in Section 5.4.3, an objective shift ('exclude') initially increases the test loss. But eventually, the two curves cross and excluding the device becomes more beneficial.

Table 5.5 summarizes the number of global epochs it takes for the curves to cross with SYNTHETIC(α, β). The rows correspond to three choices of parameters (α, β). As we can see, the values increase with τ_0 and the non-IID metric (α, β), confirming Corollary 5.4.

Table 5.4: The number of global epochs after the arriving time τ_0 until the test accuracy bounces back to that at $\tau_0 - 1$

$\boldsymbol{\tau_0}$	10	30	50	70
MNIST	4 | 4	22 |27	55 |63	59 |66
EMNIST	4 | 3	11 |12	14 |19	21 |24
SYNTHETIC	1 | 1	4 | 6	7 |12	3 | 8

Table 5.5: The number of global epochs after the departing time τ_0 until the test losses coincide for including and excluding options

$\boldsymbol{\tau_0}$	10	15	20	25	30	35	40	45	50
(.1, .1)	2	5	3	3	9	3	10	26	40
(.5, .5)	1	3	9	14	13	7	12	36	34
(1., 1.)	10	9	27	18	34	17	28	62	77

5.6 Summary

This chapter extends the federated learning paradigm to incorporate more flexible device participation. The analysis shows that incomplete local device updates can be utilized by scaling the corresponding aggregation coefficients, and a mild degree of device inactivity will not impact the convergence. Further investigation reveals how the convergence relates to heterogeneity in both the data and the device participation. The chapter also proposes techniques to fast-reboot the training after new devices arrive, and provides an analytical criterion on when to exclude a departing device.

5.7 Proof of Theorems and Corollaries

5.7.1 Proof of Theorem 5.1

Equivalent View

For ease of the analysis, we introduce for each client k and each global round τ a sequence of virtual variables $\alpha^k_{\tau E}, \alpha^k_{\tau E+1}, \ldots, \alpha^k_{(\tau+1)E-1}$. Here each $\alpha^k_t \in \{0, 1\}$ and $\sum_{i=0}^{E} \alpha^k_{\tau E+i} = s^k_\tau$. Since s^k_τ is a random variable, α^k_t's are also random variables, and the distributions of α^k_t's determine the distribution of s^k_τ. For example, if $\alpha^k_t \overset{iid}{\sim} \text{Bernoulli}(p)$, then $s^k_\tau \sim \text{Bin}(E, p)$. In general, we do not make any assumption on the distributions and correlations of α^k_t's. Our results are thus valid for any realization of s^k_τ.

With the definition of α^k_t's, we can rewrite (5.1)(5.2) as:

$$w^k_{\tau E+i+1} = w^k_{\tau E+i} - \eta_\tau g^k_{\tau E+i} \alpha^k_{\tau E+i} \tag{5.10}$$

$$w^{\mathcal{G}}_{(\tau+1)E} = w^{\mathcal{G}}_{\tau E} - \sum_{k=1}^{N} p^k_\tau \sum_{i=0}^{E} \eta_\tau g^k_{\tau E+i} \alpha^k_{\tau E+i} \tag{5.11}$$

Note that $w^{\mathcal{G}}_t$ is visible only when t is a multiple of E. To generalize it to arbitrary t, we define $\bar{w}_t$ such that $\bar{w}_0 = w^{\mathcal{G}}_0$, and

$$\bar{w}_{\tau E+i+1} = \bar{w}_{\tau E+i} - \eta_\tau \sum_{k=1}^{N} p^k_\tau g^k_{\tau E+i} \alpha^k_{\tau E+i} \tag{5.12}$$

Note that $\bar{w}_{\tau E+i} = \sum_{k=1}^{N} p^k_\tau w^k_{\tau E+i}$ only if $\sum_{k=1}^{N} p^k_\tau = 1$, which generally does not hold.

Lemma 5.1. *For any τ, $\bar{w}_{\tau E} = w^{\mathcal{G}}_{\tau E}$.*

Proof. We will prove by induction. By definition, $\bar{w}_0 = w^{\mathcal{G}}_0$. Suppose $\bar{w}_{\tau E} = w^{\mathcal{G}}_{\tau E}$, then

$$\begin{aligned}\bar{w}_{(\tau+1)E} &= \bar{w}_{(\tau+1)E-1} - \eta_\tau \sum_{k=1}^{N} p_\tau^k g_{(\tau+1)E-1}^k \alpha_{(\tau+1)E-1}^k \\ &= \cdots = \bar{w}_{\tau E} - \sum_{i=0}^{E-1} \eta_\tau \sum_{k=1}^{N} p_\tau^k g_{\tau E+i}^k \alpha_{\tau E+i}^k \\ &= w_{\tau E}^{\mathcal{G}} - \sum_{k=1}^{N} p_\tau^k \sum_{i=0}^{E-1} \eta_\tau g_{\tau E+i}^k \alpha_{\tau E+i}^k = w_{(\tau+1)E}^{\mathcal{G}}\end{aligned} \tag{5.13}$$

□

Thus, in the following analysis we will just use $\bar{w}_t$ to denote the global weight.

Key Lemmas

We first present a couple of important lemmas:

Lemma 5.2.

$$\mathbb{E}_\xi \| \sum_{k=1}^{N} p_\tau^k (g_t^k - \bar{g}_t^k) \|^2 \leq \sum_{k=1}^{N} (p_\tau^k)^2 \sigma_k^2 \tag{5.14}$$

Proof.

$$\| \sum_{k=1}^{N} p_\tau^k (g_t^k - \bar{g}_t^k) \|^2 = \sum_{k=1}^{N} \| p_\tau^k (g_t^k - \bar{g}_t^k) \|^2 + \sum_{j \neq k} p_\tau^k p_\tau^j \langle g_t^k - \bar{g}_t^k, g_t^j - \bar{g}_t^j \rangle \tag{5.15}$$

Since each client is running independently, the covariance

$$\mathbb{E}_\xi \langle g_t^k - \bar{g}_t^k, g_t^j - \bar{g}_t^j \rangle = 0 \tag{5.16}$$

Thus,

$$\mathbb{E}_\xi \| \sum_{k=1}^{N} p_\tau^k (g_t^k - \bar{g}_t^k) \|^2 = \sum_{k=1}^{N} \mathbb{E}_\xi \| p_\tau^k (g_t^k - \bar{g}_t^k) \|^2 \leq \sum_{k=1}^{N} (p_\tau^k)^2 \sigma_k^2 \tag{5.17}$$

□

Lemma 5.3. *For $i = 0, \cdots, E-1$ and all τ, k*

$$\begin{aligned}
&\mathbb{E}_{\xi}[\sum_{k=1}^{N} p_{\tau}^{k} \|\bar{w}_{\tau E+i} - w_{\tau E+i}^{k}\|^2] \\
&\qquad \leq (E-1)G^2\eta_{\tau}^2 \Big(\sum_{k=1}^{N} p_{\tau}^{k} s_{\tau}^{k} + (\sum_{k=1}^{N} p_{\tau}^{k} - 2)_{+} \sum_{k=1}^{N} \frac{(p_{\tau}^{k})^2}{p^k} s_{\tau}^{k} \Big)
\end{aligned} \tag{5.18}$$

Proof. Note that $w_{\tau E}^{k} = \bar{w}_{\tau E}$ for all k.

$$\begin{aligned}
&\|\bar{w}_{\tau E+i} - w_{\tau E+i}^{k}\|^2 = \|(\bar{w}_{\tau E+i} - \bar{w}_{\tau E}) - (w_{\tau E+i}^{k} - \bar{w}_{\tau E})\|^2 \\
=&\|\bar{w}_{\tau E+i} - \bar{w}_{\tau E}\|^2 - 2\langle \bar{w}_{\tau E+i} - \bar{w}_{\tau E}, w_{\tau E+i}^{k} - \bar{w}_{\tau E}\rangle + \|w_{\tau E+i}^{k} - \bar{w}_{\tau E}\|^2
\end{aligned} \tag{5.19}$$

From (5.10)(5.12),

$$\begin{aligned}
\sum_{k=1}^{N} p_{\tau}^{k} w_{\tau E+i}^{k} &= \sum_{k=1}^{N} p_{\tau}^{k} w_{\tau E+i-1}^{k} - \eta_{\tau} \sum_{k=1}^{N} p_{\tau}^{k} g_{\tau E+i-1}^{k} \alpha_{\tau E+i-1}^{k} \\
&= \sum_{k=1}^{N} p_{\tau}^{k} w_{\tau E+i-1}^{k} + \bar{w}_{\tau E+i} - \bar{w}_{\tau E+i-1} \\
&= \cdots = \sum_{k=1}^{N} p_{\tau}^{k} w_{\tau E}^{k} + \bar{w}_{\tau E+i} - \bar{w}_{\tau E}
\end{aligned} \tag{5.20}$$

Thus,

$$\begin{aligned}
&-2\sum_{k=1}^{N} p_{\tau}^{k} \langle \bar{w}_{\tau E+i} - \bar{w}_{\tau E}, w_{\tau E+i}^{k} - \bar{w}_{\tau E}\rangle \\
=&-2\langle \bar{w}_{\tau E+i} - \bar{w}_{\tau E}, \sum_{k=1}^{N} p_{\tau}^{k} w_{\tau E}^{k} + \bar{w}_{\tau E+i} - \bar{w}_{\tau E} - \sum_{k=1}^{N} p_{\tau}^{k} \bar{w}_{\tau E}\rangle \\
=&-2\|\bar{w}_{\tau E+i} - \bar{w}_{\tau E}\|^2
\end{aligned} \tag{5.21}$$

$$\sum_{k=1}^{N} p_{\tau}^{k} \|\bar{w}_{\tau E+i} - w_{\tau E+i}^{k}\|^2 = (\sum_{k=1}^{N} p_{\tau}^{k} - 2)\|\bar{w}_{\tau E+i} - \bar{w}_{\tau E}\|^2 + \sum_{k=1}^{N} p_{\tau}^{k} \|w_{\tau E+i}^{k} - \bar{w}_{\tau E}\|^2 \tag{5.22}$$

$$
\begin{aligned}
&\|\bar{w}_{\tau E+i}-\bar{w}_{\tau E}\|^2=\|\sum_{j=0}^{i-1}\eta_\tau\sum_{k=1}^{N}p_\tau^k g_{\tau E+j}^k\alpha_{\tau E+j}^k\|^2\\
=&\|\eta_\tau\sum_{k=1}^{N}p_\tau^k\Big(\sum_{j=0}^{i-1}g_{\tau E+j}^k\alpha_{\tau E+j}^k\Big)\|^2=\eta_\tau^2\|\sum_{k=1}^{N}p^k\Big(\frac{p_\tau^k}{p^k}\sum_{j=0}^{i-1}g_{\tau E+j}^k\alpha_{\tau E+j}^k\Big)\|^2 \qquad (5.23)\\
\le&\eta_\tau^2\sum_{k=1}^{N}\frac{(p_\tau^k)^2}{p^k}\|\sum_{j=0}^{i-1}g_{\tau E+j}^k\alpha_{\tau E+j}^k\|^2
\end{aligned}
$$

Here

$$
\begin{aligned}
&\|\sum_{j=0}^{i-1}g_{\tau E+j}^k\alpha_{\tau E+j}^k\|^2\\
=&\sum_{j=0}^{i-1}\|g_{\tau E+j}^k\alpha_{\tau E+j}^k\|^2+2\sum_{p<q}\langle g_{\tau E+p}^k\alpha_{\tau E+p}^k,g_{\tau E+q}^k\alpha_{\tau E+q}^k\rangle\\
\le&\sum_{j=0}^{i-1}\|g_{\tau E+j}^k\alpha_{\tau E+j}^k\|^2+2\sum_{p<q}\|g_{\tau E+p}^k\alpha_{\tau E+p}^k\|\|g_{\tau E+q}^k\alpha_{\tau E+q}^k\| \qquad (5.24)\\
\le&\sum_{j=0}^{i-1}\|g_{\tau E+j}^k\alpha_{\tau E+j}^k\|^2+\sum_{p<q}\Big(\|g_{\tau E+p}^k\alpha_{\tau E+p}^k\|^2+\|g_{\tau E+q}^k\alpha_{\tau E+q}^k\|^2\Big)\\
=&i\sum_{j=0}^{i-1}\|g_{\tau E+j}^k\alpha_{\tau E+j}^k\|^2
\end{aligned}
$$

So

$$\mathbb{E}_\xi\|\sum_{j=0}^{i-1}g_{\tau E+j}^k\alpha_{\tau E+j}^k\|^2\le iG^2\sum_{j=0}^{i-1}\alpha_{\tau E+j}^k\le(E-1)G^2s_\tau^k \qquad (5.25)$$

Plug (5.25) to (5.23) we have

$$\mathbb{E}_\xi\|\bar{w}_{\tau E+i}-\bar{w}_{\tau E}\|^2\le(E-1)G^2\eta_\tau^2\sum_{k=1}^{N}\frac{(p_\tau^k)^2}{p^k}s_\tau^k \qquad (5.26)$$

Similarly

$$\begin{aligned}\mathbb{E}_\xi \sum_{k=1}^N p_\tau^k \|w_{\tau E+i}^k - \bar{w}_{\tau E}\|^2 =& \mathbb{E}_\xi \sum_{k=1}^N p_\tau^k \|\eta_\tau \sum_{j=0}^{i-1} g_{\tau E+j}^k \alpha_{\tau E+j}^k\|^2 \\ \leq& (E-1)G^2\eta_\tau^2 \sum_{k=1}^N p_\tau^k s_\tau^k\end{aligned} \tag{5.27}$$

Plug (5.26)(5.27) to (5.22) we have

$$\begin{aligned}&\mathbb{E}_\xi[\sum_{k=1}^N p_\tau^k \|\bar{w}_{\tau E+i} - w_{\tau E+i}^k\|^2] \\ &\leq (E-1)G^2\eta_\tau^2 \Big(\sum_{k=1}^N p_\tau^k s_\tau^k + (\sum_{k=1}^N p_\tau^k - 2)_+ \sum_{k=1}^N \frac{(p_\tau^k)^2}{p^k} s_\tau^k\Big)\end{aligned} \tag{5.28}$$

□

Bounding $\|\bar{w}_{\tau E+i+1} - w^*\|^2$

$$\begin{aligned}&\|\bar{w}_{\tau E+i+1} - w^*\|^2 \\ =& \|\bar{w}_{\tau E+i} - \eta_\tau \sum_{k=1}^N p_\tau^k \alpha_{\tau E+i}^k g_{\tau E+i}^k - w^* \\ &- \eta_\tau \sum_{k=1}^N p_\tau^k \alpha_{\tau E+i}^k \bar{g}_{\tau E+i}^k + \eta_\tau \sum_{k=1}^N p_\tau^k \alpha_{\tau E+i}^k \bar{g}_{\tau E+i}^k\|^2 \\ =& \underbrace{\|\bar{w}_{\tau E+i} - w^* - \eta_\tau \sum_{k=1}^N p_\tau^k \alpha_{\tau E+i}^k \bar{g}_{\tau E+i}^k\|^2}_{A_1} + \eta_\tau^2 \|\sum_{k=1}^N p_\tau^k \alpha_{\tau E+i}^k (\bar{g}_{\tau E+i}^k - g_{\tau E+i}^k)\|^2 \\ &+ \underbrace{2\eta_\tau \langle \bar{w}_{\tau E+i} - w^* - \eta_\tau \sum_{k=1}^N p_\tau^k \alpha_{\tau E+i}^k \bar{g}_{\tau E+i}^k, \sum_{k=1}^N p_\tau^k \alpha_{\tau E+i}^k (\bar{g}_{\tau E+i}^k - g_{\tau E+i}^k)\rangle}_{A_2}\end{aligned} \tag{5.29}$$

Since $\mathbb{E}_\xi[g^k_{\tau E+i}] = \bar{g}^k_{\tau E+i}$, we have $\mathbb{E}_\xi[A_2] = 0$. We then bound A_1.

$$\begin{aligned} A_1 = &\|\bar{w}_{\tau E+i} - w^* - \eta_\tau \sum_{k=1}^{N} p^k_\tau \alpha^k_{\tau E+i} \bar{g}^k_{\tau E+i}\|^2 \\ =&\|\bar{w}_{\tau E+i} - w^*\|^2 \underbrace{-2\eta_\tau \langle \bar{w}_{\tau E+i} - w^*, \sum_{k=1}^{N} p^k_\tau \alpha^k_{\tau E+i} \bar{g}^k_{\tau E+i}\rangle}_{B_1} \\ &+ \underbrace{\eta_\tau^2 \|\sum_{k=1}^{N} p^k_\tau \alpha^k_{\tau E+i} \bar{g}^k_{\tau E+i}\|^2}_{B_2} \end{aligned} \tag{5.30}$$

Since F_k is L-smooth,

$$\|\alpha^k_{\tau E+i} \bar{g}^k_{\tau E+i}\|^2 \leq 2L(F_k(w^k_{\tau E+i}) - F^*_k)\alpha^k_{\tau E+i} \tag{5.31}$$

By the convexity of l_2 norm

$$\begin{aligned} B_2 &= \eta_\tau^2 \|\sum_{k=1}^{N} p^k_\tau \alpha^k_{\tau E+i} \bar{g}^k_{\tau E+i}\|^2 = \eta_\tau^2 \|\sum_{k=1}^{N} p^k (\frac{p^k_\tau}{p^k} \alpha^k_{\tau E+i} \bar{g}^k_{\tau E+i})\|^2 \\ &\leq \eta_\tau^2 \sum_{k=1}^{N} \frac{(p^k_\tau)^2}{p^k} \|\alpha^k_{\tau E+i} \bar{g}^k_{\tau E+i}\|^2 \leq 2L\theta\eta_\tau^2 \sum_{k=1}^{N} p^k_\tau (F_k(w^k_{\tau E+i}) - F^*_k)\alpha^k_{\tau E+i} \end{aligned} \tag{5.32}$$

$$\begin{aligned} B_1 &= -2\eta_\tau \langle \bar{w}_{\tau E+i} - w^*, \sum_{k=1}^{N} p^k_\tau \alpha^k_{\tau E+i} \bar{g}^k_{\tau E+i}\rangle \\ &= -2\eta_\tau \sum_{k=1}^{N} p^k_\tau \langle \bar{w}_{\tau E+i} - w^*, \alpha^k_{\tau E+i} \bar{g}^k_{\tau E+i}\rangle \\ &= -2\eta_\tau \sum_{k=1}^{N} p^k_\tau \langle \bar{w}_{\tau E+i} - w^k_{\tau E+i}, \alpha^k_{\tau E+i} \bar{g}^k_{\tau E+i}\rangle \\ &- 2\eta_\tau \sum_{k=1}^{N} p_k \langle w^k_{\tau E+i} - w^*, \alpha^k_{\tau E+i} \bar{g}^k_{\tau E+i}\rangle \end{aligned} \tag{5.33}$$

Here

$$\begin{aligned}
&-2\langle \bar{w}_{\tau E+i} - w^k_{\tau E+i}, \alpha^k_{\tau E+i}\bar{g}^k_{\tau E+i}\rangle \\
\leq& 2|\langle \bar{w}_{\tau E+i} - w^k_{\tau E+i}, \alpha^k_{\tau E+i}\bar{g}^k_{\tau E+i}\rangle| \\
\leq& 2\alpha^k_{\tau E+i}\|\bar{w}_{\tau E+i} - w^k_{\tau E+i}\|\|\bar{g}^k_{\tau E+i}\| \\
\leq& \Big(\frac{1}{\eta_\tau}\|\bar{w}_{\tau E+i} - w^k_{\tau E+i}\|^2 + \eta_\tau\|\bar{g}^k_{\tau E+i}\|^2\Big)\alpha^k_{\tau E+i}
\end{aligned} \tag{5.34}$$

Since F_k is μ-strongly convex

$$\langle w^k_{\tau E+i} - w^*, \alpha^k_{\tau E+i}\bar{g}^k_{\tau E+i}\rangle \geq \big((F_k(w^k_{\tau E+i}) - F_k(w^*)) + \frac{\mu}{2}\|w^k_{\tau E+i} - w^*\|^2\big)\alpha^k_{\tau E+i} \tag{5.35}$$

Plug (5.34)(5.35) to (5.33)

$$\begin{aligned}
B_1 \leq \sum_{k=1}^{N} p^k_\tau \alpha^k_{\tau E+i}\Bigg(& \|\bar{w}_{\tau E+i} - w^k_{\tau E+i}\|^2 + \eta_\tau^2\|\bar{g}^k_{\tau E+i}\|^2 \\
& - 2\eta_\tau\big((F_k(w^k_{\tau E+i}) - F_k(w^*)) + \frac{\mu}{2}\|w^k_{\tau E+i} - w^*\|^2\big)\Bigg)
\end{aligned} \tag{5.36}$$

