

Federated Learning as a Mean-Field Game

Arash Mehrjou
amehrjou@inf.ethz.ch

ETH Zürich
&
Max Planck Institute for Intelligent Systems

Abstract

We establish a connection between federated learning, a concept from machine learning, and mean-field games, a concept from game theory and control theory. In this analogy, the local federated learners are considered as the players and the aggregation of the gradients in a central server is the mean-field effect. We present federated learning as a differential game and discuss the properties of the equilibrium of this game. We hope this novel view to federated learning brings together researchers from these two distinct areas to work on fundamental problems of large-scale distributed and privacy-preserving learning algorithms.

1 Introduction

In the following sections, we briefly review the necessary material from federated learning and mean-field games that will be used later when similarities start to appear. In federated learning (Section 1.1), we present the general idea and one of its standard algorithm called Federated Averaging (FedAvg). The general setting of stochastic games is introduced in Section 2. Two main pillars of mean-field games that are stochastic optimal control and stochastic differential games are introduced in Sections 2.1 and 2.2 respectively. The introduction of mean-field games is completed in Section 2.3. Finally, the connection between federated learning and mean-field games is established in Section 3.

1.1 Federated Learning

The concept of federated learning refers to a learning scenario where not all data is available to a single learner, instead is distributed among many devices that can be location-wise distant. Each device together with the fraction of data available to it is called a *client*. We call the set of all clients participating in this distributed learning process, a *federation*. Each client learns a local model using its local dataset and contributes to a centralized model called the *server* at each *round* of learning. The centralized model is then used to update the client models and the loop continues for the subsequent rounds until some measure of performance is met.

We are particularly interested in the setting where a large number of clients are participating in the federation. For example, there are hundreds of millions of active mobile phones in the world each of which has its own local dataset from a particular modality (textual, auditory, visual, etc) generated by its user. One of the first commercial applications of federated learning was to use the enormous wealth of textual data generated by users of smartphones to develop a language model to predict the next words while typing. The amount of data generated by each user is not sufficient to develop a satisfactory local model. Hence, it would be useful to use the data provided by millions of other users to learn a better model while preserving the users' privacy by not sending their data to a distant server.

The key constraint in federated learning is that the clients can only communicate via updating a centralized model and then getting updated by it. This model of communication is enforced due to privacy constraints or technical limitations such as limited bandwidth. In the mobile phone example, the data generated by each user, most likely, comes from a distinct distribution depending on the age, location, education, and other attributes of the user. Moreover, the communication is costly as it is normally carried out via the cellphones' internet connection.

Morally speaking, the objective of federated learning is to use locally trained models to find a single centralized model that is as good as the model that could have been trained if all datasets were available

in one place to a single learner. Here, we formalize this objective mathematically as it will be needed in the coming sections.

The concept of federated learning can be studied separately from the learning algorithm used by the clients. For convenience, we present the main ideas for a multi-class classification problem where a function is sought to map members of the input space $\mathcal{X}$ to the output space $\mathcal{Y}$ consisting of a finite number of values, e.g., $|\mathcal{Y}|=c$. Let $\Delta_{\mathcal{Y}}$ be the simplex over $\mathcal{Y}$ i.e., $\sum_i o_i = 1$ for $o = [o_1, o_2, \dots, o_c]$. The learning algorithm searches in the hypothesis family $\mathcal{H}$ consisting of hypothesis functions $h \in \mathcal{H} : \mathcal{X} \rightarrow \Delta_{\mathcal{Y}}$ based on a specified loss function where $h(x)$ is the probability distribution over $[1, 2, \dots, c]$ determining the probability that the input instance x belongs to every class of $\mathcal{Y}$. For every hypothesis h and the labeled sample (x, y) , the loss function $\ell : \mathcal{H} \times \Delta_{\mathcal{Y}} \times \mathcal{Y} \rightarrow \mathbb{R}_0^+$ gives a non-negative values $\ell(h(x), y)$. Every *task* is a distribution on $\mathcal{X} \times \mathcal{Y}$ denoted by $\mathcal{D}$. The *risk* (expected loss) of the hypothesis h on task $\mathcal{D}$ is denoted by $\mathcal{L}_{\mathcal{D}}(h)$ where

$$\mathcal{L}_{\mathcal{D}}(h) = \mathbf{E}_{(x,y) \sim \mathcal{D}} [\ell(h(x), y)] \quad (1)$$

and $h_{\mathcal{D}} := \operatorname{argmin}_{h \in \mathcal{H}} \mathcal{L}_{\mathcal{D}}(h)$ is the hypothesis with minimum risk.