Plug (5.32)(5.36) to (5.30)

$$\begin{aligned}
& A_1 \leq \|\bar{w}_{\tau E+i} - w^*\|^2 + 2L\theta\eta_\tau^2\sum_{k=1}^{N} p^k_\tau \alpha^k_{\tau E+i}(F_k(w^k_{\tau E+i}) - F_k^*) \\
& + \sum_{k=1}^{N} p^k_\tau \alpha^k_{\tau E+i}\Bigg(\|\bar{w}_{\tau E+i} - w^k_{\tau E+i}\|^2 + \underbrace{\eta_\tau^2\|\bar{g}^k_{\tau E+i}\|^2}_{\leq 2\eta_\tau^2 L(F_k(w^k_{\tau E+i}) - F_k^*)} \\
& -2\eta_\tau\big((F_k(w^k_{\tau E+i}) - F_k(w^*)) + \frac{\mu}{2}\|w^k_{\tau E+i} - w^*\|^2\big)\Bigg) \\
& \leq \|\bar{w}_{\tau E+i} - w^*\|^2 - \mu\eta_\tau\sum_{k=1}^{N} p^k_\tau \alpha^k_{\tau E+i}\|w^k_{\tau E+i} - w^*\|^2 \\
& + \sum_{k=1}^{N} p^k_\tau \alpha^k_{\tau E+i}\|\bar{w}_{\tau E+i} - w^k_{\tau E+i}\|^2 + C
\end{aligned} \tag{5.37}$$

where

$$\begin{aligned}C \overset{\Delta}{=} & 2(1+\theta)L\eta_\tau^2 \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k (F_k(w_{\tau E+i}^k) - F_k^*) \\ & -2\eta_\tau \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k (F_k(w_{\tau E+i}^k) - F_k(w^*))\end{aligned} \tag{5.38}$$

$$\begin{aligned} & \|w_{\tau E+i}^k - w^*\|^2 = \|w_{\tau E+i}^k - \bar{w}_{\tau E+i} + \bar{w}_{\tau E+i} - w^*\|^2 \\ = & \|w_{\tau E+i}^k - \bar{w}_{\tau E+i}\|^2 + \|\bar{w}_{\tau E+i} - w^*\|^2 + 2\langle w_{\tau E+i}^k - \bar{w}_{\tau E+i}, \bar{w}_{\tau E+i} - w^*\rangle \\ \geq & \|w_{\tau E+i}^k - \bar{w}_{\tau E+i}\|^2 + \|\bar{w}_{\tau E+i} - w^*\|^2 - 2\|w_{\tau E+i}^k - \bar{w}_{\tau E+i}\|\|\bar{w}_{\tau E+i} - w^*\| \\ \geq & \|w_{\tau E+i}^k - \bar{w}_{\tau E+i}\|^2 + \|\bar{w}_{\tau E+i} - w^*\|^2 \\ & -(2\|w_{\tau E+i}^k - \bar{w}_{\tau E+i}\|^2 + \frac{1}{2}\|\bar{w}_{\tau E+i} - w^*\|^2) \\ = & \frac{1}{2}\|\bar{w}_{\tau E+i} - w^*\|^2 - \|w_{\tau E+i}^k - \bar{w}_{\tau E+i}\|^2 \end{aligned} \tag{5.39}$$

Thus,

$$\begin{aligned} A_1 \leq & (1 - \frac{1}{2}\mu\eta_\tau \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k)\|\bar{w}_{\tau E+i} - w^*\|^2 \\ & +(1+\mu\eta_\tau)\sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k \|\bar{w}_{\tau E+i} - w_{\tau E+i}^k\|^2 + C \end{aligned} \tag{5.40}$$

Let $\gamma_\tau = 2\eta_\tau(1-(1+\theta)L\eta_\tau)$. Assume $\eta_\tau \leq \frac{1}{2(1+\theta)L}$, hence $\eta_\tau \leq \gamma_\tau \leq 2\eta_\tau$.

$$\begin{aligned}
C &= -2\eta_\tau(1-(1+\theta)L\eta_\tau)\sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k (F_k(w_{\tau E+i}^k) - F_k^*) \\
&\qquad + 2\eta_\tau \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k (F_k(w^*) - F_k^*) \\
&= -\gamma_\tau \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k (F_k(w_{\tau E+i}^k) - F_k^* + F_k(w^*) - F_k(w^*)) \\
&\qquad + 2\eta_\tau \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k (F_k(w^*) - F_k^*) \\
&= -\gamma_\tau \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k (F_k(w_{\tau E+i}^k) - F_k(w^*)) \\
&\qquad + (2\eta_\tau - \gamma_\tau) \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k (F_k(w^*) - F_k^*) \\
&\leq \underbrace{-\gamma_\tau \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k (F_k(w_{\tau E+i}^k) - F_k(w^*))}_{D} + 2(1+\theta)L\eta_\tau^2 \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k \Gamma_k
\end{aligned} \tag{5.41}$$

Next we bound D

$$
\begin{aligned}
&\sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k (F_k(w_{\tau E+i}^k) - F_k(w^*)) \\
&= \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k (F_k(w_{\tau E+i}^k) - F_k(\bar{w}_{\tau E+i})) \\
&\qquad + \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k (F_k(\bar{w}_{\tau E+i}) - F_k(w^*)) \\
&\geq \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k \langle \nabla F_k(\bar{w}_{\tau E+i}), w_{\tau E+i}^k - \bar{w}_{\tau E+i} \rangle \\
&\qquad + \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k (F_k(\bar{w}_{\tau E+i}) - F_k(w^*)) \\
&\geq - \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k \|\nabla F_k(\bar{w}_{\tau E+i})\| \|w_{\tau E+i}^k - \bar{w}_{\tau E+i}\| \qquad (5.42) \\
&\qquad + \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k (F_k(\bar{w}_{\tau E+i}) - F_k(w^*)) \\
&\geq - \frac{1}{2} \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k (\eta_\tau \underbrace{\|\nabla F_k(\bar{w}_{\tau E+i})\|^2}_{\leq 2L(F_k(\bar{w}_{\tau E+i}) - F_k^*)} + \frac{1}{\eta_\tau} \|w_{\tau E+i}^k - \bar{w}_{\tau E+i}\|^2) \\
&\qquad + \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k (F_k(\bar{w}_{\tau E+i}) - F_k(w^*)) \\
&\geq - \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k \Big(\eta_\tau L (F_k(\bar{w}_{\tau E+i}) - F_k^*) + \frac{1}{2\eta_\tau} \|w_{\tau E+i}^k - \bar{w}_{\tau E+i}\|^2 \Big) \\
&\qquad + \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k (F_k(\bar{w}_{\tau E+i}) - F_k(w^*))
\end{aligned}
$$

Thus,

$$\begin{aligned}
C &\leq \gamma_\tau \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k (\eta_\tau L \underbrace{(F_k(\bar{w}_{\tau E+i}) - F_k^*)}_{F_k(\bar{w}_{\tau E+i}) - F_k(w^*) + F_k(w^*) - F_k^*} + \frac{1}{2\eta_\tau} \| w_{\tau E+i}^k - \bar{w}_{\tau E+i} \|^2) \\
&\quad - \gamma_\tau \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k (F_k(\bar{w}_{\tau E+i}) - F_k(w^*)) + 2(1+\theta) L \eta_\tau^2 \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i} \Gamma_k \\
&= \gamma_\tau (\eta_\tau L - 1) \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k (F_k(\bar{w}_{\tau E+i}) - F_k(w^*)) \\
&\quad + \underbrace{\frac{\gamma_\tau}{2\eta_\tau}}_{\leq 1} \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k \| w_{\tau E+i}^k - \bar{w}_{\tau E+i} \|^2 \\
&\quad + 2(1+\theta) L \eta_\tau^2 \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i} \Gamma_k + \underbrace{\gamma_\tau}_{\leq 2\eta_\tau} \eta_\tau L \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k \Gamma_k \\
&\leq \gamma_\tau (\eta_\tau L - 1) \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k (F_k(\bar{w}_{\tau E+i}) - F_k(w^*)) \\
&\quad + \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k \| w_{\tau E+i}^k - \bar{w}_{\tau E+i} \|^2 + 2(2+\theta) L \eta_\tau^2 \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k \Gamma_k
\end{aligned} \tag{5.43}$$

Plug (5.43) to (5.40) we have

$$\begin{aligned}
A_1 &\leq \| \bar{w}_{\tau E+i} - w^* \|^2 - \mu \eta_\tau \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k \| w_{\tau E+i}^k - w^* \|^2 \\
&\quad + 2 \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k \| \bar{w}_{\tau E+i} - w_{\tau E+i}^k \|^2 \\
&\quad + 2(2+\theta) L \eta_\tau^2 \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k \Gamma_k \\
&\quad + \gamma_\tau (\eta_\tau L - 1) \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k (F_k(\bar{w}_{\tau E+i}) - F_k(w^*))
\end{aligned} \tag{5.44}$$

Plug (5.44) to (5.29),

$$\begin{aligned}\|\bar{w}_{\tau E+i+1}-w^*\|^2 \leq &(1-\frac{1}{2}\mu\eta_\tau\sum_{k=1}^{N}p_\tau^k\alpha_{\tau E+i}^k)\|\bar{w}_{\tau E+i}-w^*\|^2\\ &+\eta_\tau^2\|\sum_{k=1}^{N}p_\tau^k\alpha_{\tau E+i}^k(\bar{g}_{\tau E+i}^k-g_{\tau E+i}^k)\|^2\\ &+\underbrace{(2+\mu\eta_\tau)}_{\leq 2+\frac{\mu}{2(1+\theta)L}}\sum_{k=1}^{N}p_\tau^k\alpha_{\tau E+i}^k\|\bar{w}_{\tau E+i}-w_{\tau E+i}^k\|^2\\ &+2(2+\theta)L\eta_\tau^2\sum_{k=1}^{N}p_\tau^k\alpha_{\tau E+i}^k\Gamma_k\\ &+\underbrace{\gamma_\tau(1-\eta_\tau L)}_{\leq 2\eta_\tau}\sum_{k=1}^{N}p_\tau^k\alpha_{\tau E+i}^k(F_k(w^*)-F_k(\bar{w}_{\tau E+i}))\end{aligned} \tag{5.45}$$

Define

$$\begin{aligned}C_{\tau E+i}=&(2+\frac{\mu}{2(1+\theta)L})\sum_{k=1}^{N}p_\tau^k\alpha_{\tau E+i}^k\|\bar{w}_{\tau E+i}-w_{\tau E+i}^k\|^2\\ &+\|\sum_{k=1}^{N}p_\tau^k\alpha_{\tau E+i}^k(\bar{g}_{\tau E+i}^k-g_{\tau E+i}^k)\|^2\\ &+2(2+\theta)L\sum_{k=1}^{N}p_\tau^k\alpha_{\tau E+i}^k\Gamma_k\end{aligned} \tag{5.46}$$

Thus,

$$\begin{aligned}&\|\bar{w}_{\tau E+i+1}-w^*\|^2 \leq (1-\frac{1}{2}\mu\eta_\tau\sum_{k=1}^{N}p_\tau^k\alpha_{\tau E+i}^k)\|\bar{w}_{\tau E+i}-w^*\|^2+\eta_\tau^2 B_{\tau E+i}\\ &+2\eta_\tau\sum_{k=1}^{N}p_\tau^k\alpha_{\tau E+i}^k(F_k(w^*)-F_k(\bar{w}_{\tau E+i}))\end{aligned} \tag{5.47}$$

Apply the lemmas we have

$$\begin{aligned}\mathbb{E}_{\xi}[C_{\tau E+i}] \leq& \sum_{k=1}^{N}(p_{\tau}^{k})^{2}\alpha_{\tau E+i}^{k}\sigma_{k}^{2} + 2(2+\theta)L\sum_{k=1}^{N}p_{\tau}^{k}\alpha_{\tau E+i}^{k}\Gamma_{k} \\ &+ (2+\frac{\mu}{2(1+\theta)L})(E-1)G^{2}\Big(\sum_{k=1}^{N}p_{\tau}^{k}s_{\tau}^{k} + (\sum_{k=1}^{N}p_{\tau}^{k}-2)_{+}\sum_{k=1}^{N}\frac{(p_{\tau}^{k})^{2}}{p^{k}}s_{\tau}^{k}\Big)\end{aligned} \tag{5.48}$$

For convenience we write $\Delta_{\tau E+i} = \|\bar{w}_{\tau E+i} - w^{*}\|^{2}$, and $\bar{\Delta}_{\tau E+i} = \mathbb{E}[\Delta_{\tau E+i}]$, where the expectation is taken over all random variables up to $\tau E + i$.

Bounding $\|\bar{w}_{\tau E} - w^{*}\|$

Summing from τE to $(\tau+1)E$ we have

$$\begin{aligned}\sum_{i=1}^{E}\Delta_{\tau E+i} \leq& \sum_{i=0}^{E-1}(1-\frac{1}{2}\mu\eta_{\tau}\sum_{k=1}^{N}p_{\tau}^{k}\alpha_{\tau E+i}^{k})\Delta_{\tau E+i} \\ &+\eta_{\tau}^{2}C_{\tau} + 2\eta_{\tau}\sum_{k=1}^{N}p_{\tau}^{k}s_{\tau}^{k}(F_{k}(w^{*}) - F_{k}(\bar{w}_{\tau E+l}))\end{aligned} \tag{5.49}$$

where $C_{\tau} = \sum_{i=0}^{E-1}C_{\tau E+i}$, and $\bar{w}_{\tau E+l} = \operatorname{argmin}_{\bar{w}_{\tau E+i}}\sum_{k=1}^{N}p_{\tau}^{k}\alpha_{\tau E+i}^{k}F_{k}(\bar{w}_{\tau E+i})$. Reorganize it we can get

$$\begin{aligned}\Delta_{(\tau+1)E} \leq& \Delta_{\tau E} - \frac{1}{2}\mu\eta_{\tau}\sum_{i=0}^{E-1}\sum_{k=1}^{N}p_{\tau}^{k}\alpha_{\tau E+i}^{k}\Delta_{\tau E+i} \\ &+ \eta_{\tau}^{2}C_{\tau} + 2\eta_{\tau}\sum_{k=1}^{N}p_{\tau}^{k}s_{\tau}^{k}(F_{k}(w^{*}) - F_{k}(\bar{w}_{\tau E+l}))\end{aligned} \tag{5.50}$$

We then seek to find a lower bound for $\Delta_{\tau E+i}$.

$$\begin{aligned}\sqrt{\Delta_{\tau E+i+1}} &= \|\bar{w}_{\tau E+i+1} - w^{*}\| = \|\bar{w}_{\tau E+i+1} - \bar{w}_{\tau E+i} + \bar{w}_{\tau E+i} - w^{*}\| \\ &\leq \|\bar{w}_{\tau E+i+1} - \bar{w}_{\tau E+i}\| + \sqrt{\Delta_{\tau E+i}} \\ &= \|\eta_{\tau}\sum_{k=1}^{N}p_{\tau}^{k}\alpha_{\tau E+i}^{k}g_{\tau E+i}^{k}\| + \sqrt{\Delta_{\tau E+i}}\end{aligned} \tag{5.51}$$

Define $h_{\tau E+i} = \| \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k g_{\tau E+i}^k \|$. Thus,

$$\begin{aligned}\sqrt{\Delta_{(\tau+1)E}} &\leq \sqrt{\Delta_{(\tau+1)E-1}} + \eta_\tau h_{(\tau+1)E-1} \\ &\leq \cdots \leq \sqrt{\Delta_{\tau E+i}} + \sum_{j=i}^{E-1} \eta_\tau h_{\tau E+j}\end{aligned} \tag{5.52}$$

$$\begin{aligned}\Delta_{(\tau+1)E} &\leq \Delta_{\tau E+i} + 2\sqrt{\Delta_{\tau E+i}}(\sum_{j=i}^{E-1} \eta_\tau h_{\tau E+j}) + (\sum_{j=i}^{E-1} \eta_\tau h_{\tau E+j})^2 \\ &\leq 2\Delta_{\tau E+i} + 2(\sum_{j=i}^{E-1} \eta_\tau h_{\tau E+j})^2\end{aligned} \tag{5.53}$$

$$\Delta_{\tau E+i} \geq \frac{1}{2}\Delta_{(\tau+1)E} - (\sum_{j=i}^{E-1} \eta_\tau h_{\tau E+j})^2 \geq \frac{1}{2}\Delta_{(\tau+1)E} - (\sum_{j=0}^{E-1} \eta_\tau h_{\tau E+j})^2 \tag{5.54}$$

Plug (5.54) to (5.50) we can get

$$\begin{aligned}(1 + \frac{1}{4}\mu\eta_\tau \sum_{k=1}^{N} p_\tau^k s_\tau^k)\Delta_{(\tau+1)E} &\leq \Delta_{\tau E} + \frac{1}{2}\mu\eta_\tau^3 \sum_{k=1}^{N} p_\tau^k s_\tau^k (\sum_{i=0}^{E-1} h_{\tau E+i})^2 + \eta_\tau^2 C_\tau \\ &\quad + 2\eta_\tau \sum_{k=1}^{N} p_\tau^k s_\tau^k (F_k(w^*) - F_k(\bar{w}_{\tau E+l}))\end{aligned} \tag{5.55}$$

Define $H_\tau = (\sum_{i=0}^{E-1} h_{\tau E+i})^2$. Apply Lemma 5.2, Lemma 5.3 and Assumption 5.4, we have

$$\begin{aligned}\mathbb{E}_\xi[h_{\tau E+i}^2] &= \mathbb{E}_\xi \| \sum_{k=1}^{N} p_\tau^k \alpha_{\tau E+i}^k g_{\tau E+i}^k \|^2 \\ &\leq \sum_{k=1}^{N} \frac{(p_\tau^k)^2}{p^k} \mathbb{E}_\xi \| \alpha_{\tau E+i}^k g_{\tau E+i}^k \|^2 \leq \sum_{k=1}^{N} \frac{(p_\tau^k)^2}{p^k} G^2 \alpha_{\tau E+i}^k\end{aligned} \tag{5.56}$$

$$\mathbb{E}_\xi[H_\tau] = \mathbb{E}_\xi[(\sum_{i=0}^{E-1} h_{\tau E+i})^2] \leq \mathbb{E}_\xi[E \sum_{i=0}^{E-1} h_{\tau E+i}^2] \leq EG^2 \sum_{k=1}^{N} \frac{(p_\tau^k)^2}{p^k} s_\tau^k \tag{5.57}$$

$$
\begin{aligned}
\mathbb{E}_\xi[C_\tau] = \sum_{i=0}^{E-1} \mathbb{E}_\xi[C_{\tau E+i}] = \sum_{k=1}^{N} (p_\tau^k)^2 s_\tau^k \sigma_k^2 + 2(2+\theta)L \sum_{k=1}^{N} p_\tau^k s_\tau^k \Gamma_k \\
+ (2 + \frac{\mu}{2(1+\theta)L}) E(E-1) G^2 \Big(\sum_{k=1}^{N} p_\tau^k s_\tau^k + \theta(\sum_{k=1}^{N} p_\tau^k - 2)_+ \sum_{k=1}^{N} p_\tau^k s_\tau^k \Big)
\end{aligned} \tag{5.58}
$$

Write $\bar{\Delta}_{\tau E+i} = \mathbb{E}_\xi[\Delta_{\tau E+i}]$,$\bar{C}_\tau = \mathbb{E}_\xi[C_\tau]$, $\bar{H}_\tau = \mathbb{E}_\xi[(\sum_{i=0}^{E-1} h_{\tau E+i})^2]$, then

$$
\begin{aligned}
(1 + \frac{1}{4}\mu\eta_\tau \sum_{k=1}^{N} p_\tau^k s_\tau^k) \bar{\Delta}_{(\tau+1)E} \le \bar{\Delta}_{\tau E} + \frac{1}{2}\mu\eta_\tau^3 \sum_{k=1}^{N} p_\tau^k s_\tau^k \bar{H}_\tau + \eta_\tau^2 \bar{C}_\tau \\
+ 2\eta_\tau \mathbb{E}_\xi \sum_{k=1}^{N} p_\tau^k s_\tau^k (F_k(w^*) - F_k(\bar{w}_{\tau E+l}))
\end{aligned} \tag{5.59}
$$

Let $z_\tau = 0$ indicate the event that for all k, $\mathbb{E}[p_\tau^k s_\tau^k] = c_\tau p^k$ for come constant c_τ that does not depend on k, otherwise $z_\tau^k = 1$. Note that if $z_\tau = 0$, then $\sum_{k=1}^{N} p_\tau^k s_\tau^k (F_k(w^*) - F_k(\bar{w}_{\tau E+l})) = c_\tau (F(w^*) - F(\bar{w}_{\tau E+l})) \le 0$. Otherwise, we have

$$
\begin{aligned}
\sum_{k=1}^{N} p_\tau^k s_\tau^k (F_k(w^*) - F_k(\bar{w}_{\tau E+l})) &= \sum_{k=1}^{N} p_\tau^k s_\tau^k (\underbrace{F_k(w^*) - F_k^*}_{\Gamma_k} + \underbrace{F_k^* - F_k(\bar{w}_{\tau E+l})}_{\le 0}) \\
&\le \sum_{k=1}^{N} p_\tau^k s_\tau^k \Gamma_k
\end{aligned} \tag{5.60}
$$

Put it together

$$
\sum_{k=1}^{N} p_\tau^k s_\tau^k (F_k(w^*) - F_k(\bar{w}_{\tau E+l})) \le z_\tau \sum_{k=1}^{N} p_\tau^k s_\tau^k \Gamma_k \tag{5.61}
$$

Assume $\eta_\tau \leq \frac{4}{\mu E\theta} \leq \frac{4}{\mu \sum_{k=1}^N p_\tau^k s_\tau^k}$, divide both sides with $1 + \frac{1}{4}\mu\eta_\tau \sum_{k=1}^N p_\tau^k s_\tau^k$ in (5.59) we can get

$$\begin{aligned}
\bar{\Delta}_{(\tau+1)E} &\leq \Big(1 - \frac{\frac{1}{4}\mu\eta_\tau \sum_{k=1}^N p_\tau^k s_\tau^k}{1 + \frac{1}{4}\mu\eta_\tau \sum_{k=1}^N p_\tau^k s_\tau^k}\Big)\bar{\Delta}_{\tau E} + 2\eta_\tau^2 \bar{H}_\tau + \eta_\tau^2 \bar{C}_\tau \\
&\quad + 2\eta_\tau z_\tau \sum_{k=1}^N p_\tau^k s_\tau^k \Gamma_k \\
&\leq \Big(1 - \frac{1}{8}\mu\eta_\tau \sum_{k=1}^N p_\tau^k s_\tau^k\Big)\bar{\Delta}_{\tau E} + \eta_\tau^2 B_\tau + 2\eta_\tau z_\tau \sum_{k=1}^N p_\tau^k s_\tau^k \Gamma_k
\end{aligned} \tag{5.62}$$

Note that p_τ^k, s_τ^k are independent with $\bar{\Delta}_{\tau E}$. Taking expectation over p_τ^k and s_τ^k we get

$$\mathbb{E}[\bar{\Delta}_{(\tau+1)E}] \leq \Big(1 - \frac{1}{8}\mu\eta_\tau \mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k]\Big)\bar{\Delta}_{\tau E} + \eta_\tau^2 \mathbb{E}[B_\tau] + 2\eta_\tau z_\tau \sum_{k=1}^N \mathbb{E}[p_\tau^k s_\tau^k]\Gamma_k \tag{5.63}$$

5.7.2 Proof of Theorem 5.1

When the distributions of s_τ^k do not change with time, we have $B_\tau = B$. We prove the convergence by induction. Let $\eta_\tau = \frac{8}{\mu \mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k]} \frac{2E}{\tau E + \gamma}$. Initially, $\frac{V_0}{\gamma^2} \geq \mathbb{E}[\bar{\Delta}_0]$. Suppose $\mathbb{E}[\bar{\Delta}_{\tau E}] \leq \frac{M_\tau D + V}{\tau E + \gamma}$, then

$$\begin{aligned}
\mathbb{E}[\bar{\Delta}_{(\tau+1)E}] &\leq \frac{\tau E + \gamma - 2E}{\tau E + \gamma} \frac{M_\tau D + V}{\tau E + \gamma} \\
&+ \left(\frac{16E}{\mu \mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k]}\right)^2 \frac{B}{(\tau E + \gamma)^2} + \frac{\frac{1}{2} z_\tau D}{\tau E + \gamma} \\
&\leq \frac{\tau E + \gamma - E}{(\tau E + \gamma)^2} (M_\tau D + V) + \frac{\frac{1}{2} z_\tau D}{\tau E + \gamma} \\
&+ \underbrace{\left(\frac{16E}{\mu \mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k]}\right)^2 \frac{B}{(\tau E + \gamma)^2} - \frac{E(M_\tau D + V)}{(\tau E + \gamma)^2}}_{\leq 0} \\
&\leq \frac{M_\tau D + V}{(\tau+1)E + \gamma} + \frac{\frac{1}{2}\frac{\tau E + \gamma + E}{\tau E + \gamma} z_\tau D}{(\tau+1)E + \gamma} \leq \frac{M_{\tau+1} D + V}{(\tau+1)E + \gamma}
\end{aligned} \tag{5.64}$$

Thus $\bar{\Delta}_{(\tau+1)E} \leq \frac{M_{\tau+1} D + V}{(\tau+1)E + \gamma}$.

We can check it satisfies the previous assumptions regarding η_τ:

$$\begin{aligned}\eta_\tau \leq \eta_0 =& \frac{16E/(\mu\mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k])}{E+\gamma} \\ \leq& \frac{16E/(\mu\mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k])}{32E(1+\theta)L/(\mu\mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k])} = \frac{1}{2(1+\theta)L}\end{aligned} \tag{5.65}$$

$$\eta_\tau \leq \eta_0 = \frac{16E/(\mu\mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k])}{E+\gamma} \leq \frac{16E/(\mu\mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k])}{4E^2\theta/(\mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k])} = \frac{4}{\mu E\theta} \tag{5.66}$$

Extension to Time-Varying Distributions

When the distribution of s_τ^k vary with time, we can still establish a convergence with slightly different definitions. Redefine $\gamma = \max\left\{\frac{32E(1+\theta)L}{\mu \min_\tau \mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k]}, \frac{4E^2\theta}{\min_\tau \mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k]}\right\}$, $V_\tau = \max\left\{\gamma^2\mathbb{E}\|w_0^{\mathcal{G}} - w^*\|^2, \left(\frac{16E}{\mu}\right)^2 \sum_{t=0}^{\tau-1} \frac{\mathbb{E}[B_t]}{\left(\mathbb{E}[\sum_{k=1}^N p_t^k s_t^k]\right)^2}\right\}$

We now prove by induction that with this definition, we can obtain

$$\mathbb{E}[\bar{\Delta}_{\tau E}] \leq \frac{M_\tau D}{\tau E+\gamma} + \frac{V_\tau}{(\tau E+\gamma)^2} \tag{5.67}$$

Let $\eta_\tau = \frac{8}{\mu\mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k]}\frac{2E}{(\tau+1)E+\gamma}$. Initially, $\frac{V_0}{\gamma^2} \geq \mathbb{E}[\bar{\Delta}_0]$. Suppose $\mathbb{E}[\bar{\Delta}_{\tau E}] \leq \frac{M_\tau D}{\tau E+\gamma} + \frac{V_\tau}{(\tau E+\gamma)^2}$, then

$$\begin{aligned}\mathbb{E}[\bar{\Delta}_{(\tau+1)E}] \leq& \frac{\tau E+\gamma-E}{(\tau+1)E+\gamma}\left(\frac{M_\tau D}{\tau E+\gamma} + \frac{V_\tau}{(\tau E+\gamma)^2}\right) \\ &+ \frac{(16E)^2\mathbb{E}[\bar{B}_\tau + 2\bar{H}_\tau]}{(\mu\mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k])^2\left((\tau+1)E+\gamma\right)^2} + \frac{z_\tau D}{(\tau+1)E+\gamma} \\ \leq& \frac{(\tau E+\gamma-E)M_\tau D}{(\tau E+\gamma)^2 - E^2} + \frac{\tau E+\gamma-E}{(\tau E+\gamma)^2-E^2}\frac{V_\tau}{(\tau+1)E+\gamma} \\ &+ \frac{(16E)^2\mathbb{E}[\bar{B}_\tau + 2\bar{H}_\tau]}{(\mu\mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k])^2\left((\tau+1)E+\gamma\right)^2} + \frac{z_\tau D}{(\tau+1)E+\gamma} \\ \leq& \frac{M_\tau D}{(\tau+1)E+\gamma} + \frac{V_\tau}{\left((\tau+1)E+\gamma\right)^2} \\ &+ \frac{(16E)^2\mathbb{E}[\bar{B}_\tau + 2\bar{H}_\tau]}{(\mu\mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k])^2\left((\tau+1)E+\gamma\right)^2} + \frac{z_\tau D}{(\tau+1)E+\gamma} \\ =& \frac{M_{\tau+1} D}{(\tau+1)E+\gamma} + \frac{V_{\tau+1}}{\left((\tau+1)E+\gamma\right)^2}\end{aligned} \tag{5.68}$$

Thus $\bar{\Delta}_{(\tau+1)E} \leq \frac{M_{\tau+1}D}{(\tau+1)E+\gamma} + \frac{V_{\tau+1}}{((\tau+1)E+\gamma)^2}$.

Easy to check previous assumptions regarding η_τ are all satisfied.

5.7.3 Proof of Theorem 5.2

- Departure Case: $\tilde{n} = n - n_l$

$$
\begin{aligned}
\|\tilde{w}^* - w^*\| \leq \frac{2}{\mu}\|\nabla F(\tilde{w}^*)\| = \frac{2}{\mu}\left\|\nabla F(\tilde{w}^*) - \underbrace{\nabla \tilde{F}(\tilde{w}^*)}_{=0}\right\| \\
=& \frac{2}{\mu}\left\|\sum_{k\neq l}(p^k - \tilde{p}^k)\nabla F_k(\tilde{w}^*) + p^l \nabla F_l(\tilde{w}^*)\right\| \\
=& \frac{2}{\mu}\left\|\sum_{k\neq l}\left(\frac{n_k}{n} - \frac{n_k}{n-n_l}\right)\nabla F_k(\tilde{w}^*) + p^l \nabla F_l(\tilde{w}^*)\right\| \\
=& \frac{2}{\mu}\left\|-\sum_{k\neq l}\left(\frac{n_l n_k}{n(n-n_l)}\right)\nabla F_k(\tilde{w}^*) + p^l \nabla F_l(\tilde{w}^*)\right\| \\
=& \frac{2}{\mu}\left\|-p^l \underbrace{\sum_{k\neq l}\tilde{p}^k \nabla F_k(\tilde{w}^*)}_{=\nabla \tilde{F}(\tilde{w}^*)=0} + p^l \nabla F_l(\tilde{w}^*)\right\| \\
=& \frac{2p^l}{\mu}\left\|\nabla F_l(\tilde{w}^*)\right\| \leq \frac{2p^l}{\mu}\sqrt{2L\left(F_l(\tilde{w}^*) - F_l^*\right)} = \frac{2\sqrt{2L}}{\mu}p^l\sqrt{\tilde{\Gamma}_l}
\end{aligned}
$$

- Arrival Case: $\tilde{n} = n + n_l$

$$
\begin{aligned}
\|\tilde{w}^* - w^*\| = \|w^* - \tilde{w}^*\| \leq \frac{2}{\mu}\|\nabla \tilde{F}(w^*)\| = \frac{2}{\mu}\left\|\nabla \tilde{F}(w^*) - \underbrace{\nabla F(w^*)}_{=0}\right\| \\
= \cdots =& \frac{2}{\mu}\left\|-\tilde{p}^l \underbrace{\sum_{k\neq l}p^k \nabla F_k(w^*)}_{=\nabla F(w^*)=0} + \tilde{p}^l \nabla F_l(w^*)\right\| \\
=& \frac{2\tilde{p}^l}{\mu}\left\|\nabla F_l(w^*)\right\| = \frac{2\sqrt{2L}}{\mu}\tilde{p}^l\sqrt{\Gamma_l}
\end{aligned}
$$

5.7.4 Proof of Corollary 5.2

Scheme A

In Scheme A, we only consider devices whose $s_\tau^k = E$. Let q_τ^k be an indicator denoting if client k is complete in round τ. Thus, $K_\tau = \sum_{k=1}^N q_\tau^k$.

Homogeneous participation. Obviously q_τ^k's are homogeneous when s_τ^k's are homogeneous. Thus, $\mathbb{E}[q_\tau^k] = q_\tau$, where $q_\tau = \mathbb{P}(s_\tau = E)$. We then have $\mathbb{P}(K_\tau = 0) = (1 - q_\tau)^N$. When choosing $p_\tau^k = \frac{Np^k}{K_\tau} q_\tau^k$, $\theta = N$. Note that by the definition of q_τ^k, we have $q_\tau^k s_\tau^k = E q_\tau^k$, so $\mathbb{E}[p_\tau^k s_\tau^k] = E\mathbb{E}[p_\tau^k]$. Similarly, we can replace all s_τ^k terms with E. Next we calculate $\mathbb{E}[p_\tau^k]$:

$$\begin{aligned}
&\mathbb{E}_q[p_\tau^k | K_\tau \neq 0] = Np^k \mathbb{E}_q\Big[\frac{q_\tau^k}{\sum_{i=1}^N q_\tau^i}\Big| K_\tau \neq 0]\\
=&Np^k \sum_{i=0}^{N-1} \frac{1}{1+i}\binom{N-1}{i}\frac{(q_\tau)^i(1-q_\tau)^{N-1-i}q_\tau}{1-(1-q_\tau)^N}\\
=&Np^k \sum_{i=1}^{N} \frac{1}{i}\binom{N-1}{i-1}\frac{(q_\tau)^i(1-q_\tau)^{N-i}}{1-(1-q_\tau)^N}\\
=&Np^k \sum_{i=1}^{N} \frac{1}{i}\frac{(N-1)!}{(i-1)!(N-i)!}\frac{(q_\tau)^i(1-q_\tau)^{N-i}}{1-(1-q_\tau)^N}\\
=&p^k \sum_{i=1}^{N} \binom{N}{i}\frac{(q_\tau)^i(1-q_\tau)^{N-i}}{1-(1-q_\tau)^N} = p^k\frac{1-\binom{N}{0}(q_\tau)^0(1-q_\tau)^N}{1-(1-q_\tau)^N} = p^k
\end{aligned} \tag{5.69}$$

Similarly,

$$\begin{aligned}
&\mathbb{E}_q[(p_\tau^k)^2 | K_\tau \neq 0] = (Np^k)^2 \mathbb{E}_q\Big[\frac{q_\tau^k}{(\sum_{i=1}^N q_\tau^i)^2}\Big| K_\tau \neq 0]\\
=&(Np^k)^2 \sum_{i=0}^{N-1} \frac{1}{(1+i)^2}\binom{N-1}{i}\frac{(q_\tau)^i(1-q_\tau)^{N-1-i}q_\tau}{1-(1-q_\tau)^N}\\
=&(Np^k)^2 \sum_{i=1}^{N} \frac{1}{i^2}\binom{N-1}{i-1}\frac{(q_\tau)^i(1-q_\tau)^{N-i}}{1-(1-q_\tau)^N}\\
=&N(p^k)^2 \sum_{i=1}^{N} \frac{1}{i}\binom{N}{i}\frac{(q_\tau)^i(1-q_\tau)^{N-i}}{1-(1-q_\tau)^N}\\
=&N(p^k)^2\mathbb{E}\Big[\frac{1}{K_\tau}|K_\tau \neq 0\Big]
\end{aligned} \tag{5.70}$$

It is possible that $\sum_{k=1}^N p_\tau^k > 2$, so we need to calculate $\mathbb{E}[p_\tau^k p_\tau^l | K_\tau \neq 0]$

$$\begin{aligned}
&\mathbb{E}_q[p_\tau^k p_\tau^l | K_\tau \neq 0] = N^2 p^k p^l \mathbb{E}\Big[\frac{q_\tau^k q_\tau^l}{(\sum_{i=1}^N q_\tau^i)^2} | K_\tau \neq 0\Big] \\
=& N^2 p^k p^l \sum_{i=0}^{N-2} \frac{1}{(2+i)^2} \binom{N-2}{i} \frac{(q_\tau)^i (1-q_\tau)^{N-2-i} (q_\tau)^2}{1-(1-q_\tau)^N} \\
=& \frac{N}{N-1} p^k p^l \sum_{i=2}^{N} \frac{i-1}{i} \binom{N}{i} \frac{(q_\tau)^i (1-q_\tau)^{N-i}}{1-(1-q_\tau)^N} \\
=& \frac{N}{N-1} p^k p^l \mathbb{E}\Big[1 - \frac{1}{K_\tau} | K_\tau \neq 0\Big]
\end{aligned} \tag{5.71}$$

For all k and τ, $\mathbb{E}[p_\tau^k s_\tau^k | K_\tau \neq 0] = Ep^k$, thus $z_\tau = 0, M_\tau = 0$ for all k, τ.

Therefore, $\mathbb{E}[B] = O(N^2 \mathbb{E}[\frac{1}{K_\tau} | K_\tau \neq 0] + \sum_{k=1}^N (p^k \sigma_k)^2 + \Gamma), \gamma = O(N)$, hence $V = O(N^2 \mathbb{E}[\frac{1}{K_\tau} | K_\tau \neq 0] + \sum_{k=1}^N (p^k \sigma_k)^2 + \Gamma)$. Plug them into Theorem 5.1, we can get an asymptotic rate of $O\left(\frac{\mathbb{E}[\frac{N^2}{K_\tau}] + \bar{\sigma}_N^2 + \Gamma}{\tau}\right)$.

Heterogeneous Participation. When s_τ^k's (i.e., q_τ^k's) are heterogeneous, generally $\mathbb{E}[p_\tau^k] \neq p^k$, furthermore, we may have $z_\tau = 1$ for all τ. To see this, consider an example where a device k_0 has $q_\tau^{k_0} = 1$, i.e. $\mathbb{P}(s_\tau^k = E) = 1$, whereas all the rest devices have $\mathbb{E}[q_\tau^k] = q_\tau$, then we can show that

$$\mathbb{E}_q[p_\tau^{k_0} | K_\tau \neq 0] = \mathbb{E}_q[p_\tau^{k_0}] = N p^{k_0} \mathbb{E}_q\Big[\frac{q_\tau^{k_0}}{(\sum_{i=1}^N q_\tau^i)^2}\Big] = p^{k_0} \frac{1-(1-q_\tau)^N}{q_\tau} \tag{5.72}$$

and for $k \neq k_0$

$$\begin{aligned}
&\mathbb{E}_q[p_\tau^k | K_\tau \neq 0] = \mathbb{E}_q[p_\tau^k] = N p^k \mathbb{E}_q\Big[\frac{q_\tau^k}{(\sum_{i=1}^N q_\tau^i)^2}\Big] \\
=& \frac{p^k}{(N-1) q_\tau} \sum_{i=2}^N (i-1) \binom{N}{i} (q_\tau)^i (1-q_\tau)^{N-i} \\
=& \frac{p^k}{(N-1) q_\tau} \Big(N q_\tau - N q_\tau (1-q_\tau)^{N-1} - (1 - (1-q_\tau)^N - N q_\tau (1-q_\tau)^{N-1})\Big) \\
=& p^k \frac{N q_\tau + (1-q_\tau)^N - 1}{(N-1) q_\tau}
\end{aligned} \tag{5.73}$$

Thus, different k will have different ratio of $\mathbb{E}[p_\tau^k s_\tau^k / p^k] = E\mathbb{E}[p_\tau^k / p^k]$, which indicates

$z_\tau = 1$. Since this is true for all τ, we have $M_\tau = \tau$. Thus according to Theorem 5.1, the learning will not converge to the global optimal, and the remainder loss is bounded by D/E.

Scheme B

In Scheme B, $p_\tau^k = p^k$ is a fixed number, so we only need to take expectation over s_τ^k, and $c_p = 1$. Since $\sum_{k=1}^N p^k = 1 < 2$, we can bound $\mathbb{E}[(\sum_{k=1}^N p_\tau^k - 2)_+ (\sum_{k=1}^N p_\tau^k s_\tau^k)] < 0$.

Homogeneous Participation. When s_τ^k's are homogeneous, i.e. $s_\tau^k \overset{iid}{\sim} s_\tau$, then $\mathbb{E}[p_\tau^k s_\tau^k]/p^k = \mathbb{E}[s_\tau]$. This is the same for all k, thus $z_\tau = 0$ and $M_\tau = 0$. Moreover, we have $\mathbb{E}[B] = O(\mathbb{E}[s_\tau](\bar{\sigma}_N^2 + \Gamma))$, $\gamma = O(1/\mathbb{E}[s_\tau])$, $V = O\left((\bar{\sigma}_N^2 + \Gamma)\frac{1}{\mathbb{E}[s_\tau]}\right)$, which yields an asymptotic convergence rate of $O\left(\frac{\bar{\sigma}_N^2 + \Gamma}{\tau \mathbb{E}[s_\tau]}\right)$.

Heterogeneous Participation. When s_τ^k's are heterogeneous, $\mathbb{E}[p_\tau^k s_\tau^k]/p^k = \mathbb{E}[s_\tau^k]$ varies with k. Thus, $z_\tau = 1$ and $M_\tau = \tau$. Therefore, the algorithm will not converge to the global optimum according to Theorem 5.1.

Scheme C

In Scheme C, $p_\tau^k = \frac{Ep^k}{s_\tau^k}$, so $\theta = E$. It is possible that $\sum_{k=1}^K p_\tau^k > 2$, so we need to calculate $\mathbb{E}\Big[(\sum_{k=1}^N p_\tau^k)(\sum_{k=1}^N p_\tau^k s_\tau^k)\Big]$.

Homogeneous Participation. When s_τ^k's are homogeneous, $\mathbb{E}[p_\tau^k s_\tau^k]/p^k = E$ for all k. Thus, $z_\tau = 0, M_\tau = 0$.

Moreover, we have

$$\mathbb{E}[\sum_{k=1}^N p_\tau^k] = E\mathbb{E}\Big[\frac{1}{s_\tau}\Big] \tag{5.74}$$

$$\mathbb{E}[\sum_{k=1}^N (p_\tau^k)^2] = \Big(E\mathbb{E}\Big[\frac{1}{s_\tau}\Big]\Big)^2 \sum_{k=1}^N (p^k)^2 \tag{5.75}$$

$$\mathbb{E}[\sum_{k=1}^N (p_\tau^k)^2 s_\tau^k] = E^2\mathbb{E}\Big[\frac{1}{s_\tau}\Big] \sum_{k=1}^N (p^k)^2 \tag{5.76}$$

$$\mathbb{E}\Big[(\sum_{k=1}^N p_\tau^k)(\sum_{k=1}^N p_\tau^k s_\tau^k)\Big] = E^2\mathbb{E}\Big[\frac{1}{s_\tau}\Big] \tag{5.77}$$

Therefore, we have $\mathbb{E}[B] = O\left(\mathbb{E}\left[\frac{1}{s_\tau}\right](\bar{\sigma}_N + \Gamma)\right) = V$, which yields a convergence rate of $O\left(\frac{\bar{\sigma}_N^2+\Gamma}{\tau(\mathbb{E}[1/s_\tau])^{-1}}\right)$.