Consider a federation of p learners (known as clients in FL literature) indexed by $k \in \{1, 2, \dots, p\}$. The k -th learner has the data distribution $\mathcal{D}_k$ where m_k samples of it $S_k = \{(x_{i,k}, y_{i,k}), \dots, (x_{m_k,k}, y_{m_k,k})\}$ is available for learning. Hence, total number of samples available to all learners is $m = m_1 + \dots + m_p$. Let the hatted notation $\hat{\mathcal{D}}_k$ be the uniform distribution on the samples S_k available to client k , known as the empirical distribution. The objective of federated learning is to learn from the p sample sets $\{S_1, \dots, S_p\}$ a hypothesis h that performs well on some *target* distribution denoted by $\mathcal{D}^t$. The problem falls into different categories depending on the chosen target distribution.

The target distribution is often constructed by a mixture of the client's distributions as

$$\bar{\mathcal{U}} = \sum_{k=1}^p \frac{m_k}{m} \mathcal{D}_k, \quad (2)$$

where the weight of each client is proportional to its share of the total samples. As a result, the empirical target distribution becomes $\hat{\mathcal{U}} = \sum_{k=1}^p \frac{m_k}{m} S_k$ based on which the actual evaluation is carried out. The server sends the central model to the clients and they use their local data to update their copy of the model. The updated models are then sent to the server and aggregated to build a new central model. This two-phase process repeats until a stopping criterion is met (see Algorithm 1).

Based on the way client's updates are aggregated, two branches can be thought of. In the first branch, each client produces its update as the gradient $g^k = m_k^{-1} \sum_{x \in S_k} \nabla_w \ell(h(x; w), y)$ computed on its local data. In a non-stochastic setting, every client uses its whole local data (deterministic gradient descent) to compute its local gradients. If the learning rate is shared across all clients, the central model is updated by averaging the local gradients as $w_{r+1} \leftarrow w_r - \eta \sum_{k=1}^p \frac{m_k}{m} g^k$. The update strategy of the well-known method **FedSGD**[1] is in principle similar to this gradient aggregation rule.

An alternative branch is to apply the gradients locally and then update the central model by averaging over all clients' models. This aggregation method gives more flexibility as the clients can train their local models for multiple epochs and with different learning rates. The local models are updated by $w_{r+1}^k \leftarrow w_r^k - \eta_k m_k^{-1} \sum_{x \in \mathcal{B}} \nabla_w \ell(h(x); w)$ for as many epochs as needed. Once all clients update their models in parallel, the central model is updated by $w_{r+1} \leftarrow \sum_{k=1}^p \frac{m_k}{m} w_r^k$. This method is called **FedAvg** [1] and its pseudo-code is provided in Algorithm 1. Theoretically speaking, the performance of the output model of **FedAvg** can be poor when the data distributions of the clients are too different. Intuitively, different distributions can be seen as different tasks and averaging over the weights of the models that solve different tasks can result in a model whose performance is worse in either of them. This effect has been shown by averaging the weights of two digit-recognition networks that are trained on distinct subsets of MNIST long enough that the models begin to overfit to their local data sets [2].

Both above-mentioned learning methods weigh the client's contribution to the updated central model by m_k/m . This is a reasonable weighting choice because, as suggested by the target task in Equation (2), the learned model has to perform well on the mixture of clients' distributions where each client's share in the evaluation dataset is proportional to the size of its local dataset.

Algorithm 1 Federated learning by model averaging (FedAvg)

Input: $\{S_1, \dots, S_p\}$: Local datasets, $C \in [0, 1]$: Fraction of clients chosen for each FL update round, E : The number of epochs for training each local model, **batchsize** : The size of the mini-batches, η : learning rate, ℓ : loss function, $\mathcal{H} = \{h(\cdot; w)\}$: The hypothesis family, parameterised vector w .

Server side:

```
1:  $r(\text{round index}) \leftarrow 0$ 
2: while Stopping criterion not met do
3:    $m \leftarrow \min(1, Cp)$ 
4:    $M_r \leftarrow$  A random set of  $n$  clients chosen at round  $r$ 
5:   for each client  $k \in M_r$  in parallel do
6:      $w_{r+1}^k \leftarrow \text{ClientUpdate}(k, w_r)$ 
7:    $w_{r+1} \leftarrow 1/m \sum_{k=1}^n w_r^k$ 
8:   end for
9: end while
```

ClientUpdate(k, w) :

```
1:  $B \leftarrow$  Break  $S_k$  into batches of size batchsize
2: for each epoch  $i : 1$  to  $E$  do
3:   for each batch  $b \in B$  do
4:      $w \leftarrow w - \frac{\eta}{|b|} \sum_{(x,y) \in b} \nabla \ell(h(x; w), y)$ 
5:   end for
6: end for
7: return  $w$ 
```

2 Stochastic Games

The mean-field theory introduced by Lasry-Lions [3] has attracted a great deal of attention in recent years. The framework is concerned with a system of a large number of rational agents that try to achieve their own goals in the presence of the effect of other agents. In mean-field games (MFG), the agents are all assumed similar in the sense that one agent can be taken as a representative agent of the entire population.