Heterogeneous Participation. Even when s_τ^k's are heterogeneous, we still have $\mathbb{E}[p_\tau^k s_\tau^k]/p^k = E$ for active all k. Thus, $z_\tau = 0$ only if $I_\tau = 1$. Thus, $M_\tau = \sum_{t=0}^{\tau-1} I_t$. Moreover,

$$\mathbb{E}[\sum_{k=1}^N p_\tau^k] = E\sum_{k=1}^N p^k \mathbb{E}\Big[\frac{1}{s_\tau^k}\Big] \tag{5.78}$$

$$\mathbb{E}[\sum_{k=1}^N (p_\tau^k)^2] = E^2\sum_{k=1}^N \Big(p^k \mathbb{E}\Big[\frac{1}{s_\tau^k}\Big]\Big)^2 \tag{5.79}$$

$$\mathbb{E}[\sum_{k=1}^N (p_\tau^k)^2 s_\tau^k] = E^2\sum_{k=1}^N (p^k)^2 \mathbb{E}\Big[\frac{1}{s_\tau^k}\Big] \tag{5.80}$$

$$\mathbb{E}\Big[(\sum_{k=1}^N p_\tau^k)(\sum_{k=1}^N p_\tau^k s_\tau^k)\Big] = E^2\sum_{k=1}^N p^k \mathbb{E}\Big[\frac{1}{s_\tau^k}\Big] \tag{5.81}$$

Thus, $\mathbb{E}[B] = O\left(\sum_{k=1}^N (p^k\sigma_k)^2\mathbb{E}\left[\frac{1}{s_\tau^k}\right] + \Gamma\right) = V$, and the convergence rate is $O\left(\frac{\sum_{t=0}^{\tau-1} I_t D + \sum_{k}^{N}(p^k\sigma_k)^2\mathbb{E}\left[\frac{1}{s_\tau^k}\right] + \Gamma}{\tau}\right)$

5.7.5 Proof of Corollary 5.3

We first introduce the following lemma:

Lemma 5.4. *Suppose device l arrives, then for any w, we have*

$$F_l(w) = \frac{1}{\tilde{p}^l}\Big(\tilde{F}(w) - \frac{n}{\tilde{n}}F(w)\Big) \tag{5.82}$$

Proof. We expand the right hand side expression and show it equals $F_l(w)$:

$$\begin{aligned} \frac{1}{\tilde{p}^l}\Big(\tilde{F}(w) - \frac{n}{\tilde{n}}F(w)\Big) &= \frac{1}{\tilde{p}^l}\left(\sum_{k=1}^N \tilde{p}^k F_k(w) + \tilde{p}^l F_l(w) - \sum_{k=1}^N \frac{n}{\tilde{n}} p^k F_k(w)\right) \\ =&\frac{1}{\tilde{p}^l}\left(\sum_{k=1}^N \tilde{p}^k F_k(w) + \tilde{p}^l F_l(w) - \sum_{k=1}^N \tilde{p}^k F_k(w)\right) = F_l(w) \end{aligned} \tag{5.83}$$

□

Next we investigate the effect of applying additional update from l. Suppose the current global weight is $w^{\mathcal{G}}_{\tau E} = w$, and assume we perform full batch gradient for the additional update. After the update, it becomes

$$w' = w - \eta_\tau \delta^l \nabla F_l(w) \tag{5.84}$$

We are interested in the distance between w' and the new global optimum $\tilde{w}^*$:

$$\begin{aligned} \|w' - \tilde{w}^*\|^2 &= \|w - \eta_\tau \delta^l \nabla F_l(w) - \tilde{w}^*\|^2 \\ &= \|w - \tilde{w}^*\|^2 \underbrace{-2\eta_\tau \delta^l \langle w - \tilde{w}^*, \nabla F_l(w)\rangle + \left(\eta_\tau \delta^l\right)^2 \|\nabla F_l(w)\|^2}_{A(w,\delta^l)} \end{aligned} \tag{5.85}$$

Obviously, the additional update helps fast-reboot if $A(w, \delta^l) < 0$.

Applying Lemma 5.4 we can get

$$A(w, \delta^l) = -2\frac{\eta_\tau \delta^l}{\tilde{p}^l} \langle w - \tilde{w}^*, \nabla \tilde{F}(w) - \frac{n}{\tilde{n}} \nabla F(w)\rangle + \left(\eta_\tau \delta^l\right)^2 \|\nabla F_l(w)\|^2 \tag{5.86}$$

Write $b = w - w^*$, and use the mean value theorem we have

$$\begin{aligned} &-\langle w - \tilde{w}^*, \nabla F_l(w)\rangle = -\langle b + w^* - \tilde{w}^*, \nabla F_l(w^*) + \nabla^2 F_l(\xi) b\rangle \\ &\quad = -\langle w^* - \tilde{w}^*, \nabla F_l(w^*)\rangle - \langle w^* - \tilde{w}^*, \nabla^2 F_l(\xi) b\rangle - \langle b, \nabla F_l(w)\rangle \\ &\quad \le -\langle w^* - \tilde{w}^*, \nabla F_l(w^*)\rangle + \|w^* - \tilde{w}^*\| \|\nabla^2 F_l(\xi)\|_2 \|b\| + \|\nabla F_l(w)\| \|b\| \\ &\quad \le -\langle w^* - \tilde{w}^*, \nabla F_l(w^*)\rangle + (\|w^* - \tilde{w}^*\| + 1) W \|b\| \\ &\quad \le -\frac{1}{\tilde{p}^l} \langle w^* - \tilde{w}^*, \nabla \tilde{F}(w^*) - \frac{n}{\tilde{n}} \underbrace{\nabla F(w^*)\rangle}_{=0} + \left(\frac{2\sqrt{2L}}{\mu} \tilde{p}^l \sqrt{\Gamma_l} + 1\right) W \|b\| \\ &\quad \le -\frac{1}{\tilde{p}^l} \left(\tilde{F}(w^*) - \tilde{F}(\tilde{w}^*)\right) + \left(\frac{2\sqrt{2L}}{\mu} \tilde{p}^l \sqrt{\Gamma_l} + 1\right) W \|b\| \end{aligned} \tag{5.87}$$

Therefore,

$$\begin{aligned} A(w, \delta^l) \le & 2\frac{\eta_\tau \delta^l}{\tilde{p}^l} \left(\left(\frac{2\sqrt{2L}}{\mu} \tilde{p}^l \sqrt{\Gamma_l} + 1\right) W \|b\| - \left(\tilde{F}(w^*) - \tilde{F}(\tilde{w}^*)\right)\right) \\ & + (\eta_\tau \delta^l)^2 W^2 \end{aligned} \tag{5.88}$$

For $\delta^l > 0$, the right hand side can be negative if and only if $\|b\| < \frac{\tilde{F}(w^*)-\tilde{F}(\tilde{w}^*)}{\left(\frac{2\sqrt{2L}}{\mu}\tilde{p}^l\sqrt{\Gamma_l}+1\right)\tilde{p}^l W}$.

5.7.6 Proof of Corollary 5.4

The loss bound without objective shift is $f_0(\tau) = \frac{(\tau-\tau_0)D+V}{\tau E+\gamma}$, and the bound with shift is $f_1(\tau) = \frac{\frac{V}{\tau_0 E+\gamma}+\Gamma_l}{(\tau-\tau_0)E+\gamma}$. Note that $f_0(\tau)$ is a monotonic function. When it is increasing, we just need $f_0(\tau_0) = f_1(\tau)$, which yields

$$\tau - \tau_0 = \frac{1-\gamma}{E} + \frac{\Gamma_l(\tau_0 E+\gamma)}{EV} = O\left(\frac{\Gamma_l \tau_0}{V}\right) \tag{5.89}$$

Now we consider monotonically decreasing $f_0(\tau)$, which is more commonly observed in experiments. Let $C_1 = DE, C_2 = \gamma D + VE - E\Gamma_l$, $C_3 = V(\gamma-1)$, the only possible root for the quadratic equation $f_0(\tau) = f_1(\tau)$ is

$$\begin{aligned}\tau - \tau_0 &= \frac{EV}{\tau_0 E+\gamma} - C_2 \\ &\quad + \sqrt{4C_1\Gamma_l(\tau_0 E+\gamma) + \left(\frac{EV}{\tau_0 E+\gamma}\right)^2 - \frac{2C_2 EV}{\tau_0 E+\gamma} + (C_2^2 - 4C_1C_3)} \\ &= O(\sqrt{\tau_0\Gamma_l})\end{aligned} \tag{5.90}$$

Chapter 6

Soft Clustered Federated Learning

6.1 Introduction

Federated learning (FL) is known to suffer from the notorious non-IID (non- independently and identically distributed) data issue, in the sense that the model training is significantly slackened when clients' local data distributions are heterogeneous. In response to this challenge, some recent works propose to bypass data heterogeneity by performing *local model personalization*. Instead of pursuing one universally applicable model shared by all clients, these algorithms' training objective is to create one model for each client that fits its local data. Personalization methods include local fine tuning [88], model interpolation [58], and multi-task learning [89]. In this chapter, we focus on an alternative approach: *clustered federated learning*, which we generalize to train both cluster and personalized models on realistic distributions of client data.

Clustered FL relaxes the assumption of FL that each client has an unique data distribution; instead, it allows different clients to share one data distribution, with fewer source data distributions than clients. The objective of clustered FL is to train one model for every distribution. In traditional clustered FL, a client can only be associated with one data distribution. We thus call this method *hard clustered federated learning*. Under the hard association assumption, the non-IID problem can be easily resolved: simply group clients with the same data distribution into one cluster, then conduct conventional FL on each cluster, within which the data distribution is now IID among clients. Unlike other personalization methods, clustered FL thus produces centrally available models that can be selectively migrated to new users that are unwilling, or unable, to engage in the subsequent local adaptation process (e.g. fine tuning) due to privacy concerns or resource limitations. This convenience in model adoption is particularly valuable for the current training-testing-deployment lifecycle of FL where deployment, rather than the training itself, is the end goal [45].

However, hard clustered FL faces two fundamental problems in practice. First, *multiple clients may be unlikely to possess identical data distributions.* In fact, the real-world user data is more likely to follow a *mixture* of multiple distributions [59]. E.g., if each client is a mobile phone and we wish to model its user's content preferences, we might expect the clients to be clustered into adults and children. However, adult users may occasionally view children's content, and devices owned by teenagers (or shared by parents and children) may possess large fractions of data from both distributions. Similarly, content can be naturally grouped by users' interests (e.g., genres of movies), each of which may have a distinct distribution. Data from users with multiple interests then reflects a mixture of these distributions. Since the mixture ratios can vary for different clients, they may have different overall distributions even though the source distributions are identical. Clustering algorithms like the Gaussian mixture model [75] use a similar rationale. Clients may then require models personalized to their distributions to make accurate predictions on their data, in addition to the cluster models used for new users.

Hard clustered FL's second challenge is that *it cannot effectively exploit similarities between different clusters.* Though FL clients may have non-IID data distributions, two different distributions may still exhibit some similarity, as commonly assumed in personalization works [89]. For example, young people may have more online slang terms in their chatting data, but all users (generally) follow the same basic grammar rules. Thus, the knowledge distilled through the training on one distribution could be transferred to accelerate the training of others. However, in most hard clustered FL algorithms, different cluster models are trained independently, making it difficult to leverage the potential structural likeness among distributions. Note that unlike in other personalization methods where the discussion of similarity is restricted to similarities between individual clients, here we focus on the broader similarities between source cluster distributions. Thus, we can gain better insight into the general data relationship rather than just the relationships between participating clients.

To overcome clustered FL's first challenge of the hard association assumption, in this chapter, we utilize *soft clustered federated learning.* In soft clustered FL, we suppose that the data of each client follows a mixture of multiple distributions. However, training cluster models using clients with mixed data raises two new challenges. First, *the workload of clients can explode.* When all the data of a client comes from the same distribution, as in hard clustered FL, it ideally only needs to contribute towards one training task: training that distribution's model. However, in soft clustered FL, a client has multiple data sources. A natural extension of hard clustered FL is for the client to help train all cluster models whose distributions are included in its mixture [59]. However, the workload of participating clients then grows linearly with the number of clusters, which can be large (though typically much smaller than the number of clients) for some applications. This multiplying of client workload can make soft clustered FL infeasible, considering the resource restrictions on typical FL user devices and the long convergence time for many FL models [61]. Second, *the training of cluster models and the local personalization are distinct.* In hard clustered FL, client models are the same as cluster models since a client is uniquely bound to one cluster. In soft clustered FL, local distributions differ from individual cluster distributions, and thus training cluster models does not directly help the local personalization. Complicating things further, these local distributions and their exact relationships to the cluster models are unknown a priori. Combining the training of cluster and personalized models is then challenging.

To solve these two challenges, and handle the second disadvantage of hard clustered FL discussed above, we utilize the proximal local updating trick, which is originally developed in **FedProx** [52] to grant clients the use of different local solvers in FL. During the course of proximal local updating, instead of working on fitting the local model to the local dataset, each client optimizes a proximal local objective function that both carries local information and encodes knowledge from all cluster models. We name this proposed algorithm **FedSoft**.

In **FedSoft**, since the fingerprints of all clusters are integrated into one optimization objective, clients only need to solve *one single optimization problem*, for which the workload is almost the same as in conventional FL. In addition, by combining local data with cluster models in the local objective, clients can *perform local personalization on the fly.* Eventually, the server obtains collaboratively trained cluster models that can be readily applied to new users, and each participating client gets one personalized model as a byproduct. Proximal local updating allows a cluster to *utilize the knowledge of similar distributions*, overcoming the second disadvantage of the hard clustered FL. Intuitively, with all clusters present in the proximal objective, a client can take as reference training targets any cluster models whose distributions take up non-trivial fractions of its data. These component distributions, co-existing in the same dataset, are similar by nature. Thus, a personalized local model integrating all its component distributions can in turn be utilized by the component clusters to exploit their similarities.

Our *contributions* are: We design the **FedSoft** algorithm for efficient soft clustered FL. We establish a convergence guarantee that relates the algorithm's performance to the divergence of different distributions, and validate the effectiveness of the learned cluster and personalized models in experiments under various mixture patterns. Our results show the training of cluster models converges linearly to a remaining error determined by the cluster heterogeneity, and that **FedSoft** can outperform existing FL implementations in both global cluster models for future users and personalized local models for participating clients.

6.2 Related Works

The training objective of hard clustered FL is to simultaneously identify the cluster partitions and train a model for each cluster. Existing works generally adopt an Expectation-Maximization (EM) like algorithm, which iteratively alternates between the cluster identification and model training. Based on how the partition structure is discovered, these algorithms can be classified into four types:

The first type leverages the distance between model parameters, e.g., [101] proposes to determine client association based on the distances between client models and server models. Similarly, [10] suggests to apply a distance-based hierarchical clustering algorithm directly on client models. The second type determines the partition structure based on the gradient information, e.g., the **CFL** [82] algorithm splits clients into bi-partitions based on the cosine similarity of the client gradients, and then checks whether a partition is congruent (i.e., contains IID data) by examining the norm of gradients on its clients. Likewise, the **FedGroup** [30] algorithm quantifies the similarity among client gradients with the so-called Euclidean distance of decomposed cosine similarity metric, which decomposes the gradient into multiple directions using singular value decomposition. The third type utilizes the training loss, e.g., in **HyperCluster** [58], each client is greedily assigned to the cluster whose model yields the lowest loss on its local data. A generalization guarantee is provided for this algorithm. [34] proposes a similar algorithm named **IFCA**, for which a convergence bound is established under the assumption of good initialization and all clients having the same amount of data. The fourth type uses exogenous information about the data, e.g., [42] and [70] group patients into clusters respectively based on their electronic medical records and imaging modality. This information usually entails direct access to the user data and thus cannot be applied in the general case.

Recently, [59] proposes a multi-task learning framework similar to soft clustered FL that allows client data to follow a mixture of distributions. Their proposed **FedEM** algorithm adopts an EM algorithm and estimates the mixture coefficients based on the training loss. However, **FedEM** requires every client to perform a local update for each cluster in each round, which entails significantly more training time than conventional **FedAvg**. Their analysis moreover assumes a special form of the loss function with all distributions having the same marginal distribution, which is unrealistic. In contrast, **FedSoft** requires only a subset of clients to return gradients for only one optimization task in each round. Moreover, we show its convergence for generic data distributions and loss functions.

The proximal local updating procedure that we adopt incorporates a regularization term in the local objective, which is also used for model personalization outside clustered settings. Typical algorithms include **FedAMP** [43], which adds an attention-inducing function to the local objective, and **pFedMe** [29], which formulates the regularization as Moreau envelopes.

6.3 Formulation and Algorithm

Mixture of distributions. Assume that each data point at each client is drawn from *one of* the S distinct data distributions $\mathcal{P}_1, \cdots, \mathcal{P}_S$. Similar to general clustering problems, we take S as a hyperparameter determined a priori. Data points from all clients that follow the same distribution form a *cluster*. In soft clustered FL, a client may possess data from multiple clusters. Given a loss function $l(w; x, y)$, the (real) *cluster risk* $F_s(w)$ is the expected loss for data following $\mathcal{P}_s$:

$$F_s(w) \triangleq \mathbb{E}_{(x,y)\sim\mathcal{P}_s}[l(w;x,y)] \tag{6.1}$$

We then wish to find S cluster models $c_1^* \cdots c_S^*$ such that all cluster objectives are minimized simultaneously. These cluster models will be co-trained by all clients through coordination at the central server:

$$c_s^* = \text{argmin}_w F_s(w), s = 1, \cdots, S \tag{6.2}$$

Suppose a client $k \in [N]$ with local dataset $\mathcal{D}_k$ has $|\mathcal{D}_k| = n_k$ data points, among which n_{ks} data points are sampled from distribution $\mathcal{P}_s$. The real risk of a client can thus be written as an average of the cluster risks:

$$\begin{aligned} f_k(w_k) &\triangleq \frac{1}{n_k}\mathbb{E}\left[\sum_{s=1}^{S} \sum_{(x_k^i, y_k^i)\sim\mathcal{P}_s} l(w_k; x_k^i, y_k^i)\right] \\ &= \frac{1}{n_k}\sum\nolimits_{s=1}^{S} n_{ks}F_s(w_k) = \sum\nolimits_{s=1}^{S} u_{ks}F_s(w_k) \end{aligned} \tag{6.3}$$

Here we define $u_{ks} \triangleq n_{ks}/n_k \in [0,1]$ as the *importance weight* of cluster s to client k. In general, u_{ks}'s are unknown in advance and the learning algorithm attempts to estimate their values during the learning iterations. It is worth noting that while we directly work on real risks, our formulation and analysis can be easily extended to empirical risks by introducing local-global divergences as in [52].

Proximal local updating. Since f_k is a mixture of cluster risks, minimizing (6.3) alone does not help solve (6.2). Thus, we propose each client instead optimize a proximal form of (6.3):

$$h_k(w_k; c^t, u^t) \triangleq f_k(w_k) + \frac{\lambda}{2}\sum\nolimits_{s=1}^{S} u_{ks}^t \|w_k - c_s^t\|^2 \tag{6.4}$$

Here λ is a hyperparameter and u_{ks}^t denotes the estimation of u_{ks} at time t. In the local updating step, every client searches for the optimal local model w_k^* that minimizes h_k given the current global estimation of cluster models $\{c_s^t\}$. As in [52], clients may use any local solver to optimize h_k. This design of the proximal objective entails cluster models $\{c_s^t\}$ be shared among all clients through the server, as is usual in clustered FL [34]. We thus alternatively call $\{c_s^t\}$ the *centers*.

The regularization term $\frac{\lambda}{2}\sum_{s=1}^{S} u_{ks}^t \|w_k - c_s^t\|^2$ in the proximal objective serves as a reference point for the local model training. It allows clients to work on their own specific dataset while taking advantage of and being guided by the globally shared knowledge of the centers. The regularization is weighted by the importance weights u_{ks}, so that a client will pay more attention to distributions that have higher shares in its data. To see why compounded regularization helps identify individual centers, assume we have a perfect guess of $u_{ks}^t \equiv u_{ks}$. The minimization of (6.4) can then be decoupled as a series of sub- optimization problems $h_k(w_k; c^t) = \sum_{s=1}^{S} u_{ks}\left(F_s(w_k) + \frac{\lambda}{2}\|w_k - c_s^t\|^2\right)$. Thus, after h_k is minimized, the sub-problems corresponding to large u_{ks} will also be approximately solved. We can hence utilize the output local model w_k^{t*} to update these centers with large u_{ks}. Moreover, w_k^{t*} trained in this manner forges all its component distributions $\{\mathcal{D}_s | u_{ks}^t \neq 0\}$, which may share some common knowledge. Thus, the training of these clusters are bonded through the training of their common clients, exploiting similarities between the clusters.

The output model w_k^{t*} is itself a well personalized model that leverages both local client knowledge and the global cluster information. [59] shows that under certain conditions, the optimal client model for soft clustered FL is a mixture of the optimal cluster models. The same implication can also be captured by our proximal updating formulation. When $\sum_s u_{ks}^t = 1$, the gradient ∇h_k is

$$\nabla_{w_k} h_k = \nabla f_k(w_k) + \lambda\left(w_k - \sum\nolimits_{s=1}^{S} u_{ks}^t c_s^t\right) \tag{6.5}$$

which implies that w_k^* should be centered on $\sum_s u_{ks} c_s^*$. As a result, through the optimization of all h_k, not only will the server obtain the trained cluster models, but also each client will obtain a sufficiently personalized local model.

Algorithm design. We formally present **FedSoft** in Algorithm 6.1. The first step of the algorithm is to estimate the importance weights $\{u_{ks}^t\}$ for each client k (lines 3-14). The algorithm obtains them by finding the center that yields the smallest loss value for every data point belonging to that client, and counting the number of points $\{n_{ks}^t\}$ matched to every cluster s. If a client k has no samples matched to s ($n_{ks}^t = 0$), the algorithm sets $u_{ks}^t = \sigma$, where $0 < \sigma \ll 1$ is a pre-defined smoothing parameter.

Algorithm 6.1 FedSoft

```
1: Input: Global epoch $T$, importance weights estimation interval $\tau$, number of clients $N$, client selection size $K$, counter smoother $\sigma$
2: for $t = 0, \cdots, T-1$ do
3:     if $t \ \ mod \ \ \tau = 0$ then
4:         Server sends centers $\{c_s^t\}$ to all clients
5:         for each client $k$ do
6:             for each data point $(x_k^i, y_k^i)$ do
7:                 $j = \text{argmin}_s l(c_s^t; x_k^i, y_k^i)$
8:                 $n_{kj}^t = n_{kj}^t + 1$
9:             Send $u_{ks}^t = \max\{\frac{n_{ks}^t}{n_k}, \sigma\}$ to server
10:    else
11:        Set $u_{ks}^t = u_{ks}^{t-1}$
12:    Server computes $v_{sk}^t$ as in (6.6)
13:    Server selects $S$ sets of clients $Sel_s^t \subset [N]$ at random for each cluster, where $|Sel_s^t| = K$, and each client gets selected with probability $v_{sk}^t$
14:    Selected clients download $\{c_s^t\}$, then compute and report $w_k^{t+1} = \text{argmin}_{w_k} h_k(w_k; c^t, u^t)$
15:    Server aggregates $c_s^{t+1} = \frac{1}{K}\sum_{k \in Sel_s^t} w_k^{t+1}$
```

Once the server receives the importance weights $\{u_{ks}^t\}$, it computes the *aggregation weights* v_{sk}^t as follows (line 15):

$$v_{sk}^t = \frac{u_{ks}^t n_k}{\sum_{k' \in Sel_s^t} u_{k's}^t n_{k'}} \tag{6.6}$$

i.e., a client that has higher importance weight on cluster s will be given higher aggregation weight, and vice versa. The introduction of the smoother σ avoids the situation where $\sum_k u_{ks}^t = 0$ for some cluster, which could happen in the very beginning of the training when the center does not exhibit strength on any distributions. In that case, $v_{sk}^t = \frac{1}{N}$, i.e., the cluster will be updated in a manner that treats all clients equally. Otherwise, since σ is very small, a client with $u_{ks}^t = \sigma$ will be assigned a $v_{sk}^t \approx 0$, and the aggregation weights of other clients will not be affected.