The difference between MFG and standard optimal control problem is that both the evolution of the states and the decision making of the representative agent is influenced by the whole community of other agents. Inspired by field theory in statistical physics, the effect of the community of agents can be modeled by a mean-field term. Hence the mean-field game boils down to an optimal control problem and a stochastic evolution problem. As a result, the formulation of an MFG can be described as a coupled system of partial differential equations (PDEs). One PDE is the Hamilton-Jacobi-Bellman (HJB) equation that models the evolution for the optimal control part and the other PDE is the Fokker-Plank (FP) equation for the stochastic evolution problem. In this section, we provide a formal mathematical presentation of an MFG setting that will be needed in the coming sections.

A mean-field game is a setting where a large number of agents are interacting each of which tries to achieve its own goal while being influenced by an aggregated effect of the other agents. An agent, as a building block of this population, solves a stochastic optimal control problem. Hence, we first review the setting of stochastic control when the system is governed by Itô dynamics:

2.1 Stochastic Optimal Control

We start with assuming the probability space $(\Omega, \mathcal{F}, P)$ where Ω is the sample space, $\mathcal{F}$ is a sigma field and P is the probability measure. Let $\mathcal{F}_{t \geq 0}$ be a right-continuous filtration and assume an m -dimensional $(\mathcal{F}_t)_{t \geq 0}$ -adapted Brownian motion is given.

For a measurable space $(U, \mathcal{U})$, called *space of actions*, we define the set of *control strategies* $\mathbb{U}_t$ for every $t \in [0, T]$ as

$$\mathbb{U}_t := \{u : [t, T] \times \Omega \rightarrow \mathcal{U} : u \text{ is } \mathcal{F}\text{-optional}\}. \quad (3)$$

The states of the agent under the control $u \in \mathbb{U}_s$ when started from $X_0 = x \in \mathbb{R}^d$ is described by the

Itô's integral as

$$X_t = x + \int_s^t b_r(X_r, u_r) dr + \int_s^t \sigma_r(X_r) dW_r \quad (4)$$

where $b : [0, T] \times \mathbb{R}^d \times U \rightarrow \mathbb{R}^d$ and $\sigma : [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}^{d \times m}$ are measurable maps. In this expression, b and σ are called the *drift* and *diffusion* coefficients respectively. We assume the required regularity conditions (See theorem 2.5 of Chapter 5 of [4]) for the existence of the strong solution $(X_t^{s,x,u})_{t \geq 0}$.

We define the instantaneous objective that the agent seeks to maximize, called *running reward* (or running cost if the agent minimizes it) function as

$$f : [0, T] \times \mathbb{R}^d \times U \rightarrow \mathbb{R} \quad (5)$$

and a terminal reward that materializes at the final time T as

$$g : \mathbb{R}^d \rightarrow \mathbb{R}. \quad (6)$$

Both f and g are assumed to be measurable and bounded to avoid pathological situations.

When the agent follows the control strategy $u \in \mathbb{U}_t$, the *expected payoff* for a certain $t \in [0, T]$ is defined as

$$J_t(x, u) := \mathbf{E} \left[\int_t^T f_s(X_s^{t,x,u}, u_s) ds + g(X_T^{t,x,u}) \right]. \quad (7)$$

The value function is defined as the maximal obtainable expected payoff by the control strategies available in $\mathbb{U}_t$ as

$$V_t(x) := \sup_{u \in \mathbb{U}_t} J_t(x, u). \quad (8)$$

This definition of the value function makes it a solution of an equation called Hamilton-Jacobi-Bellman (HJB) equation. This result is formalized in the so-called *verification theorem*. Before presenting the theorem, we define the function $H : [0, T] \times \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^{d \times d}$, called *Hamiltonian* as

$$H_t(x, z, \gamma) := \sup_{u \in U} \{f_t(x, u) + \langle z, b_t(x, u) \rangle + \frac{1}{2} \text{Tr}[\sigma_t \sigma_t^\top(x) \gamma]\} \quad (9)$$

where $\langle \cdot, \cdot \rangle$ represents the inner product. Now comes the key theorem of this section that characterises the value function.

Theorem 1 (Verification Theorem, Thm. 3.5.2 of [5]). *Let $v \in C^{1,2}([0, T] \times \mathbb{R}^d)$ and its growth is upper bounded by a quadratic function, i.e.*

$$|v_t(x)| \leq C(1 + |x|^2), \quad \forall (t, x) \in [0, T] \times \mathbb{R}^d. \quad (10)$$

Suppose that $v_T(x) = g(x)$ on $\mathbb{R}^d$ and v is a solution of the HJB equation

$$\partial_t v_t(x) = H_t(x, \partial_x v_t(x), D_{xx} v_t(x)). \quad (11)$$

Then $v = V$, i.e., the function v coincides with the value function as defined by (8).

The control can be easily derived from the value function using the following result.

Lemma 2 (Optimal Control Lemma). *Suppose the conditions of the verification theorem are satisfied and there exists a measurable function $\tilde{u} : [0, T] \times \mathbb{R}^d \rightarrow U$ such that,*

$$H_t(x, \partial_x v_t(x), D_{xx} v_t(x)) = f_t(x, \tilde{u}_t(x)) + \partial_x v_t(x) + \frac{1}{2} \text{Tr}[\sigma \sigma^\top D_{xx} v_t](x), \quad (12)$$

then $\tilde{u}$ is an optimal (Markovian) control.

In short, Theorem 1 states that a sufficiently smooth and growth-bounded solution of the HJB equation (11) is the value function of the optimal control problem (8) and the optimal control is found by maximizing the Hamiltonian point-wise.

When we construct the connection between the federated learning and MFG in Section 3, each local client is perceived as an agent who tries to solve an optimal control problem as Equation (8). The exact form of the expected payoff J_t and the Hamiltonian H_t for each local learning client will be detailed in Section 3.

To prepare for the presentation of mean-field games in the next section, we first briefly review the concept of stochastic differential games.

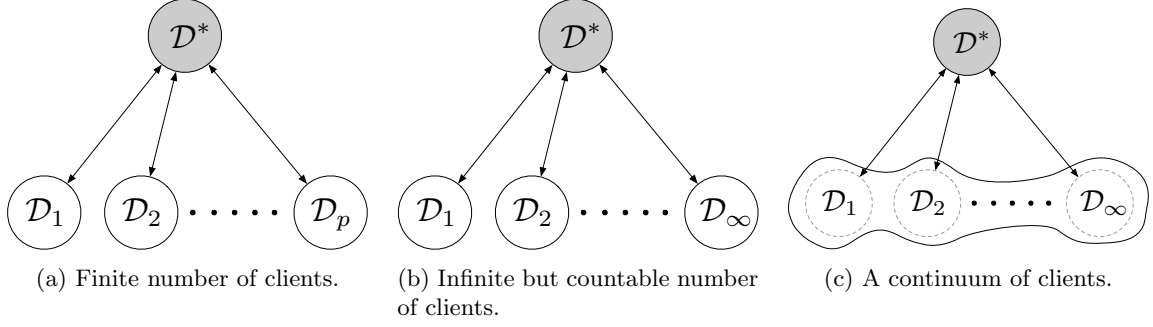


Figure 1: Schematics of federated learning with different cardinality of the set of clients.

2.2 Stochastic Differential Games

We considered a single agent in Section 2.1 that controls a stochastic system. The extension of the setting to a finite number of interacting agents is studied in game theory. Assume $p \in \mathbb{N}$ agents try to maximize their benefits from a common stochastic system. We provide the mathematical formalization of the theory here as a preparation to introduce the mean-field games in the next section.

Let's first fix a few notational conventions. Assume O represents a property such as a state, control, etc. Then, $O_{i,t}$ is the property of the i -th agent at time t . The notation O_i is the property O for the i -th agent for the entire considered time (e.g. $0 \leq t \leq T$ or $0 \leq t \leq \infty$). A boldfaced notation is used to show the concatenation of the properties for all agents at a certain time. That is, $\mathbf{O}_t = (O_{1,t}, \dots, O_{p,t})$. Hence, the coarsest-grained object is $\mathbf{O}$ that encapsulates the property O of all agents at all considered times.