Though calculating and reporting $\{u_{ks}^t\}$ is computationally trivial compared to the actual training procedure, sending centers to all clients may introduce large communication costs. **FedSoft** thus allows the estimations of u_{ks} to be used for up to $\tau \geq 1$ iterations (line 3). In practice, a client can start computing u_{ks}^t for a cluster before it receives all other centers, the delay of transmission is thus tolerable.

Next, relevant clients run proximal local updates to find the minimizer w_k^{t+1} for the proximal objective h_k^t, which entails solving only one optimization problem (line 17). In the case when all clients participate, the cluster models are produced by aggregating all client models: $c_s^{t+1} = \sum_{k=1}^{N} v_{sk}^t w_k^{t+1}$. However, requiring full client participation is impractical in the federated setting. We thus use the client selection trick [61] to reduce the training cost (lines 16). For each cluster s, the algorithm randomly selects a small subset of clients Sel_s^t to participate in the local updating at time t, where $|Sel_s^t| = K < N$.

Clustered FL generally entails more clients to be selected compared to conventional FL, to ensure the convergence of all cluster models. Since **FedSoft** clients can contribute to multiple centers, however, we select only $|\cup_s Sel_s^t|$ clients instead of $\sum_s |Sel_s^t| = SK$ clients in the usual clustered FL. For example, if each distribution has the same share in every client, then in expectation only $N\left(1 - \left(1 - \frac{K}{N}\right)^S\right)$ clients will be selected. This number equals $2K - \frac{K^2}{N}$ when $S = 2$, i.e., $\frac{K^2}{N}$ clients are selected by both clusters.

Once the server receives the local models $\{w_k^{t+1}\}$ for selected clients, it produces the next centers by simply averaging them (line 18). After completion, the algorithm yields trained cluster models $\{c_s^T\}$ as outputs, and each client obtains a personalized local model w_k^T as a byproduct.

6.4 Convergence Analysis

In this section, we provide a convergence guarantee for **FedSoft**. First, note that we can rewrite (6.4) as follows:

$$h_k(w_k; c^t) = \sum_{s, u_{ks} \neq 0} u_{ks} h_{ks}(w_k; c_s^t) \tag{6.7}$$

$$h_{ks}(w_k; c_s^t) \triangleq F_s(w_k) + \frac{\lambda}{2} \frac{u_{ks}^t}{u_{ks}} \|w_k - c_s^t\|^2 \tag{6.8}$$

Here h_{ks} is only defined for $u_{ks} \neq 0$, and we call optimizing each h_{ks} a *sub-problem* for client k.

Our analysis relies on the following assumptions:

Assumption 6.1. *(γ_0-inexact solution) Each client produces a γ_0-inexact solution w_k^{t+1} for the local minimization of (6.4):*

$$\|\nabla h_k(w_k^{t+1}; c^t)\| \leq \gamma_0 \min_s \|\nabla F_s(c_s^t)\| \tag{6.9}$$

Assumption 6.2. *(β-similarity of sub-problems) The sub-problems h_{ks} of each client k have similar optimal points:*

$$\sum_{s'} u_{ks'} \|\nabla h_{ks'}(w_{ks}^*; c_s^t)\|^2 \leq \beta \|\nabla h_{ks}(c_s^t; c_s^t)\|^2, \forall s \tag{6.10}$$

for some $\beta > 0$, where $w_{ks}^* = argmin_{w_{ks}} h_{ks}(w_{ks}; c_s^t)$.

Assumption 6.3. *(Strong convexity and smoothness) Cluster risks are μ_F strongly convex and L_F smooth.*

Assumption 6.4. *(Bounded initial error) At a certain time of the training, all centers have bounded distance from their optimal points. We begin our analysis at that point:*

$$\|c_s^0 - c_s^*\| \leq (0.5 - \alpha_0) \sqrt{\mu_F / L_F} \delta, \forall s \tag{6.11}$$

where $0 < \alpha_0 \leq 0.5$.

Assumption 6.1 assumes significant progress is made on the proximal minimization of h_k, which is a natural extension from assumptions in **FedProx** [52]. Assumption 6.2 ensures the effectiveness of the joint optimization of h_k, i.e., solving one sub-problem can help identify the optimal points of others. Intuitively, if the sub-problems are highly divergent, we would not expect that solving them together would yield a universally good solution. This assumption quantifies our previous reasoning that different distributions co-existing in one local dataset have some similarities, which is the prerequisite for local models to converge and cluster models to be able to learn from each other. Assumption 6.3 is standard [34], and Assumption 6.4 is introduced by [34] in order to bound the estimation error of u_{ks}^t (Lemma 6.1). Note that with Assumption 6.3, each sub-problem h_{ks} is also μ_λ strongly convex and L_λ smooth, where $\mu_\lambda \geq \mu_F, L_\lambda \geq L_F$, and the subscript λ indicates they increase with λ.

To measure the distance of different clusters, we quantify

$$\delta \leq \|c_s^* - c_{s'}^*\| \leq \Delta, \forall s \neq s' \tag{6.12}$$

As we will see later, soft clustered FL performs best when δ and Δ are close. Intuitively, a very small δ indicates two clusters are almost identical, and thus might be better combined into one distribution. On the other hand, a very large Δ implies that two clusters are too divergent, making it hard for one model to acquire useful knowledge from the other.

Next, we bound $\mathbb{E}[u_{ks}^t]$ with respect to the true u_{ks}, for which we reply on the following lemma [34]:

Lemma 6.1. *Suppose Assumptions 6.3 and 6.4 hold. Denoting by $\mathcal{E}_t^{j,j'}$ the event that a data point $(x_j, y_j) \sim \mathcal{P}_j$ is incorrectly classified into cluster $j' \neq j$ at t, there exists a c_ϵ such that*

$$\mathbb{P}(\mathcal{E}_t^{j,j'}) \leq p_\epsilon \triangleq \frac{c_\epsilon}{\alpha_0^2 \delta^4} \tag{6.13}$$

Based on Lemma 6.1, we can bound $\mathbb{E}[u_{ks}^t]$ as follows

Theorem 6.1. *(Bounded estimation errors) The expectation of u_{ks}^t is bounded as*

$$\mathbb{E}[u_{ks}^t] \leq (1 - p_\epsilon) u_{ks} + p'_\epsilon \tag{6.14}$$

Here $p'_\epsilon = p_\epsilon + \sigma$, and the expectation is taken over the randomness of samples.

Next, we seek to characterize each sub-problem h_{ks} at the γ_0-inexact solution w_k^{t+1} that approximately minimizes h_k. Intuitively, w_k^{t+1} should perform better for sub-problems with larger u_{ks}. On the other hand, if $u_{ks} = 0$, we generally cannot expect that w_k^{t+1} will be close to c_s^*. We summarize this intuition in Theorems 6.2 and 6.3.

Theorem 6.2. *(Inexact solutions of sub-problems) If $u_{ks} > 0$, and Assumptions 6.1 to 6.3 hold, then*

$$\|\nabla h_{ks}(w_k^t; c_s^t)\| \leq \frac{\gamma}{\sqrt{u_{ks}}} \|\nabla F_s(c_s^t)\| \tag{6.15}$$

where $\gamma = \sqrt{(\gamma_0^2 + \beta) L_\lambda / \mu_\lambda}$.

Theorem 6.3. *(Divergence of local model to centers) If Assumptions 6.1, 6.3, and 6.4 hold, we have*

$$\|w_k^{t+1} - c_s^t\| \leq r\Delta, \forall s \tag{6.16}$$

where $r = \frac{\gamma S+1}{4}\sqrt{\frac{L_F}{\mu_F}} + \frac{1}{2}\sqrt{\frac{\mu_F}{L_F}} + 1$.

Theorem 6.2 indicates that if the h_k is solved with high quality w_k^{t+1} (small γ_0), and the sub-problems are sufficiently similar (small β), then sub-problems with

$u_{ks} > 0$ can also be well solved by w_k^{t+1}. It also justifies using $v_{sk}^t \propto u_{ks}^t$ as aggregation weights in (6.6). In the case $u_{ks} = 0$, according to Theorem 6.3 (which holds for any u_{ks}), approaching c_s^t with w_k^t will introduce an error of at most $O(\Delta)$.

Finally, we show the convergence of $F_s(c_s^t)$.

Theorem 6.4. *(Convergence of centers) Suppose Assumptions 6.1 to 6.4 hold, and define the quantities:* $n \triangleq \sum_k n_k, n_s \triangleq \sum_k u_{ks} n_k, m_s \triangleq \sum_{k,u_{ks}\neq 0} n_k, \bar{m}_s \triangleq \sum_{k,u_{ks}=0} n_k, \hat{m}_s \triangleq (1-p_\epsilon) m_s + p'_\epsilon \sum_{k,u_{ks}\neq 0} \frac{n_k}{u_{ks}}$. *Suppose* λ *is chosen such that* $\rho \triangleq \frac{n_s - \gamma m_s}{\lambda} - \frac{(\gamma+1) L_F m_s}{\mu_\lambda \lambda} - \frac{p'_\epsilon \bar{m}_s}{2\mu_\lambda} - \frac{L_F(\gamma+1)^2 \hat{m}_s}{2\mu_\lambda^2} - \frac{4 L_F (\gamma+1)^2 \hat{m}_s}{\mu_\lambda^2 \sqrt{K}} - \frac{(\gamma+1)^2 \hat{m}_s + (1-p_\epsilon) n_s + p'_\epsilon n}{\mu_\lambda \sqrt{2K}} > 0$ *and denote* $R \triangleq \frac{1}{2}\left(\mu_\lambda + L_F\right) \bar{m}_s r^2 + \frac{(4L_F + \mu_\lambda) \bar{m}_s r^2}{\sqrt{K}}$. *Then we have*

$$
\begin{aligned}
&\mathbb{E}[F_s(c_s^{t+1})] - F_s(c_s^t) \\
\leq &- \frac{\rho \|\nabla F_s(c_s^t)\|^2}{(1-p_\epsilon) n_s + p'_\epsilon n} + \frac{p'_\epsilon R \Delta^2}{(1-p_\epsilon) n_s - p'_\epsilon (S-2) n}
\end{aligned}
\tag{6.17}
$$

at any time t*, where the expectation is taken over the selection of clients and all* $\{u_{ks}^t\}$.

Corollary 6.1. *Suppose* $F_s(c_s^0) - F_s^* = B_s$. *After* T *iterations,*

$$
\sum_{t=1}^{T} \frac{\rho \mathbb{E} \|\nabla F_s(c_s^t)\|^2}{(1-p_\epsilon) n_s + p'_\epsilon n} \leq \frac{B_s}{T} + O(p_\epsilon \Delta^2) \tag{6.18}
$$

The choices of λ to make $\rho > 0$ is discussed in [52]. From Corollary 6.1, the gradient norm converges to a remaining error controlled by p_ϵ. Intuitively, when $c_s^t = c_s^*$, further updating c_s^t with misclassified models will inevitably move c_s^t away from c_s^*. This bias cannot be removed unless we have a perfect guess of u_{ks}. Recall that $p_\epsilon = O(\frac{1}{\delta^4})$, and thus the remaining term is $O(\frac{\Delta^2}{\delta^4})$, which decreases as Δ approaches δ. Thus, **FedSoft** performs better when the divergences between clusters are more homogeneous. Note that Corollary 6.1 seems to imply the remaining error will explode if $\delta \to 0$, but Lemma 6.1 is only valid when $p_\epsilon < 1$. Thus when δ is very small, i.e., there exist two distributions that are extremely similar, the remaining error is determined by the maximum divergence of the other distributions. Furthermore, the divergence Δ determines the degree of non-IID of a local dataset (not among clients), which also implicitly affects the accuracy of local solutions γ_0. Intuitively, a larger Δ implies it is more difficult to exactly solve a local problem involving multiple distributions, resulting in a greater γ_0.

To see the role of cluster heterogeneity, suppose $\|c_1^* - c_2^*\|$ is closer than the distance of all other centers to c_1, then the misclassified samples for cluster 1 are more likely to be matched to cluster 2. Thus, cluster 2 gets more updates from data that it does not own, producing greater remaining training error that drags its center towards cluster 1. On the other hand, if the cluster divergence is homogeneous, then the effect of mis-classification is amortized among all clusters, resulting in a universally smaller remaining error.

Theorem 6.4 shows the convergence of cluster models $\{c_s\}$ in terms of the cluster risks $\{F_s\}$. For the local models $\{w_k\}$, we focus on how clients integrate global knowledge into their local personalizations, which cannot be captured only with the original client risk functions $\{f_k(w)\}$. Thus, we are interested in the convergence performance of $\{w_k\}$ with respect to the proximal objective $\{h_k\}$. Note that under Assumption 6.3, **FedSoft** is effectively a cyclic block coordinate descent algorithm on a jointly convex objective function of $\{w_k\}$ and $\{c_s\}$, for which the convergence is guaranteed:

Theorem 6.5. *(Joint convergence of cluster and client models) For fixed importance weights* $\tilde{u}$*, let* $w^*, c^* = \text{argmin} \sum_{k=1}^{N} h_k(w_k; c, \tilde{u}_k)$*, and* w^T, c^T *be the outputs of* ***FedSoft****. Then* $w^T \to w^*, c^T \to c^*$ *linearly with* T*.*

The impact of τ on the convergence. Note that τ only affects the accuracy of the importance weight estimations u_{ks}^t, which determines the estimation error p_ϵ.

To incorporate τ into the analysis, we first generalize Assumption 6.4 as follows [34]

$$\|c_s^t - c_s^*\| \leq (0.5 - \alpha_t) \sqrt{\mu_F / L_F} \delta, \forall s \tag{6.19}$$

where $0 < \alpha_t \leq 0.5$ for all t.

If the algorithm works correctly, the distance between c_s^t and c_s^* should decrease over time, thus α_t will gradually increase from α_0 to 0.5. Then we can change the definition of p_ϵ in Lemma 6.1:

$$\mathbb{P}(\mathcal{E}_t^{j,j'}) \leq p_\epsilon^{t,\tau} \triangleq \frac{c_\epsilon}{\alpha_{\tau\lfloor t/\tau \rfloor}^2 \delta^4} \tag{6.20}$$

Here we replace α_0 with $\alpha_{\tau\lfloor t/\tau \rfloor}$, which takes the same value within each estimation interval. Since we expect α_t to be increasing, we have $\alpha_{\tau\lfloor t/\tau \rfloor} \leq \alpha_t$. Thus, the estimation error $p_\epsilon^{t,\tau}$ increases when we choose a larger interval τ. Plugging this new definition of $p_\epsilon^{t,\tau}$ to Corollary 6.1 we have

$$\sum_{t=1}^{T} \frac{\rho^{t,\tau}\mathbb{E}\|\nabla F_s(c_s^t)\|^2}{(1-p_\epsilon^{t,\tau})n_s + (p_\epsilon^{t,\tau})'n} \leq \frac{B_s}{T} + O(p_\epsilon^{0,\tau}\Delta^2) \tag{6.21}$$

where $\rho^{t,\tau}$ is defined by replacing all p_ϵ with $p_\epsilon^{t,\tau}$. This gives us the same asymptotic convergence rate with time as in Corollary 6.1, except for small differences in constant terms.

6.5 Experiments

In this section, we verify the effectiveness of **FedSoft** with two base datasets under various mixture patterns.

6.5.1 Experiment Settings

For all experiments, we use $N = 100$ clients, and the number of samples in each client n_k is chosen uniformly at random from 100 to 200. For ease of demonstration, for every base dataset, we first investigate the mixture of $S = 2$ distributions and then increase S. In the case with two distributions, suppose the cluster distributions are named $\mathcal{D}_A$ and $\mathcal{D}_B$. We evaluate the following partition patterns:

- 10:90 partition: 50 clients have a mixture of 10% $\mathcal{D}_A$ and 90% $\mathcal{D}_B$, and 50 have 10% $\mathcal{D}_B$ and 90% $\mathcal{D}_A$.
- 30:70 partition: Same as above except the ratio is 30:70.
- Linear partition: Client k has $(0.5 + k)\%$ data from $\mathcal{D}_A$ and $(99.5 - k)\%$ data from $\mathcal{D}_B$, $k = 0, \cdots, 99$.

We further introduce the random partition, where each client has a random mixture vector generated by dividing the $[0, 1]$ range into S segments with $S - 1$ points drawn from Uniform$(0, 1)$. We use all four partitions for $S = 2$, and only use the random partition when $S > 2$ for simplification. Each partition produces non-IID local distributions, i.e., clients have different local data distributions. Specifically, the 10:90 and 30:70 partitions yield 2 local distributions, while the linear and random partitions yield 100. Unless otherwise noted, we choose **FedSoft**'s estimation interval

$\tau = 2$, client selection size $K = 60$, counter smoother σ = 1e-4, and all experiments are run until both cluster and client models have fully converged. All models are randomly initialized with the Xavier normal [35] initializer without pre-training, so that the association among clients, centers, and cluster distributions is built automatically during the training process.

We compare **FedSoft** with two baselines: **IFCA** [34] and **FedEM** [59]. Both baseline algorithms produce one center for each cluster, but they do not explicitly generate local models as in **FedSoft**. Nevertheless, they also estimate the importance weights for each client, and we thus use the center corresponding to the largest importance weight as a client's local model. Since we expect cluster models will be deployed to new users, we evaluate their test accuracy/error on holdout datasets sampled from the corresponding cluster distributions. For local models, they are expected to fit the local data of participating clients, we hence evaluate their accuracy/error on local training datasets. Throughout this section, we use $\bar{c}$ and $\bar{w}$ to represent the average accuracy/error of the cluster and client models, not the accuracy/error of the averaged models.

We use two base datasets to generate the various cluster distributions. A different model is adopted for each dataset.

- **Synthetic Data**. We generate synthetic datasets according to $y_i = \langle x_i, \theta_s \rangle + \epsilon_i$ where $\theta_s \sim \mathcal{N}(0, \sigma_0^2 I_{10})$, $x_i \sim \mathcal{N}(0, I_{10})$, $\epsilon_i \sim \mathcal{N}(0, 1)$ [34]. Unless otherwise noted, we use $\sigma_0 = 10$. We use a conventional linear regression model without the intercept term. All clients use Adam as the local solver. The number of local epochs equals 10, batch size equals 10, and the initial learning rate equals 5e-3. The same solver is used for both **FedSoft** and the baselines. The regularization weight $\lambda = 1.0$ for **FedSoft**.
- **EMNIST Letters**. We use the handwritten images of English letters in the EMNIST dataset to create 2 distributions for the lower and uppercase letters [36], each with 26 classes. Then we rotate these images counterclockwise by 90° [56], resulting in 4 total distributions. In the $S = 2$ setting we compare the two 0° distributions. A rotation variant CNN model is used for this dataset. We use a CNN model comprising two convolutional layers with kernel size equal to 5 and padding equal to 2, each followed by the max-pooling with kernel size equal to 2, then connected to a fully-connected layer with 512 hidden neurons

followed by ReLU. All clients use Adam as the local solver, with the number of local epochs equal to 5, batch size equal to 5, and initial learning rate equal to 1e-5. The same solver is used for both **FedSoft** and the baselines. The regularization weight $\lambda = 0.1$ for **FedSoft**.

In general, the letter distributions share more similarities with each other, while the synthetic distributions are more divergent, e.g., letters like "O" have very similar upper and lowercase shapes and are invariant to rotations. On the other hand, data generated from $y = x$ and $y = -x$ can be easily distinguished. We thus expect the mixture of synthetic data to benefit more from the personalization ability of **FedSoft**.

6.5.2 Performance Evaluation

The typical convergence process of **FedSoft** is shown in Figure 6.1 for the mixture of two synthetic distributions under the 10:90 partition. The left/right columns represent the first/second distributions. Center indices are assigned randomly in the beginning. The importance weight estimations $\bar{u}^t_{a:b}$ are averaged on clients with the mixture coefficients $a : b$ (i.e., they have the same local distribution). In this example of the synthetic data, **FedSoft** is able to automatically distinguish the two cluster distributions. After around 5 global epochs, center 1 starts to exhibit strength on the first cluster distribution, and center 0 concentrates on the other, which implies a correct association between centers and cluster distributions. Similarly, the importance weight estimations u^t_{ks}, which are initially around 50:50, soon converge to the real mixture ratio 10:90.