Let $[p] := \{1, 2, 3, \dots, p\}$ be the set of p agents (or *players* in this context). Each agent $i \in [p]$ has its own space of strategies

$$\mathbb{U}_{i,t} := \{u : [t, T] \times \Omega \rightarrow U_i : u \text{ is } \mathcal{F}\text{-progressive}\}, \quad (13)$$

where the pair $(U_i, \mathcal{U}_i)$ is the measurable space of actions for the i -th agent and $t \in [0, T]$. By defining the strategies of the agents as stochastic processes, it is important to notice the implicit assumptions on the randomness structure of the game to avoid measure theoretic issues.

Remark 3 (Information Structure of the game). *The strategies of all players $\{\mathbb{U}_1, \mathbb{U}_2, \dots, \mathbb{U}_p\}$ are adapted to the same filtration. That is, there is a mutual source of randomness among all players.*

Let $X_{i,t} \in \mathbb{R}^d$ represents the d -dimensional state of the i -th player. By concatenating the states of all p players in $\mathbf{X}_t \in \mathbb{R}^{pd}$, the concatenated state evolution forms a stochastic system that is jointly controlled by all players of the game as

$$\mathbf{X}_t = \mathbf{x}_s + \int_s^t b_s(\mathbf{X}_s, u_{1,s}, u_{2,s}, \dots, u_{p,s}) + \int_s^t \sigma_s(\mathbf{X}_s) dW_s. \quad (14)$$

By assuming the necessary requirements on the coefficients b and σ , there exists a strong solution for (14) represented by $(\mathbf{X}_t^{s,\mathbf{x},\mathbf{u}})_{t \geq 0}$ that starts from the initial state $\mathbf{x}_s$ at time $t = s$ where $\mathbf{u} = (u_1, u_2, \dots, u_p)$ is the concatenation of the strategies of all players. The payoff for the i -th agent at time t is then defined as

$$J_{i,t}(\mathbf{x}, u_i, u_{-i}) := \mathbf{E} \left[\int_t^T f_{i,s}(\mathbf{X}_s^{t,\mathbf{x},\mathbf{u}}, \mathbf{u}_s) ds + g_i(\mathbf{X}_T^{t,\mathbf{x},\mathbf{u}}) \right] \quad (15)$$

where f_i and g_i are the running and terminal cost for the i -th agent. The Nash equilibrium is then defined for this stochastic differential game as the following.

Definition 1. *A vector of control strategies $(\tilde{u}_1, \tilde{u}_2, \dots, \tilde{u}_p) \in \mathbb{U}_{1,0} \times \mathbb{U}_{2,0} \times \dots \times \mathbb{U}_{p,0}$ is called a Nash-equilibrium of the stochastic differential game If*

$$J_{i,0}(\mathbf{x}, \tilde{u}_i, \tilde{u}_{-i}) \geq J_{i,0}(\mathbf{x}, u_i, \tilde{u}_{-i}) \quad (16)$$

for every player $i \in [p]$ and every strategy $u_i \in \mathbb{U}_{i,0}$

This definition implies that, at a Nash equilibrium, there is no motivation for any agent to change its control strategy while other agents stay with their current strategies. The existence of Nash-equilibrium in a stochastic differential game can be shown by PDE methods (e.g. see [6]). Dealing with Nash-equilibria becomes quickly intractable when the number of players increases in a strategically interacting game. The mean-field view offers a mathematically delicate way to handle the prohibitively large number of agents.

2.3 Mean-field games

The interaction among distinct players in a stochastic differential game can quickly become intractable when the number of players increases. A simplifying step is to symmetrize the problem and consider all agents alike. Hence, instead of studying every player separately, one can study a *representative player* whose interaction with the other players is statistical. As a result, the running cost f and the terminal cost g in the payoff (15) do not depend on the player index $i \in [p]$. We assume for now that σ is an $n \times n$ diagonal matrix. Hence, each player has its own stochastic dynamics whose Brownian motion is independent of the Brownian motion of the other players. Moreover, we made the assumption that a representative player interacts with the other players statistically. This makes the drift b a function of the distribution of players' states instead of each player's action. That is

$$b : [0, T] \times \mathbb{R}^d \times \mathcal{P}(\mathbb{R}^d) \times U \rightarrow \mathbb{R}^d \quad (17)$$

where $\mathcal{P}(\mathbb{R}^d)$ denotes the set of all probability measures defined on $\mathbb{R}^d$. In the absence of the population distribution of players' states, b is a function of the empirical distribution of states at time t as

$$\mu_t^p = \sum_{i \in [p]} \delta_{X_{i,t}}, \in \mathcal{P}(\mathbb{R}^d) \quad (18)$$

Remark 4 (Convergence of Random measures). *The empirical distribution μ^p is a random measure. It is intuitively clear that as $p \rightarrow \infty$, this random measure gets closer to a deterministic measure. The conditions under which the random empirical measure converges to a deterministic measure in the narrow topology P -a.s. is investigated in Glivenko-Cantelli theorem [7].*