Table 6.1 lists the mean squared error (MSE) or accuracy of the output cluster models. Each row represents the distribution of a test dataset. The center with the smallest error or highest accuracy is underlined for each test distribution. **FedSoft** produces high quality centers under all mixture patterns. In particular, each center exhibits strength on one distribution, which indicates that **FedSoft** builds correct associations for the centers and cluster distributions. The performance gap between two distributions using the same center is larger for the synthetic data. This is because the letter distributions have smaller divergence than the synthetic distributions. Thus, letter models can more easily transfer the knowledge of one distribution to another, and a center focusing on one distribution can perform well on the other. Notably,

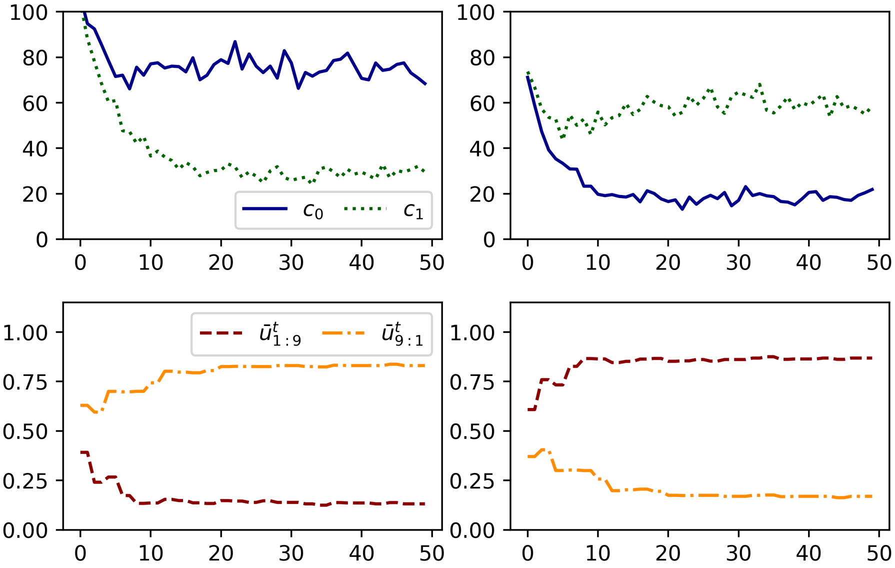


Figure 6.1: Evolution of the test mean squared error of centers (top) and the importance weight estimation of clients (bottom) over time

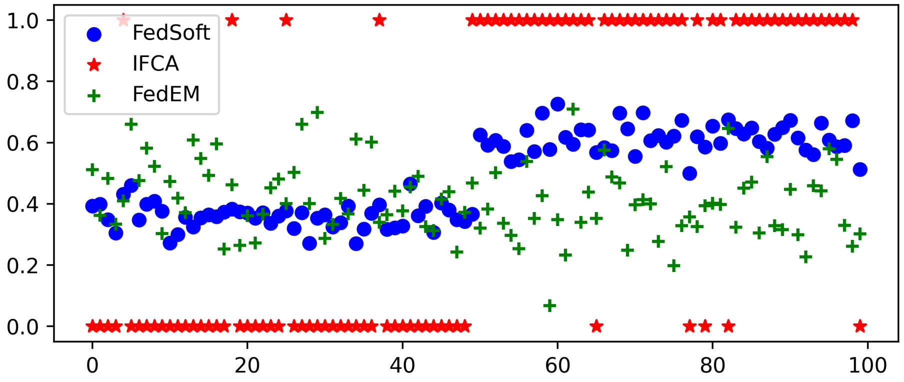


Figure 6.2: The clients' estimation of importance weights on the first cluster (u^T_{k0})

the 30:70 mixture has the worst performance for both datasets, which is due to the degrading of local solvers when neither distribution dominates. Thus, the local problems under this partition are solved less accurately, resulting in poor local models and a large value of γ in Theorem 6.2, which then produces high training loss on cluster models according to Theorem 6.4.

Table 6.2 compares **FedSoft** with the baselines. Here c^*_{lo}/c^*_{up} represents the accuracy of the center that performs best on the lower/upper distribution, and the number in the parenthesis indicates the index of that center. $\bar{w}$ is the accuracy of

Table 6.1: MSE or accuracy of cluster models for the mixture of two distributions

Synthetic data: mean squared error

	10:90		30:70		Linear		Random	
	c_0	c_1	c_0	c_1	c_0	c_1	c_0	c_1
θ_0	68.4	$\underline{29.5}$	$\underline{44.2}$	49.6	$\underline{38.2}$	59.1	$\underline{42.2}$	60.6
θ_1	$\underline{21.8}$	58.6	41.3	$\underline{36.3}$	47.1	$\underline{27.8}$	42.7	$\underline{27.0}$

EMNIST letters: accuracy (%)

	10:90		30:70		Linear		Random	
	c_0	c_1	c_0	c_1	c_0	c_1	c_0	c_1
Lower	68.9	$\underline{70.3}$	$\underline{65.9}$	65.8	$\underline{71.8}$	71.7	72.0	$\underline{72.5}$
Upper	$\underline{74.6}$	73.3	70.1	$\underline{70.4}$	73.9	$\underline{74.1}$	$\underline{77.7}$	77.2

Table 6.2: Comparison between FedSoft and baselines on the letters data

	10:90			Linear		
	c^*_{lo}	c^*_{up}	$\bar{w}$	c^*_{lo}	c^*_{up}	$\bar{w}$
FedSoft	70.3(1)	74.6(0)	90.9	71.8(0)	74.1(1)	86.5
IFCA	58.5(0)	61.3(1)	65.2	55.4(0)	57.2(0)	62.9
FedEM	67.4(1)	69.8(1)	63.6	65.9(0)	69.0(0)	62.4

local models averaged over all clients. Not only does **FedSoft** produce more accurate cluster and local models, but it also achieves better balance between the two trained centers. Similarly, Figure 6.2 shows the importance estimation of clients for the first cluster. **FedSoft** and **IFCA** are able to build the correct association (though the latter is a hard partition), while **FedEM** appears to be biased to the other center by putting less weights (< 0.5) on the first one.

Next, we evaluate the algorithm with the random partition for the mixture of more distributions. Tables 6.3 and 6.4 show the MSE or accuracy of cluster models for the mixture of 8 and 4 distributions on synthetic and letters data, where we still observe high-quality outcomes and good association between centers and cluster distributions.

6.5.3 Impact of Regularization Weight

In previous experiments on the letters dataset, we choose $\lambda = 0.1$, which is selected through grid search. We show in Table 6.5 the accuracy of cluster and client models for different choices of λ on the linear partition of letters dataset, while all other

Table 6.3: Test MSE of centers for the mixture of 8 synthetic distributions with randomly generated weights $\theta_0 \cdots \theta_7$

	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7
θ_0	62.2	62.7	64.0	63.2	61.4	$\underline{57.6}$	63.7	61.9
θ_1	65.0	67.8	69.4	65.8	64.3	67.1	$\underline{64.2}$	64.5
θ_2	60.1	59.9	$\underline{57.8}$	58.6	62.6	63.2	60.0	59.9
θ_3	96.8	95.4	96.8	98.8	$\underline{93.2}$	93.8	95.3	96.8
θ_4	86.1	89.8	91.8	87.3	85.4	86.6	85.6	$\underline{84.9}$
θ_5	161.2	160.3	$\underline{156.0}$	160.0	164.7	167.3	162.0	163.6
θ_6	110.2	106.7	$\underline{104.8}$	109.0	111.0	107.8	111.5	110.1
θ_7	34.5	$\underline{33.8}$	34.8	33.9	34.2	34.1	34.4	35.0

Table 6.4: Test accuracy (%) of centers for the mixture of 4 distributions with original and 90°-rotated letter images

	c_0		c_1		c_2		c_3	
	0°	90°	0°	90°	0°	90°	0°	90°
Lower	71.5	$\underline{67.6}$	71.3	67.3	71.3	$\underline{67.6}$	$\underline{72.3}$	67.3
Upper	70.2	71.7	$\underline{70.8}$	71.3	70.3	$\underline{71.9}$	70.3	71.0

parameters are kept unchanged. As we can see, when $\lambda = 0$, no global knowledge is passed to clients, thus the local training is done separately without any cooperation, resulting in poorly trained models. On the other hand, when λ is increased to 1, the local updating is dominated by fitting the local model to the average of global models, and the local knowledge is less emphasized, which also reduces the algorithm performance.

Table 6.5: Accuracy of cluster and client models for different choices of λ

	$\lambda = 0$			$\lambda = 0.1$			$\lambda = 1$		
	c_0	c_1	$\bar{w}$	c_0	c_1	$\bar{w}$	c_0	c_1	$\bar{w}$
Lower	48.4	$\underline{50.1}$	-	$\underline{71.8}$	71.7	-	55.1	$\underline{55.2}$	-
Upper	$\underline{47.7}$	47.5	-	73.9	$\underline{74.1}$	-	$\underline{58.7}$	58.6	-
Local	-	-	72.7	-	-	86.5	-	-	63.6

6.5.4 Impact of Divergence in Distributions

Finally, we show how the divergence among different distributions Δ affects the performance of **FedSoft**. For this experiment we use the mixture of two synthetic distributions under the random partition, and we control the divergence by choosing different values of σ_0 (i.e. Δ increases with σ_0). As we can see, the MSE significantly increases as the divergence between distributions gets larger, which validates Theorem 6.4.

Table 6.6: MSE of cluster and client models for different choices of σ_0

	$\sigma_0 = 1$			$\sigma_0 = 10$			$\sigma_0 = 50$			$\sigma_0 = 100$		
	c_0	c_1	$\bar{w}$	c_0	c_1	$\bar{w}$	c_0	c_1	$\bar{w}$	c_0	c_1	$\bar{w}$
θ_0	5.9	$\underline{4.2}$	-	$\underline{42.2}$	60.6	-	225.3	$\underline{89.9}$	-	782.4	$\underline{454.4}$	-
θ_1	$\underline{3.2}$	4.6	-	42.7	$\underline{27.0}$	-	$\underline{117.6}$	89.9	-	$\underline{432.8}$	812.0	-
Local	-	-	0.6	-	-	4.96	-	-	18.8	-	-	78.3

6.6 Summary

This chapter proposes **FedSoft**, an efficient algorithm generalizing traditional clustered federated learning approaches to allow clients to sample data from a mixture of distributions. By incorporating proximal local updating, **FedSoft** enables simultaneous training of cluster models for future users, and personalized local models for participating clients, which is achieved without increasing the workload of clients. Theoretical analysis shows the convergence of **FedSoft** for both cluster and client models, and the algorithm exhibits good performance in experiments with various mixture patterns.

6.7 Proof of Theorems

6.7.1 Proof of Theorem 6.1

Let $G_{ki}, i = 1, 2 \cdots$ be some virtual group of client k's data points that follow the same distribution. Thus,

Proof.

$$\begin{aligned}\mathbb{E}[u_{ks}^t] \leq & \frac{1}{n_k}\left(\sum_{G_{ki} \sim \mathcal{P}_s} \mathbb{P}\left(\mathrm{argmin}_{s'} F_{s'}(G_{ki}) = s\right) |G_{ki}| \right. \\ & \left. + \sum_{G_{ki} \not\sim \mathcal{P}_s} \mathbb{P}\left(\mathrm{argmin}_{s'} F_{s'}(G_{ki}) = s\right) |G_{ki}| \right) + \sigma \\ \leq & \frac{1}{n_k}\left(n_{ks} + \sum_{G_{ki} \not\sim \mathcal{P}_s} p_\epsilon |G_{ki}| \right) = \frac{1}{n_k}\left(n_{ks} + p_\epsilon(n_k - n_{ks})\right) + \sigma \\ = & (1 - p_\epsilon) u_{ks} + p'_\epsilon \end{aligned} \tag{6.22}$$

□

6.7.2 Proof of Theorem 6.2

Proof. For simplification we drop the dependency of c_s^t in h_k and h_{ks}. Take $w_{ks}^* = \mathrm{argmin}_{w_{ks}} h_{ks}(w_{ks})$, we have

$$\begin{aligned} & \gamma_0^2 \|\nabla F_s(c_s^t)\|^2 \geq \|\nabla h_k(w_k^t)\|^2 \geq 2\mu_\lambda \left(h_k(w_k^t) - h_k(w_{ks}^*)\right) \\ = & 2\mu_\lambda \sum_{s'} u_{ks'} \left(h_{ks'}(w_k^t) - h_{ks'}(w_{ks}^*)\right) \\ = & 2\mu_\lambda u_{ks} \left(h_{ks}(w_k^t) - h_{ks}^*\right) - 2\mu_\lambda \sum_{s' \neq s} u_{ks'} \left(h_{ks'}(w_{ks}^*) - h_{ks'}(w_k^t)\right) \\ \geq & \frac{\mu_\lambda}{L_\lambda} u_{ks} \|\nabla h_{ks}(w_k^t)\|^2 - \sum_{s' \neq s} u_{ks'} \|\nabla h_{ks'}(w_{ks}^*)\|^2 \\ \geq & \frac{\mu_\lambda}{L_\lambda} u_{ks} \|\nabla h_{ks}(w_k^t)\|^2 - \beta \|\nabla F_s(c_s^t)\|^2 \end{aligned} \tag{6.23}$$

Thus,

$$\frac{\mu_\lambda}{L_\lambda} u_{ks} \|\nabla h_{ks}(w_k^t)\|^2 \leq \left(\gamma_0^2 + \beta\right) \|\nabla F_s(c_s^t)\|^2 \tag{6.24}$$

Reorganizing, we have

$$\|\nabla h_{ks}(w_k^t)\| \leq \frac{\sqrt{(\gamma_0^2 + \beta) L_\lambda / \mu_\lambda}}{\sqrt{u_{ks}}} \|\nabla F_s(c_s^t)\| = \frac{\gamma}{\sqrt{u_{ks}}} \|\nabla F_s(c_s^t)\| \tag{6.25}$$

□

6.7.3 Proof of Theorem 6.3

Proof. Let $s' = \text{argmax}_s u_{ks}$, we have $u_{ks'} \geq \frac{1}{S}$, thus

$$\begin{aligned}
&\|w_k^{t+1} - c_s^t\| = \|w_k^{t+1} - c_{s'}^t + c_{s'}^t - c_s^t\| \\
\leq & \frac{1}{\mu_\lambda} \|\nabla h_{ks'}(w_k^{t+1}; c_{s'}^t) - \nabla h_{ks'}(c_{s'}^t; c_{s'}^t)\| + \|c_{s'}^t - c_s^t\| \\
\leq & \frac{1}{\mu_F} \left(\frac{\gamma}{u_{ks'}} \|\nabla F_{s'}(c_{s'}^t)\| + \|\nabla F_{s'}(c_{s'}^t)\| \right) + \frac{1}{2} \sqrt{\frac{\mu_F}{L_F}} \delta + \Delta \\
\leq & \frac{(\gamma S + 1) L_F}{\mu_F} \|c_{s'}^t - c_{s'}^*\| + \left(\frac{1}{2} \sqrt{\frac{\mu_F}{L_F}} + 1 \right) \Delta \\
\leq & \left(\frac{\gamma S + 1}{4} \sqrt{\frac{L_F}{\mu_F}} + \frac{1}{2} \sqrt{\frac{\mu_F}{L_F}} + 1 \right) \Delta
\end{aligned} \tag{6.26}$$

□

6.7.4 Proof of Theorem 6.4

We first introduce the following lemma:

Lemma 6.2. $\mathbb{E}\left[\frac{u_{ks}^t n_k}{\sum_{k'} u_{k's}^t n_{k'}}\right] \leq \mathbb{E}[u_{ks}^t n_k] \mathbb{E}\left[\frac{1}{\sum_{k'} u_{k's}^t n_{k'}}\right]$, *where the expectation is taken over* $\{u_{ks}^t\}$.

Proof. Let $r^t = \sum_{k' \neq k} u_{k's}^t n_{k'}$, and note that $u_{ks}^t \perp r^t$ since each client estimates its u_{ks}^t independently. Thus,

$$
\begin{aligned}
&\mathbb{E}\left[\frac{u_{ks}^t n_k}{\sum_{k'} u_{k's}^t n_{k'}}\right] = \mathbb{E}\left[\frac{u_{ks}^t n_k}{u_{ks}^t n_k + r^t}\right] = \mathbb{E}\left[1 - \frac{r^t}{u_{ks}^t n_k + r^t}\right] \\
=&\mathbb{E}_{r^t}\left[\mathbb{E}_{\{u_{ks}^t\}|r^t}\left[1 - \frac{r^t}{u_{ks}^t n_k + r^t}\right]\right] \leq \mathbb{E}_{r^t}\left[1 - \frac{r^t}{\mathbb{E}[u_{ks}^t n_k | r^t] + r^t}\right] \\
=&\mathbb{E}[u_{ks}^t n_k]\mathbb{E}_{r^t}\left[\frac{1}{\mathbb{E}[u_{ks}^t n_k] + r^t}\right] \leq \mathbb{E}[u_{ks}^t n_k]\mathbb{E}_{r^t}\left[\mathbb{E}_{\{u_{ks}^t\}|r^t}\left[\frac{1}{u_{ks}^t n_k + r^t}\right]\right] \\
=&\mathbb{E}[u_{ks}^t n_k]\mathbb{E}\left[\frac{1}{\sum_{k'} u_{k's}^t n_{k'}}\right]
\end{aligned}
\tag{6.27}
$$

□

We then formally prove Theorem 6.4:

Proof. For $u_{ks} \neq 0$, define

$$
e_{ks}^{t+1} \triangleq \nabla h_{ks}(w_k^{t+1}; c_s^t) = \nabla F_s(w_k^{t+1}) + \lambda \frac{u_{ks}^t}{u_{ks}}(w_k^{t+1} - c_s^t) \tag{6.28}
$$

using Theorem 6.2, we have

$$
\|e_{ks}^{t+1}\| \leq \frac{\gamma}{\sqrt{u_{ks}}}\|\nabla F_s(c_s^t)\| \leq \frac{\gamma}{u_{ks}}\|\nabla F_s(c_s^t)\| \tag{6.29}
$$

Define

$$
\bar{c}_s^{t+1} \triangleq \sum_k v_{sk}^t w_k^{t+1} = \mathbb{E}_{v_{sk}^t}[w_k^{t+1}] \tag{6.30}
$$

thus,

$$
\begin{aligned}
\bar{c}_s^{t+1} - c_s^t = &-\frac{1}{\lambda}\sum_{k, u_{ks}\neq 0}\left(\frac{u_{ks} n_k}{\sum_{k'} u_{k's}^t n_{k'}}\nabla F_s(w_k^{t+1}) - \frac{u_{ks} n_k}{\sum_{k'} u_{k's}^t n_{k'}} e_{ks}^{t+1}\right) \\
&+ \sum_{k, u_{ks}=0} v_{sk}^t (w_k^{t+1} - c_s^t)
\end{aligned}
\tag{6.31}
$$

Next we bound $\|w_k^{t+1} - c_s^t\|$ for $u_{ks} \neq 0$,

$$
\begin{aligned}
&\|w_k^{t+1} - c_s^t\| \leq \frac{1}{\mu_\lambda}\|\nabla h_{ks}(w_k^{t+1}; c_s^t) - \nabla h_{ks}(c_s^t; c_s^t)\| \\
\leq &\left(\frac{\gamma}{\mu_\lambda \sqrt{u_{ks}}} + \frac{1}{\mu_\lambda}\right)\|\nabla F_s(c_s^t)\| \leq \frac{\gamma+1}{\mu_\lambda \sqrt{u_{ks}}}\|\nabla F_s(c_s^t)\| \leq \frac{\gamma+1}{\mu_\lambda u_{ks}}\|\nabla F_s(c_s^t)\|
\end{aligned}
\tag{6.32}
$$

Therefore,

$$\begin{aligned}&\|\bar{c}_s^{t+1}-c_s^t\|^2 \leq \mathbb{E}_{v_{sk}^t}[\|w_k^{t+1}-c_s^t\|^2]\\ \leq&\left(\frac{\gamma+1}{\mu_\lambda}\right)^2 \sum_{k,u_{ks}\neq 0} \frac{u_{ks}^t n_k}{u_{ks}\sum_{k'} u_{k's}^t n_{k'}}\|\nabla F_s(c_s^t)\|^2 + \sum_{k,u_{ks}=0} \frac{u_{ks}^t n_k}{\sum_{k'} u_{k's}^t n_{k'}} r^2\Delta^2\end{aligned} \tag{6.33}$$

Define M_s^{t+1} such that $\bar{c}_s^{t+1}-c_s^t = -\frac{1}{\lambda}\left(\frac{\sum_k u_{ks}n_k}{\sum_k u_{ks}^t n_k}\nabla F_s(c_s^t) + M_s^{t+1}\right)$

$$\begin{aligned}M_s^{t+1} =& \sum_{k,u_{ks}\neq 0}\left(\frac{u_{ks}n_k}{\sum_{k'} u_{k's}^t n_{k'}}\nabla F_s(w_k^{t+1}) - \frac{u_{ks}n_k}{\sum_{k'} u_{k's}^t n_{k'}} e_{ks}^{t+1}\right)\\ &- \frac{\sum_k u_{ks}n_k}{\sum_k u_{ks}^t n_k}\nabla F_s(c_s^t) - \lambda \sum_{k,u_{ks}=0} v_{sk}^t (w_k^{t+1}-c_s^t)\\ =&\frac{1}{\sum_k u_{ks}^t n_k}\sum_{k,u_{ks}\neq 0}\left(u_{ks}n_k\left(\nabla F_s(w_k^{t+1}) - \nabla F_s(c_s^t)\right) - u_{ks}n_k e_{ks}^{t+1}\right)\\ &- \lambda\sum_{k,u_{ks}=0} v_{sk}^t(w_k^{t+1}-c_s^t)\end{aligned} \tag{6.34}$$

thus,

$$\begin{aligned}\|M_{t+1}\| \leq& \frac{1}{\sum_k u_{ks}^t n_k}\sum_{k,u_{ks}\neq 0}\left(u_{ks}n_k L_F\|w_k^{t+1}-c_s^t\| + u_{ks}n_k\|e_{ks}^{t+1}\|\right)\\ &+ \lambda\sum_{k,u_{ks}=0} v_{sk}^t\|w_k^{t+1}-c_s^t\|\\ \leq&\frac{m_s}{\sum_k u_{ks}^t n_k}\left(\frac{(\gamma+1)L_F}{\mu_\lambda}+\gamma\right)\|\nabla F_s(c_s^t)\| + \lambda\sum_{k,u_{ks}=0}\frac{u_{ks}^t n_k}{\sum_{k'} u_{k's}^t n_{k'}} r\Delta\end{aligned} \tag{6.35}$$

Using the smoothness of F_s, we have

$$
\begin{aligned}
&F_s(\bar{c}_s^{t+1}) \le F_s(c_s^t) + \langle \nabla F_s(c_s^t), \bar{c}_s^{t+1} - c_s^t \rangle + \frac{L_F}{2}\|\bar{c}_s^{t+1} - c_s^t\|^2 \\
\le & F_s(c_s^t) - \frac{1}{\lambda}\frac{\sum_k u_{ks} n_k}{\sum_k u_{ks}^t n_k}\|\nabla F_s(c_s^t)\|^2 - \frac{1}{\lambda}\langle \nabla F_s(c_s^t), M_s^{t+1} \rangle + \frac{L_F}{2}\|\bar{c}_s^{t+1} - c_s^t\|^2 \\
\le & F_s(c_s^t) - \frac{1}{\lambda}\frac{n_s}{\sum_k u_{ks}^t n_k}\|\nabla F_s(c_s^t)\|^2 + \frac{m_s}{\lambda \sum_k u_{ks}^t n_k}\left(\frac{(\gamma+1)L_F}{\mu_\lambda} + \gamma\right) \cdot \\
&\|\nabla F_s(c_s^t)\|^2 + \frac{L_F}{2}\left(\frac{\gamma+1}{\mu_\lambda}\right)^2 \sum_{k,u_{ks}\neq 0} \frac{u_{ks}^t n_k}{u_{ks}\sum_{k'} u_{k's}^t n_{k'}}\|\nabla F_s(c_s^t)\|^2 \\
+ & \left(\sum_{k,u_{ks}=0} \frac{u_{ks}^t n_k}{\sum_{k'} u_{k's}^t n_{k'}} r\Delta\right)\|\nabla F_s(c_s^t)\| + \frac{L_F}{2}\sum_{k,u_{ks}=0} \frac{u_{ks}^t n_k}{\sum_{k'} u_{k's}^t n_{k'}} r^2\Delta^2 \qquad (6.36) \\
= & F_s(c_s^t) - \frac{1}{\sum_k u_{ks}^t n_k}\left(\frac{n_s - \gamma m_s}{\lambda} - \frac{(\gamma+1)L_F m_s}{\mu_\lambda \lambda}\right)\|\nabla F_s(c_s^t)\|^2 \\
+ & \left(\sum_{k,u_{ks}=0} \frac{u_{ks}^t n_k}{\sum_{k'} u_{k's}^t n_{k'}} r\Delta\right)\|\nabla F_s(c_s^t)\| + \frac{L_F}{2}\sum_{k,u_{ks}=0} \frac{u_{ks}^t n_k}{\sum_{k'} u_{k's}^t n_{k'}} r^2\Delta^2 \\
+ & \frac{L_F}{2}\left(\frac{\gamma+1}{\mu_\lambda}\right)^2 \sum_{k,u_{ks}\neq 0} \frac{u_{ks}^t n_k}{u_{ks}\sum_{k'} u_{k's}^t n_{k'}}\|\nabla F_s(c_s^t)\|^2
\end{aligned}
$$

Taking expectations over $\{u_{ks}^t\}$, and applying Lemma 6.2, we have

$$
\begin{aligned}
&\mathbb{E}[F_s(\bar{c}_s^{t+1})] \le F_s(c_s^t) - \mathbb{E}\left[\frac{1}{\sum_k u_{ks}^t n_k}\right]\Bigg\{ - \left(p'_\epsilon \bar{m}_s r\Delta\right)\|\nabla F_s(c_s^t)\| \\
&+ \left(\frac{n_s - \gamma m_s}{\lambda} - \frac{(\gamma+1)L_F m_s}{\mu_\lambda \lambda}\right)\|\nabla F_s(c_s^t)\|^2 \\
&- \frac{L_F}{2}\left(\frac{\gamma+1}{\mu_\lambda}\right)^2 \hat{m}_s\|\nabla F_s(c_s^t)\|^2 - \frac{L_F}{2} p'_\epsilon \bar{m}_s r^2\Delta^2 \Bigg\} \qquad (6.37) \\
&\le F_s(c_s^t) + \mathbb{E}\left[\frac{1}{\sum_k u_{ks}^t n_k}\right]\frac{p'_\epsilon}{2}\left(\mu_\lambda + L_F\right)\bar{m}_s r^2\Delta^2 - \mathbb{E}\left[\frac{1}{\sum_k u_{ks}^t n_k}\right] \cdot \\
&\quad \left(\frac{n_s - \gamma m_s}{\lambda} - \frac{(\gamma+1)L_F m_s}{\mu_\lambda \lambda} - \frac{p'_\epsilon \bar{m}_s}{2\mu_\lambda} - \frac{L_F(\gamma+1)^2\hat{m}_s}{2\mu_\lambda^2}\right)\|\nabla F_s(c_s^t)\|^2
\end{aligned}
$$

Define

$$
\rho_0 \triangleq \frac{n_s - \gamma m_s}{\lambda} - \frac{(\gamma+1)L_F m_s}{\mu_\lambda \lambda} - \frac{p'_\epsilon \bar{m}_s}{2\mu_\lambda} - \frac{L_F(\gamma+1)^2\hat{m}_s}{2\mu_\lambda^2} \tag{6.38}
$$

$$R_0 \triangleq \frac{1}{2}\left(\mu_\lambda + L_F\right)\bar{m}_s r^2 \tag{6.39}$$

then

$$\begin{aligned}\mathbb{E}[F_s(\bar{c}_s^{t+1})] \leq & F_s(c_s^t) - \mathbb{E}\left[\frac{1}{\sum_k u_{ks}^t n_k}\right]\rho_0\|\nabla F_s(c_s^t)\|^2 \\ & + \mathbb{E}\left[\frac{1}{\sum_k u_{ks}^t n_k}\right] p'_\epsilon R_0 \Delta^2\end{aligned} \tag{6.40}$$

Next we incorporate the client selection.

We can write $c_s^{t+1} = \frac{1}{K}\sum_{l=1}^{K}\hat{c}_{s,l}^{t+1}$, where $\hat{c}_{s,l}^{t+1} = \sum_{k=1}^{N}\mathbf{1}(k \in Sel_{s,l}^t)w_k^{t+1}$, and

$$\mathbb{P}(k \in Sel_{s,l}^t) = \frac{u_{ks}^t n_k}{\sum_{k'} u_{k's}^t n_{k'}} = v_{sk}^t \tag{6.41}$$

Thus,

$$\mathbb{E}_{Sel_s^t}[\hat{c}_{s,l}^{t+1}] = \mathbb{E}_{v_{sk}^t}[w_k^{t+1}] \tag{6.42}$$

We then bound the difference between $F_s(c_s^{t+1})$ and $F_s(\bar{c}_s^{t+1})$

$$\begin{aligned}F_s(c_s^{t+1}) \leq\ & F_s(\bar{c}_s^{t+1}) + \langle\nabla F_s(\bar{c}_s^{t+1}), c_s^{t+1} - \bar{c}_s^{t+1}\rangle + \frac{L_F}{2}\|c_s^{t+1} - \bar{c}_s^{t+1}\|^2 \\ \leq & F_s(\bar{c}_s^{t+1}) + \left(\|\nabla F_s(\bar{c}_s^{t+1})\| + \frac{L_F}{2}\|c_s^{t+1} - \bar{c}_s^{t+1}\|\right)\|c_s^{t+1} - \bar{c}_s^{t+1}\| \\ \leq & F_s(\bar{c}_s^{t+1}) + \Bigg(\|\nabla F_s(\bar{c}_s^{t+1}) - \nabla F_s(c_s^t)\| + \|\nabla F_s(c_s^t)\| \\ & + \frac{L_F}{2}\left(\|c_s^{t+1} - c_s^t\| + \|\bar{c}_s^{t+1} - c_s^t\|\right)\Bigg)\|c_s^{t+1} - \bar{c}_s^{t+1}\| \\ \leq & F_s(\bar{c}_s^{t+1}) + \Bigg(\|\nabla F_s(c_s^t)\| + L_F\|\bar{c}_s^{t+1} - c_s^t\| \\ & + \frac{L_F}{2}\left(\|c_s^{t+1} - c_s^t\| + \|\bar{c}_s^{t+1} - c_s^t\|\right)\Bigg)\|c_s^{t+1} - \bar{c}_s^{t+1}\| \\ = & F_s(\bar{c}_s^{t+1}) + \underbrace{\left(\|\nabla F_s(c_s^t)\| + \frac{L_F}{2}\|c_s^{t+1} - c_s^t\| + \frac{3L_F}{2}\|\bar{c}_s^{t+1} - c_s^t\|\right)\|c_s^{t+1} - \bar{c}_s^{t+1}\|}_{Q_s^t}\end{aligned} \tag{6.43}$$

Thus, we only need to bound $\mathbb{E}[Q_s^t]$

$$\begin{aligned}
\mathbb{E}_{Sel_s^t}[Q_s^t] =& \left(\|\nabla F_s(c_s^t)\| + \frac{3L_F}{2}\|\bar{c}_s^{t+1} - c_s^t\|\right)\mathbb{E}_{Sel_s^t}[\|c_s^{t+1} - \bar{c}_s^{t+1}\|] \\
&+ \frac{L_F}{2}\mathbb{E}_{Sel_s^t}[\|c_s^{t+1} - c_s^t\| \cdot \|c_s^{t+1} - \bar{c}_s^{t+1}\|] \\
\leq& \left(\|\nabla F_s(c_s^t)\| + 2L_F\|\bar{c}_s^{t+1} - c_s^t\|\right)\mathbb{E}_{Sel_s^t}[\|c_s^{t+1} - \bar{c}_s^{t+1}\|] \\
&+ \frac{L_F}{2}\mathbb{E}_{Sel_s^t}[\|c_s^{t+1} - \bar{c}_s^{t+1}\|^2] \\
\leq& \left(\|\nabla F_s(c_s^t)\| + 2L_F\|\bar{c}_s^{t+1} - c_s^t\|\right)\sqrt{\mathbb{E}_{Sel_s^t}[\|c_s^{t+1} - \bar{c}_s^{t+1}\|^2]} \\
&+ \frac{L_F}{2}\mathbb{E}_{Sel_s^t}[\|c_s^{t+1} - \bar{c}_s^{t+1}\|^2]
\end{aligned} \tag{6.44}$$

Note that $\mathbb{E}_{v_s^t}[w_k^{t+1}] = \bar{c}_s^{t+1}$, we have

$$\begin{aligned}
\mathbb{E}_{Sel_s^t}[\|c_s^{t+1} - \bar{c}_s^{t+1}\|^2] =& \frac{1}{K^2}\mathbb{E}_{Sel_{s,l}^t}\left[\left\|\sum_{l=1}^{K}\left(\hat{c}_{s,l}^{t+1} - \bar{c}_s^{t+1}\right)\right\|^2\right] \\
\leq& \frac{2}{K^2}\sum_{l=1}^{K}\mathbb{E}_{Sel_{s,l}^t}\left[\left\|\hat{c}_{s,l}^{t+1} - \bar{c}_s^{t+1}\right\|^2\right] = \frac{2}{K}\mathbb{E}_{v_{sk}^t}\left[\left\|w_k^{t+1} - \bar{c}_s^{t+1}\right\|^2\right] \\
=& \frac{2}{K}\mathbb{E}_{v_{sk}^t}\left[\left\|w_k^{t+1} - c_s^t\right\|^2 - 2\langle w_k^{t+1} - c_s^t, \bar{c}_s^{t+1} - c_s^t\rangle + \left\|\bar{c}_s^{t+1} - c_s^t\right\|^2\right] \\
=& \frac{2}{K}\mathbb{E}_{v_{sk}^t}\left[\left\|w_k^{t+1} - c_s^t\right\|^2 - \left\|\bar{c}_s^{t+1} - c_s^t\right\|^2\right] \leq \frac{2}{K}\mathbb{E}_{v_{sk}^t}\left[\left\|w_k^{t+1} - c_s^t\right\|^2\right]
\end{aligned} \tag{6.45}$$

Combining with (6.33), we can obtain

$$\begin{aligned}
\mathbb{E}_{Sel_s^t}[Q_s^t] \leq& \sqrt{\frac{2}{K}}\sqrt{\mathbb{E}_{v_{sk}^t}\left[\left\|w_k^{t+1} - c_s^t\right\|^2\right]}\|\nabla F_s(c_s^t)\| \\
&+ \left(2L_F\sqrt{\frac{2}{K}} + \frac{L_F}{K}\right)\mathbb{E}_{v_{sk}^t}\left[\left\|w_k^{t+1} - c_s^t\right\|^2\right] \\
\leq& \frac{1}{2}\sqrt{\frac{2}{K}}\left(\mu_\lambda\mathbb{E}_{v_{sk}^t}\left[\left\|w_k^{t+1} - c_s^t\right\|^2\right] + \frac{1}{\mu_\lambda}\|\nabla F_s(c_s^t)\|^2\right) \\
&+ \left(2L_F\sqrt{\frac{2}{K}} + \frac{L_F}{K}\right)\mathbb{E}_{v_{sk}^t}\left[\left\|w_k^{t+1} - c_s^t\right\|^2\right]
\end{aligned} \tag{6.46}$$

where

$$\begin{aligned}&\mathbb{E}\left[\mathbb{E}_{v_{sk}^t}[\|w_k^{t+1}-c_s^t\|^2]\right]\\ \leq&\mathbb{E}\left[\frac{1}{\sum_k u_{ks}^t n_k}\right]\left\{\left(\frac{\gamma+1}{\mu_\lambda}\right)^2\hat{m}_s\|\nabla F_s(c_s^t)\|^2+p'_\epsilon\bar{m}_s r^2\Delta^2\right\}\end{aligned}\tag{6.47}$$

hence,

$$\begin{aligned}\mathbb{E}[Q_s^t]\leq&\frac{1}{\mu_\lambda}\sqrt{\frac{1}{2K}}\left((\gamma+1)^2\hat{m}_s\mathbb{E}\left[\frac{1}{\sum_k u_{ks}^t n_k}\right]+1\right)\|\nabla F_s(c_s^t)\|^2\\ &+\underbrace{\left(2L_F\sqrt{\frac{2}{K}}+\frac{L_F}{K}+\frac{\mu_\lambda}{2}\sqrt{\frac{2}{K}}\right)}_{\leq\frac{4L_F+\mu_\lambda}{\sqrt{K}}}\mathbb{E}\left[\frac{1}{\sum_k u_{ks}^t n_k}\right]p'_\epsilon\bar{m}_s r^2\Delta^2\\ &+\underbrace{\left(2L_F\sqrt{\frac{2}{K}}+\frac{L_F}{K}\right)}_{\leq\frac{4L_F}{\sqrt{K}}}\mathbb{E}\left[\frac{1}{\sum_k u_{ks}^t n_k}\right]\left(\frac{\gamma+1}{\mu_\lambda}\right)^2\hat{m}_s\|\nabla F_s(c_s^t)\|^2\end{aligned}\tag{6.48}$$

Combining (6.40), (6.43), and (6.48) we have

$$\begin{aligned}&\mathbb{E}\left[F_s(c_s^{t+1})\right]\leq\mathbb{E}\left[F_s(\bar{c}_s^{t+1})\right]+\mathbb{E}[Q_s^t]\\ \leq&F_s(c_s^t)-\mathbb{E}\left[\frac{1}{\sum_k u_{ks}^t n_k}\right]\left(\rho_0-\frac{4L_F(\gamma+1)^2\hat{m}_s}{\mu_\lambda^2\sqrt{K}}-\frac{(\gamma+1)^2\hat{m}_s}{\mu_\lambda\sqrt{2K}}\right)\|\nabla F_s(c_s^t)\|^2\\ &+\frac{1}{\mu_\lambda\sqrt{2K}}\|\nabla F_s(c_s^t)\|^2\\ &+\mathbb{E}\left[\frac{1}{\sum_k(1-\sum_{s'\neq s}u_{ks}^t)n_k}\right]p'_\epsilon\left(R_0+\frac{(4L_F+\mu_\lambda)\bar{m}_s r^2}{\sqrt{K}}\right)\Delta^2\end{aligned}\tag{6.49}$$

Let λ be chosen such that $\rho_0-\frac{4L_F(\gamma+1)^2\hat{m}_s}{\mu_\lambda^2\sqrt{K}}-\frac{(\gamma+1)^2\hat{m}_s}{\mu_\lambda\sqrt{2K}}>0$, thus

$$\begin{aligned}\mathbb{E}\left[F_s(c_s^{t+1})\right] \leq & F_s(c_s^t) + \frac{p'_\epsilon\left(R_0 + \frac{(4L_F+\mu_\lambda)\bar{m}_s r^2}{\sqrt{K}}\right)\Delta^2}{(1-p_\epsilon)n_s - p'_\epsilon(S-2)n} \\ & - \frac{\left(\rho_0 - \frac{4L_F(\gamma+1)^2\hat{m}_s}{\mu_\lambda^2\sqrt{K}} - \frac{(\gamma+1)^2\hat{m}_s+(1-p_\epsilon)n_s+p'_\epsilon n}{\mu_\lambda\sqrt{2K}}\right)}{(1-p_\epsilon)n_s + p'_\epsilon n}\|\nabla F_s(c_s^t)\|^2\end{aligned} \tag{6.50}$$

□

6.7.5 Proof of Theorem 6.5

Note that under Assumption 6.3, the sum of proximal objectives $h(w_1 \cdots w_N, c_1 \cdots c_S) = \sum_{k=1}^N h_k(w_k; c, \tilde{u}_k)$ is jointly convex on $(w_1 \cdots w_N, c_1 \cdots c_S)$ for fixed $\tilde{u}_k$, and the training process of **FedSoft** can be regarded as a cyclic block coordinate descent algorithm that sequentially updates $w_1 \cdots w_N, c_1 \cdots c_S$ while other blocks are fixed. This type of algorithm is know to converge at least linearly to a stationary point [57].

To see why the averaging of centers correspond to the minimization of them, simply set the gradients to zero

$$\nabla_{c_s} h = \sum_{k=1}^N \tilde{u}_{ks}\left(c_s - w_k\right) = 0 \tag{6.51}$$

This implies the optimal c_s^* equals

$$c_s^* = \sum_k \frac{\tilde{u}_{ks} w_k}{\sum_k \tilde{u}_{ks}} \tag{6.52}$$

which is exactly the updating rule of **FedSoft**.

Chapter 7

Conclusion and Future Directions

7.1 Conclusion

The objective of this thesis is to propose efficient frameworks and algorithms to enable the collaborative use of data for mobile users in the presence of real-world physical limitations and privacy protection requirements. Based on the privacy characteristics of user data, we consider two types of mobile applications, and we propose various approaches for each of them.

Chapters 2 and 3 focus on the first type of applications, where the data is assumed to be publicly available or privacy insensitive and thus can be freely shared by all users. Specifically, Chapter 2 investigates the possibility for users to co-use public data through mobile caching, and we reveal the potential economic value of this innovative caching system under the competition from existing caching tools. In Chapter 3, we propose that user devices offload their training data in a device-to-device manner to accelerate the training of machine learning models. We design an offloading algorithm that optimally adapts to the network typologies.

Chapters 4 to 6 on the other hand concentrate on enhancing the cutting-edge federated learning framework so as to facilitate the collaborative use of privacy-sensitive user data. In Chapter 4, we introduce the client recruitment problem, and show how our proposed recruitment strategy can prominently improve the accuracy and efficiency of federated learning, leading to high quality training results within reasonable time and budget limitations. In Chapter 5, we further consider how to utilize updates from the recruited clients with more flexibility. We relax the traditional requirements for dedicated participation on federated learning devices, and design new algorithms to incorporate more flexible device participation patterns including incompleteness, inactivity, early departures and late arrivals. In Chapter 6, we discuss how to better leverage similarities between data at different clients. We propose soft clustered federated learning, which allows each client to sample data from multiple data distributions. We design efficient algorithms to relieve the training burden of user devices, parallelize the training of cluster models and local personalization, and enable different cluster models to exploit their similarities.

7.2 Future Directions

The theories and technologies revealed in this thesis may also facilitate future investigation of other research directions.

Specifically, one may further *combine the caching technology and federated learning* to tackle the physical constraint. In Chapter 2 we have discussed the sharing of public data through mobile caching. Such a caching system may well be utilized in federated learning for the storage of intermediate computation results (e.g. global models). Furthermore, the caching technology can be naturally integrated into the hierarchical federated learning system [1], which entails fine-grained modeling of the network heterogeneity by partitioning the training process into multiple tiers. It will be interesting to see how the placement of caching nodes interacts with this edge architecture, and whether caching can help reduce the resource consumption of federated learning. Likewise, *combining flexible federated learning with* ***FedSoft*** (Chapters 5 and 6) can be an interesting direction to further improve the efficiency of federated learning when the number of data distributions grows large.

The encrypted data exchange and machine learning technology can also be utilized to tackle the privacy constraint. Cutting-edge cryptography tools provide strong privacy guarantees for the exchange of user information. For example, protocols built on top of the secure multi-party computation preemptive have been widely adopted for generic data exchange tasks [114]. However, most secured communication algorithms are resource consuming, involving multiple encryption and decryption processes. Incorporating these tools in the resource restricted mobile environment is an interesting challenge. Finally, directly purchasing the ownership of data from common users remains the most straightforward way of data co-use. *The pricing of user data* therefore helps solve the privacy constraint with economic tools. The price of user data reflects both the data suppliers' value of their privacy, and the consumers' ability to gain profits from the data, e.g., through advanced data engineering technology. Our client recruitment approach as depicted in Chapter 4 can serve as a starting point for data pricing in the federated learning scenario. Recent works on the design of incentive mechanisms to encourage the user participation in federated learning [106, 108] may also be helpful.

Bibliography

[1] Mehdi Salehi Heydar Abad, Emre Ozfatura, Deniz Gunduz, and Ozgur Ercetin. Hierarchical federated learning across heterogeneous cellular networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8866–8870. IEEE, 2020. Cited on page 156.