The representative player is modeled by a d -dimensional stochastic differential equation on the same stochastic basis $(\Omega, \mathcal{F}, P)$ with a m -dimensional Brownian motion. We assume a measure on the space of $\mathbb{R}^d$ -valued continuous functions of time as $\mu \in \mathcal{P}(C([0, T], \mathbb{R}^d))$. The measure μ determines how likely every trajectory of the diffusion system is. Let define the time projection map $\pi_t : C([0, T], \mathbb{R}^d) \rightarrow \mathbb{R}^d$ that takes a sample from the input trajectory at time t . This induces a probability measure on the system's states at time t defined as

$$\mu_t := \mu \circ \pi_t^{-1}. \quad (19)$$

The *representative player* undergoes a stochastic dynamics as

$$X_t = x_s + \int_s^t b_r(X_r, \mu_r, u_r) dr + \int_s^t \sigma_r(X_r) dW_r, \quad (20)$$

where $b : [0, T] \times \mathbb{R}^d \times \mathcal{P}(\mathbb{R}^d) \times U \rightarrow \mathbb{R}^d$ and $\sigma : [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}^{d \times m}$. Similar to Equation (14), necessary conditions are assumed for the coefficients b and σ so that (20) has a strong solution. One of these conditions is that b must be Lipschitz with respect to its arguments. To check this condition with respect to the probability measure argument of b , the space $\mathcal{P}(\mathbb{R}^d)$ needs to be endowed with the Wasserstein metric; then the normal Lipschitz condition follows.

If the set of admissible controls for the representative player is shown by $\mathbb{U}$ defined as

$$\mathbb{U} := \{u : [0, T] \times \Omega \rightarrow U : u \text{ is } \mathcal{F}\text{-optional}\}, \quad (21)$$

the expected representative payoff will be defined as

$$J^\mu(x, u) := \mathbf{E} \left[\int_0^T f_t(X_t^{0,x,u,\mu}, \mu_t, u_t) dt + g(X_T^{0,x,u,\mu}, \mu_T) \right]. \quad (22)$$

where $X^{0,x,u,\mu}$ is the unique strong solution of (20). It is assumed that the running cost $f : [0, T] \times \mathbb{R}^d \times \mathcal{P}(\mathbb{R}^d) \times U \rightarrow \mathbb{R}$ and the terminal cost $g : \mathbb{R}^d \times \mathcal{P}(\mathbb{R}^d) \rightarrow \mathbb{R}$ are measurable and bounded to avoid pathological cases.

Now, we are ready to define the Nash equilibrium counterpart for the mean-field games.

Definition 2. A mean-field equilibrium is a measure $\mu \in \mathcal{P}(C([0, T], \mathbb{R}^d))$ on the trajectories $C([0, T], \mathbb{R}^d)$ such that

- There exists a strategy (control) $u^* \in \mathbb{U}$ where

$$J^\mu(x, u^*) \geq J^\mu(x, u), \forall u \in \mathbb{U} \quad (23)$$

- The probability measure μ is the law of the state trajectories of the representative player $X^{0,x,u^*,\mu}$ under the strategy u^* , that is,

$$\mu_t = \text{Law}(X_t^{0,x,u^*,\mu}). \quad (24)$$

The existence of the mean-field equilibria has been studied under several assumptions [8, 9]. At the equilibrium, the distribution (24) is plugged into (22) resulting in McKean-Vlasov type dynamics

$$X_t = x_s + \int_s^t b_r(X_r, \text{Law}(X_r), u_r) dr + \int_s^t \sigma_r(X_r) dW_r. \quad (25)$$

The solution of this equation is an equilibrium of the mean-field game whose existence requires Lipschitz assumption with respect to the Wasserstein metric on $\mathcal{P}(\mathbb{R}^d)$.

By taking a test function $\phi \in C^\infty(\mathbb{R}^d)$ and applying Itô's formula, the distributional evolution of the states, called the Fokker-Planck equation, is derived as

$$-\partial_t \mu_t(x) = \text{div}_x(b_t(x, \text{Law}(X_t), u_t) \mu_t(x)) + \frac{1}{2} D_{xx}^2(\text{Tr}[\sigma_t \sigma_t^\top](x) \mu_t(x)), \quad (26)$$

that characterizes a flow in the space of probability measures.

It can be seen from Theorem 1 and the definition of the Hamiltonian (9) that for optimal control $\hat{u}$,

$$b_t(x, \mu_t, \hat{u}_t) = D_z H_t(x, \partial_t v_x(x), D_{xx} v_t(x)) \quad (27)$$

where $v_t(x)$ is the value function defined similar to (8). Plugging (27) in (26) gives together with the HJB equation (11), two coupled PDEs of the mean-field game.

Remark 5. *Heuristically speaking, according to Equation (14), at the Nash equilibrium, each player needs to know the action of all other agents. However, in the mean-field game approach, the player only needs to have statistical information about the states of the other players.*

3 Mean-Field Federated Learning

After introducing the concept of federated learning in Section 1.1, stochastic differential games in Section 2.2 and mean-field games in Section 2.3, we are ready to make a bridge and show how federated learning (FL) can be perceived as a mean-field game. We will use the notations introduced in all previous parts throughout this section.

In a bird's-eye view, in an FL setting, each client of the federation interacts with the other clients only through its contribution to the server model. We first argue that the clients can be considered as players in a stochastic differential game.

Suppose that the local model h trained by client $k \in [p]$ is represented by the weight vector $w_k \in \mathbb{R}^d$. For example, w_k can be the weights of a neural network or the coefficients of the basis functions in a kernel-based method. In connection with MFG, the weights w_k are the states of the differential game. The differential nature of the game comes from the fact that gradient-based training can be seen as an ordinary or stochastic differential equation in the space of the model's weights. There are several causes of randomness in training. Stochastic gradient descent, as the workhorse of ML algorithms, inherits the stochasticity of choosing a single sample or a mini-batch randomly at each iteration. In neural networks, the drop-out layers act as another source of randomness during training. Inspired by [10], we encapsulate all sources of noise in a Brownian motion type of randomness that makes the weight trajectories fluctuate over the course of training. In practice, each client has access to the empirical distribution (samples from the underlying data-generating distribution). This can also be seen as another source of randomness but we assume it is also incorporated in the Brownian motion stochasticity and write loss functions based on the data-generating distribution for convenience. Hence, the loss function for client k is

$$\mathcal{L}_{\mathcal{D}_k}(w) = \mathbf{E}_{(x,y) \sim \mathcal{D}_k} [\ell(h(x; w), y)] \quad (28)$$

where $\mathcal{D}_k$ is the local data distribution of the client. Hence, the training dynamics of SGD is written as

$$dw_{k,t} = \nabla_{w_{k,t}} \mathcal{L}_{\mathcal{D}_k}(w_{k,t})dt + \sigma_{k,t}(w_{k,t})dW(t), \quad (29)$$

where $\sigma_k : [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}^{d \times m}$ is the diffusion coefficient for client k and $W(t)$ is a m -dimensional Brownian motion. Notice that, unlike (14), there is no explicit control term in this system. However, there is an implicit controller in optimization algorithms that determines how the estimated gradient at each iteration is used to update the weights. Suppose $g_{k,t}$ represents the gradient computed by client k at time t , that is, $g_{k,t} := \nabla_{w_{k,t}} \mathcal{L}_{\mathcal{D}_k}(w_{k,t})$. The way g_k influences the weight trajectory can be controlled in a generic way as

$$dw_{k,t} = b_t(g_{k,t}, u_{k,t})dt + \sigma_{k,t}(w_{k,t})dW(t), \quad (30)$$

where $u_{k,t} \in \mathcal{U}$ is the strategy employed by client k that determines the way $g_{k,t}$ is used to update $w_{k,t}$ at time t . A special case of b_t is a linear map that modulates the magnitude of gradients along each dimension of the weight vector. Hence, $b(g, u) := u \times g$ where g is a vector, u is a matrix and $\times$ is a matrix product. In particular, $u_{k,t} = \Lambda_{k,t} := \text{diag}(\lambda_{k,t}) = \text{diag}(\lambda_{k,t}^{(1)}, \lambda_{k,t}^{(2)}, \dots, \lambda_{k,t}^{(d)})$ and

$$dw_{k,t} = \Lambda_{k,t} g_{k,t} dt + \sigma_{k,t}(w_{k,t})dW(t), \quad (31)$$

shows the continuous training dynamics when the motion along each weight dimension is controllable. An incentive behind the control structure of (31) is an observation in practical applications of federated learning that shows the important effect of the number of epochs each client is trained before the aggregation phase starts. It has been observed that too long or too short training of clients are both detrimental to the outcome of federated learning [1]. Hence, $\Lambda_{k,t}$ is a proxy that emulates the length of training for client k at time t . More concretely, by upper bounding $\bar{\lambda}_{k,t} = \max(\lambda_{k,t}^{(1)}, \lambda_{k,t}^{(2)}, \dots, \lambda_{k,t}^{(d)})$, one can control the influence of the dataset available to client k to the aggregated model.