[2] Lada A Adamic and Bernardo A Huberman. Zipf's law and the internet. *Glottometrics*, 3(1):143–150, 2002. Cited on pages 2 and 17.

[3] Jeffrey G Andrews, Abhishek K Gupta, and Harpreet S Dhillon. A primer on cellular network analysis using stochastic geometry. *arXiv preprint arXiv:1604.03183*, 2016. Cited on page 16.

[4] Shehzad A Ashraf, Ismet Aktas, Erik Eriksson, Ke Wang Helmersson, and Junaid Ansari. Ultra-reliable and low-latency communication for wireless factory automation: From LTE to 5G. In *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2016. Cited on page 37.

[5] Benjamin Baron, Prométhée Spathis, Hervé Rivano, Marcelo Dias de Amorim, Yannis Viniotis, and Mostafa Ammar. Centrally-controlled mass data offloading using vehicular traffic. *IEEE Transactions on Network and Service Management*, 14(2):401–415, 2017. Cited on page 15.

[6] Ejder Bastug, Mehdi Bennis, and Mérouane Debbah. Living on the edge: The role of proactive caching in 5g wireless networks. *IEEE Communications Magazine*, 52(8):82–89, 2014. Cited on page 12.

[7] Ejder Bastug, Mehdi Bennis, Marios Kountouris, and Mérouane Debbah. Cache-enabled small cell networks: Modeling and tradeoffs. *EURASIP Journal on*

Wireless Communications and Networking, 2015(1):41, 2015. Cited on pages 16 and 23.

[8] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019. Cited on page 85.

[9] Julie Bort. The next multibillion-dollar tech market was quietly born this year, says a-list vc peter levine. *Business Insider. December*, 16:2016, 2016. Cited on page 5.

[10] Christopher Briggs, Zhong Fan, and Peter Andras. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2020. Cited on page 129.

[11] Xuejun Cai, Shunliang Zhang, and Yunfei Zhang. Economic analysis of cache location in mobile network. In *Proc. of IEEE WCNC*, pages 1243–1248. IEEE, 2013. Cited on page 15.

[12] Sebastian Caldas, Jakub Konečny, H Brendan McMahan, and Ameet Talwalkar. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210*, 2018. Cited on page 83.

[13] Ta-Cheng Chang, Liang Zheng, Maria Gorlatova, Chege Gitau, Ching-Yao Huang, and Mung Chiang. Decomposing data analytics in fog networks. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, pages 1–2, 2017. Cited on page 39.

[14] Dimitris Chatzopoulos, Carlos Bermejo, Zhanpeng Huang, and Pan Hui. Mobile augmented reality survey: From where we are to where we go. *IEEE Access*, 5:6917–6950, 2017. Cited on pages 37 and 49.

[15] Mingzhe Chen, Zhaohui Yang, Walid Saad, Changchuan Yin, H Vincent Poor, and Shuguang Cui. A joint learning and communications framework for federated learning over wireless networks. *arXiv preprint arXiv:1909.07972*, 2019. Cited on page 7.

[16] Yujing Chen, Yue Ning, and Huzefa Rangwala. Asynchronous online federated learning for edge devices. *arXiv preprint arXiv:1911.02134*, 2019. Cited on

page 85.

[17] Zheng Chen and Marios Kountouris. D2d caching vs. small cell caching: Where to cache content in a wireless network? In *Proc. of the IEEE Workshop on Signal Processing Advances in Wireless Communications*, pages 1–6. IEEE, 2016. Cited on page 29.

[18] Xu Cheng, Cameron Dale, and Jiangchuan Liu. Statistics and social network of youtube videos. In *2008 16th Interntional Workshop on Quality of Service*, pages 229–238. IEEE, 2008. Cited on page 2.

[19] Mung Chiang. *Networked Life: 20 Questions and Answers.* Cambridge University Press, 2012. Cited on page 57.

[20] Yae Jee Cho, Jianyu Wang, and Gauri Joshi. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. *arXiv preprint arXiv:2010.01243*, 2020. Cited on page 61.

[21] Cisco Systems. Demystifying 5G in industrial IOT. White Paper, 2019. Cited on pages 37 and 49.

[22] City and County of San Francisco. Sf311 cases. https://sf311.org/, 2018. [Online; accessed 20-December-2018]. Cited on page 28.

[23] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017. Cited on page 97.

[24] Pierre Coucheney, Patrick Maillé, and Bruno Tuffin. Impact of competition between isps on the net neutrality debate. *IEEE Transactions on Network and Service Management*, 10(4):425–433, 2013. Cited on page 23.

[25] Ying Cui and Dongdong Jiang. Analysis and optimization of caching and multicasting in large-scale cache-enabled heterogeneous wireless networks. *IEEE transactions on Wireless Communications*, 16(1):250–264, 2017. Cited on page 14.

[26] Georgios Damaskinos, Rachid Guerraoui, Anne-Marie Kermarrec, Vlad Nitu, Rhicheek Patra, and Francois Taiani. Fleet: Online federated learning via staleness awareness and performance prediction. *arXiv preprint arXiv:2006.07273*,

2020. Cited on page 85.

[27] Ashutosh Dhekne, Mahanth Gowda, Romit Roy Choudhury, and Srihari Nelakuditi. If wifi aps could move: A measurement study. *IEEE Transactions on Mobile Computing*, 17(10):2293–2306, 2018. Cited on page 12.

[28] Marco Di Renzo, Alessandro Guidotti, and Giovanni E Corazza. Average rate of downlink heterogeneous cellular networks over generalized fading channels: A stochastic geometry approach. *IEEE Transactions on Communications*, 61(7):3050–3071, 2013. Cited on page 28.

[29] Canh T Dinh, Nguyen H Tran, and Tuan Dung Nguyen. Personalized federated learning with moreau envelopes. *arXiv preprint arXiv:2006.08848*, 2020. Cited on page 130.

[30] Moming Duan, Duo Liu, Xinyuan Ji, Renping Liu, Liang Liang, Xianzhang Chen, and Yujuan Tan. Fedgroup: Ternary cosine similarity-based clustered federated learning framework toward high accuracy in heterogeneous data. *arXiv preprint arXiv:2010.06870*, 2020. Cited on page 129.

[31] Sanghamitra Dutta, Gauri Joshi, Soumyadip Ghosh, Parijat Dube, and Priya Nagpurkar. Slow and stale gradients can win the race: error-runtime trade-offs in distributed SGD. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 803–812, 2018. Cited on page 39.

[32] Farshid Farhat, Diman Zad Tootaghaj, Yuxiong He, Anand Sivasubramaniam, Mahmut Kandemir, and Chita R Das. Stochastic modeling and optimization of stragglers. *IEEE Transactions on Cloud Computing*, 6(4):1164–1177, 2016. Cited on page 46.

[33] Lauren Frayer. Free wi-fi on buses offers a link to future of 'smart cities'. NPR All Tech Considered, 2015. Cited on page 13.

[34] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *arXiv preprint arXiv:2006.04088*, 2020. Cited on pages 130, 132, 135, 136, 138, and 140.

[35] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010. Cited on page 140.

[36] Neel Guha, Ameet Talwalkar, and Virginia Smith. One-shot federated learning. *arXiv preprint arXiv:1902.11175*, 2019. Cited on page 140.

[37] Kenza Hamidouche, Walid Saad, and Mérouane Debbah. Breaking the economic barrier of caching in cellular networks: Incentives and contracts. In *Global Communications Conference (GLOBECOM), 2016 IEEE*, pages 1–6. IEEE, 2016. Cited on page 14.

[38] Pengchao Han, Shiqiang Wang, and Kin K Leung. Adaptive gradient sparsification for efficient federated learning: An online learning approach. *arXiv preprint arXiv:2001.04756*, 2020. Cited on page 85.

[39] Binbin Hu, Luoyang Fang, Xiang Cheng, and Liuqing Yang. In-vehicle caching (iv-cache) via dynamic distributed storage relay (d ^{2} sr) in vehicular networks. *IEEE Transactions on Vehicular Technology*, 68(1):843–855, 2018. Cited on page 15.

[40] Chuang Hu, Wei Bao, Dan Wang, and Fengming Liu. Dynamic adaptive DNN surgery for inference acceleration on the edge. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 1423–1431, 2019. Cited on page 39.

[41] Zhiwen Hu, Zijie Zheng, Tao Wang, Lingyang Song, and Xiaoming Li. Roadside unit caching: Auction-based storage allocation for multiple content providers. *IEEE Transactions on Wireless Communications*, 16(10):6321–6334, 2017. Cited on page 14.

[42] Li Huang, Andrew L Shea, Huining Qian, Aditya Masurkar, Hao Deng, and Dianbo Liu. Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records. *Journal of biomedical informatics*, 99:103291, 2019. Cited on page 130.

[43] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on non-iid data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7865–7873, 2021. Cited on page 130.

[44] Mingyue Ji, Giuseppe Caire, and Andreas F Molisch. Wireless device-to-device caching networks: Basic principles and system performance. *arXiv preprint*

arXiv:1305.5216, 2013. Cited on page 14.

[45] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019. Cited on pages 6 and 127.

[46] Jakub Konečný, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016. Cited on pages 39 and 83.

[47] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. Cited on page 52.

[48] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. Cited on page 97.

[49] Yann LeCun, Corinna Cortes, and Christopher J. C. Burges. The MNIST database of handwritten digits. Cited on page 52.

[50] Chao Li, Daniel Yang Li, Gerome Miklau, and Dan Suciu. A theory of pricing private data. *Communications of the ACM*, 60(12):79–86, 2017. Cited on pages 60 and 71.

[51] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, page 106854, 2020. Cited on page 6.

[52] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018. Cited on pages 7, 97, 128, 131, 135, and 137.

[53] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019. Cited on pages 7, 82, 84, 86, 88, and 89.

[54] Yicheng Lin, Wei Bao, Wei Yu, and Ben Liang. Optimizing user association

and spectrum allocation in hetnets: A utility perspective. *IEEE Journal on Selected Areas in Communications*, 33(6):1025–1039, 2015. Cited on page 18.

[55] Lumin Liu, Jun Zhang, S. H. Song, and Khaled B. Letaief. Client-edge-cloud hierarchical federated learning. `https://arxiv.org/abs/1905.06641`. Cited on page 85.

[56] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30:6467–6476, 2017. Cited on page 140.

[57] Zhi-Quan Luo and Paul Tseng. On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1):7–35, 1992. Cited on page 154.

[58] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020. Cited on pages 126 and 130.

[59] Othmane Marfoq, Giovanni Neglia, Aurélien Bellet, Laetitia Kameni, and Richard Vidal. Federated multi-task learning under a mixture of distributions. *Advances in Neural Information Processing Systems*, 34, 2021. Cited on pages 127, 128, 130, 132, and 140.

[60] Silvano Martello and Paolo Toth. Knapsack problems: algorithms and computer implementations. *Wiley-Interscience series in discrete math and optimization*, 1990. Cited on page 73.

[61] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017. Cited on pages 7, 36, 39, 41, 42, 60, 76, 82, 86, 97, 128, and 134.

[62] Matthew J Menne, Imke Durre, Russell S Vose, Byron E Gleason, and Tamara G Houston. An overview of the global historical climatology network-daily database. *Journal of atmospheric and oceanic technology*, 29(7):897–910, 2012. Cited on page 79.

[63] Byoung-Yoon Min, Wonkwang Shin, and Dong Ku Kim. Delay-tolerable contents offloading via vehicular caching overlaid with cellular networks. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and*

Computer Sciences, 100(1):283–293, 2017. Cited on pages 12 and 15.

[64] Kevin P Murphy. *Machine learning: A probabilistic perspective*. MIT Press, 2012. Cited on page 41.

[65] Syed Ahsan Raza Naqvi, Syed Ali Hassan, Haris Pervaiz, and Qiang Ni. Drone-aided communication as a key enabler for 5g and resilient public safety networks. *IEEE Communications Magazine*, 56(1):36–42, 2018. Cited on pages 12 and 13.

[66] Giovanni Neglia, Gianmarco Calbi, Don Towsley, and Gayane Vardoyan. The role of network topology for distributed machine learning. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 2350–2358, 2019. Cited on pages 46 and 49.

[67] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2019. Cited on pages 60, 61, and 85.

[68] Konstantinos Poularakis, George Iosifidis, Leandros Tassiulas, et al. Approximation algorithms for mobile data caching in small cell networks. *IEEE Trans. Communications*, 62(10):3665–3677, 2014. Cited on page 14.

[69] Shi Pu, Wei Shi, Jinming Xu, and Angelia Nedić. A push-pull gradient method for distributed optimization in networks. In *IEEE Conference on Decision and Control (CDC)*, pages 3385–3390, 2018. Cited on page 36.

[70] Adnan Qayyum, Kashif Ahmad, Muhammad Ahtazaz Ahsan, Ala Al-Fuqaha, and Junaid Qadir. Collaborative federated learning for healthcare: Multi-modal covid-19 diagnosis at the edge. *arXiv preprint arXiv:2101.07511*, 2021. Cited on page 130.

[71] Xukan Ran, Haolianz Chen, Xiaodan Zhu, Zhenming Liu, and Jiasi Chen. Deepdecision: A mobile deep learning framework for edge video analytics. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 1421–1429. IEEE, 2018. Cited on page 39.

[72] BLS Prakasa Rao. The rate of convergence of the least squares estimator in a non-linear regression model with dependent errors. *Journal of multivariate analysis*, 14(3):315–322, 1984. Cited on page 67.

[73] Krishna Rao. The path to 5G for health care. IEEE Perspectives on 5G Applications and Services. Cited on page 49.

[74] Francesco Restuccia, Sajal K Das, and Jamie Payton. Incentive mechanisms for participatory sensing: Survey and research challenges. *ACM Transactions on Sensor Networks (TOSN)*, 12(2):13, 2016. Cited on page 13.

[75] Douglas A Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, 741:659–663, 2009. Cited on page 127.

[76] Elsa Rizk, Stefan Vlaski, and Ali H Sayed. Dynamic federated learning. *arXiv preprint arXiv:2002.08782*, 2020. Cited on page 85.

[77] Yichen Ruan and Carlee Joe-Wong. On the economic value of mobile caching. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 984–993. IEEE, 2020. Cited on page 8.

[78] Yichen Ruan and Carlee Joe-Wong. Fedsoft: Soft clustered federated learning with proximal local updating. *arXiv preprint arXiv:2112.06053*, 2021. Cited on page 10.

[79] Yichen Ruan, Xiaoxi Zhang, and Carlee Joe-Wong. How valuable is your data? optimizing client recruitment in federated learning. In *2021 19th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*, pages 1–8. IEEE, 2021. Cited on page 9.

[80] Yichen Ruan, Xiaoxi Zhang, Shu-Che Liang, and Carlee Joe-Wong. Towards flexible device participation in federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3403–3411. PMLR, 2021. Cited on page 10.

[81] Yichen Ruan, Liang Zheng, Maria Gorlatova, Mung Chiang, and Carlee Joe-Wong. Pricing tradeoffs for data analytics in fog–cloud scenarios. *Fog and Fogonomics: Challenges and Practices of Fog Computing, Communication, Networking, Strategy, and Economics*, pages 83–106, 2020. Cited on page 5.

[82] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems*, 2020. Cited on page 129.

[83] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 2019. Cited on page 85.

[84] Mahadev Satyanarayanan. A brief history of cloud offload: A personal journey from odyssey through cyber foraging to cloudlets. *GetMobile: Mobile Computing and Communications*, 18(4):19–23, 2015. Cited on page 5.

[85] Kaiming Shen and Wei Yu. Fractional programming for communication systems—part i: Power control and beamforming. *IEEE Transactions on Signal Processing*, 66(10):2616–2630, 2018. Cited on pages 18, 25, and 26.

[86] Wonkwang Shin, Byoung-Yoon Min, and Dong Ku Kim. Vehicaching: Embracing user request on vehicle route with proactive data transportation. In *Vehicular Technology Conference (VTC Spring), 2015 IEEE 81st*, pages 1–5. IEEE, 2015. Cited on pages 12 and 15.

[87] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *ACM Conference on Computer and Communications Security (SIGSAC)*, pages 1310–1321, 2015. Cited on page 39.

[88] Khe Chai Sim, Petr Zadrazil, and Françoise Beaufays. An investigation into on-device personalization of end-to-end automatic speech recognition models. *arXiv preprint arXiv:1909.06678*, 2019. Cited on page 126.

[89] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4424–4434, 2017. Cited on pages 7, 8, 59, 126, and 127.

[90] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Distributed deep neural networks over the cloud, the edge and end devices. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 328–339, 2017. Cited on page 39.

[91] Intelligent Small Cell Trial. Rethinking the small cell business model. https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/communications-small-cell-study.pdf, 2011. Cited on page 12.

[92] Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*,

109(3):475–494, 2001. Cited on page 27.

[93] Yuwei Tu, Yichen Ruan, Su Wang, Satyavrat Wagle, Christopher G Brinton, and Carlee Joe-Wang. Network-aware optimization of distributed learning for fog computing. *arXiv preprint arXiv:2004.08488*, 2020. Cited on page 9.

[94] Tiffany Tuor, Shiqiang Wang, Bong Jun Ko, Changchang Liu, and Kin K Leung. Data selection for federated learning with relevant and irrelevant data at clients. *arXiv preprint arXiv:2001.08300*, 2020. Cited on page 64.

[95] Luigi Vigneri. *Vehicles as a mobile cloud: modelling, optimization and performance analysis*. PhD thesis, Université Côte d'Azur, 2017. Cited on pages 12, 15, and 17.

[96] Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10:3152676, 2017. Cited on page 3.

[97] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *arXiv preprint arXiv:2007.07481*, 2020. Cited on page 85.

[98] Rui Wang, Xi Peng, Jun Zhang, and Khaled B Letaief. Mobility-aware caching for content-centric wireless networks: Modeling and methodology. *IEEE Communications Magazine*, 54(8):77–83, 2016. Cited on pages 14 and 16.

[99] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, and Kevin Chan. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications*, 37(6):1205–1221, 2019. Cited on pages 7, 40, 41, 43, 45, 53, 63, and 85.

[100] Felix Ming Fai Wong, Zhenming Liu, Mung Chiang, Felix Ming Fai Wong, Zhenming Liu, and Mung Chiang. On the efficiency of social recommender networks. *IEEE/ACM Transactions on Networking*, 24(4):2512–2524, 2016. Cited on page 47.

[101] Ming Xie, Guodong Long, Tao Shen, Tianyi Zhou, Xianzhi Wang, Jing Jiang, and Chengqi Zhang. Multi-center federated learning. *arXiv preprint arXiv:2108.08647*, 2021. Cited on page 129.

[102] Chengxu Yang, QiPeng Wang, Mengwei Xu, Shangguang Wang, Kaigui Bian, and Xuanzhe Liu. Heterogeneity-aware federated learning. *arXiv preprint arXiv:2006.06983*, 2020. Cited on page 85.

[103] Howard H Yang, Zuozhu Liu, Tony QS Quek, and H Vincent Poor. Scheduling policies for federated learning in wireless networks. *IEEE transactions on communications*, 68(1):317–333, 2019. Cited on page 61.

[104] Tianbao Yang. Trading computation for communication: Distributed stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 629–637, 2013. Cited on page 40.

[105] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903*, 2018. Cited on pages 7 and 60.

[106] Han Yu, Zelei Liu, Yang Liu, Tianjian Chen, Mingshu Cong, Xi Weng, Dusit Niyato, and Qiang Yang. A fairness-aware incentive scheme for federated learning. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 393–399, 2020. Cited on page 157.

[107] Sui Yutao, Jaakko Vihriala, Agisilaos Papadogiannis, Mikael Sternad, Wei Yang, and Tommy Svensson. Moving cells: a promising solution to boost performance for vehicular users. *IEEE Communications Magazine*, 51(6):62–68, 2013. Cited on page 15.

[108] Yufeng Zhan, Peng Li, Zhihao Qu, Deze Zeng, and Song Guo. A learning-based incentive mechanism for federated learning. *IEEE Internet of Things Journal*, 7(7):6360–6368, 2020. Cited on page 157.

[109] Ke Zhang, Supeng Leng, Yejun He, Sabita Maharjan, and Yan Zhang. Cooperative content caching in 5g networks with mobile edge computing. *IEEE Wireless Communications*, 25(3):80–87, 2018. Cited on pages 12, 13, and 15.

[110] Xiaoxi Zhang, Jianyu Wang, Gauri Joshi, and Carlee Joe-Wong. Machine learning on volatile instances. *arXiv preprint arXiv:2003.05649*, 2020. Cited on page 83.

[111] Yang Zhang, Jing Zhao, and Guohong Cao. Roadcast: a popularity aware content sharing scheme in vanets. *ACM SIGMOBILE Mobile Computing and*

Communications Review, 13(4):1–14, 2010. Cited on page 15.

[112] Yao Zhang, Changle Li, Tom Hao Luan, Yuchuan Fu, Weisong Shi, and Lina Zhu. A mobility-aware vehicular caching scheme in content centric networks: Model and optimization. *IEEE Transactions on Vehicular Technology*, 68(4):3100–3112, 2019. Cited on page 15.

[113] Yuchen Zhang, John Duchi, Michael I Jordan, and Martin J Wainwright. Information-theoretic lower bounds for distributed statistical estimation with communication constraints. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2328–2336, 2013. Cited on page 40.

[114] Chuan Zhao, Shengnan Zhao, Minghao Zhao, Zhenxiang Chen, Chong-Zhi Gao, Hongwei Li, and Yu-an Tan. Secure multi-party computation: theory, practice and applications. *Information Sciences*, 476:357–372, 2019. Cited on page 156.

[115] Kaichuan Zhao, Shan Zhang, Ning Zhang, Yuezhi Zhou, Yaoxue Zhang, and Xuemin Shen. Incentive mechanism for cached-enabled small cell sharing: A stackelberg game approach. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–6. IEEE, 2017. Cited on page 14.

[116] Xinyang Zhou and Lijun Chen. Demand shaping in cellular networks. *IEEE Transactions on Control of Network Systems*, 2018. Cited on page 12.