For example, a client may have a large dataset that makes its contribution to the aggregated model, when the server averages over all the clients, non-negligible; but the dataset of that client becomes faulty or unreliable at some time t_{fault} . Decreasing the control coefficient that acts as a learning rate, in this case, can nullify the effect of this faulty dataset in the aggregated model. The client still contributes to the averaged model but the contribution is based on the best previous state of the model as the training over the faulty dataset was skipped by decreasing the learning rate.

The goal of federated learning is to find a model that performs well on a target distribution. Let $\mathcal{D}^*$ denote the target distribution. Usually, as mentioned earlier in Equation (2) of Section 1.1, the target distribution is defined as a mixture of the clients' local distributions, that is

$$\mathcal{D}^* = \sum_{k=1}^p \alpha_k \mathcal{D}_k \quad (32)$$

where $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_p)$ belongs the simplex of p values. Hence, the ultimate goal of federated learning is to minimize

$$\mathcal{L}_{\mathcal{D}^*}(w_{\text{server}}) = \mathbf{E}_{(x,y) \sim \mathcal{D}^*} [\ell(h(x; w_{\text{server}}), y)] \quad (33)$$

with respect to the server model's weight vector $w_{\text{server}} \in \mathbb{R}^d$. This loss function cannot be evaluated and optimized directly as $\mathcal{D}^*$ is not available in one place. Instead, the gradients or updated models calculated by the clients are aggregated to update the weights of the server model. For example, in the aggregation phase at time t

$$dw_{\text{server},t} = \left(\sum_{k=1}^p \alpha_k g_{k,t} \right) dt. \quad (34)$$

The server weight is then sent to the clients to update the local models. As the information content of the differential of the server's weights is the aggregation of the differentials of all clients, we add the mean-field term (34) to the weight's dynamics of each client that turns (31) into

$$dw_{k,t} = \left(\Lambda_{k,t} g_{k,t} + \sum_{k=1}^p \alpha_k g_{k,t} \right) dt + \sigma_{k,t}(w_{k,t})dW(t). \quad (35)$$

As a result, Equation (35) models the whole federated learning process in a single continuous-time SDE where the effect of local learning and aggregation are unified in the drift term. In particular, the drift term at the RHS of (35) conforms with the observation that in federated learning, the parameters of the model learned by each client is affected by the local client first, and the weights of all other clients after the models are aggregated on the server and the local models are synchronized to the server model. In the continuous-time domain, the order of local updates and aggregated updates vanishes as both occur continuously. The length of local learning though can be controlled by the controller signal $(\Lambda_t)_{t \geq 0}$ that manages the local learning rates.

One of the first motivating applications of federated learning was predicting the stream of words while typing with the cellphone's keyboard [1]. Likewise, image recognition and captioning algorithms can feed on the large bulk of data that is globally available across millions of cell phones. In these setups, the ordinary federated learning with finite number of clients (Figure 1a) can be approximated by a differential game with infinite number of clients (Figure 1b) or even a continuum of clients (Figure 1c). This allows us to move from Equation (18) to Equation (26) and derive the mean-field formulation with a probability density over the weights of the clients (see Remark 4) instead of considering each client's weight vector separately.

Let w_t be the weight vector of the representative client in a federated setting with $p \rightarrow \infty$. Assume the distribution of the gradients $g_{k,t}$ of all clients at time t is μ_t^g and the diagonal matrix $\Lambda_t \in \mathbb{R}^{d \times d}$ is the representative control that modulates the gradients along every weight dimension. Hence, we have

$$w_t = w_s + \int_s^t b_r(w_r, \mu_r^g, \Lambda_r) dr + \int_s^t \sigma_r(w_r) dW_r \quad (36)$$

whose analogy to the dynamical equation (Fokker-Planck equation) of MFG is clear when compared with Equation (20). In the following, we show that the second equation of MFG (HJB equation) can also be derived in federated learning.

To derive the HJB equation, one needs to identify an optimal control setup in federated learning. There are various ways one can define a cost for the process of federated learning. Due to the fact that clients are normally distant from each other, the cost of communication with the server is a natural definition of cost in a federated setting. This becomes more critical in applications where the clients are cell phones which operate on battery and communicate with the server through a band-limited and costly internet connection. Moreover, the cell phone devices are in action while their local models are being trained. Hence, not only their final model but also their overall performance over the rounds of federated training matters. In our continuous-time dynamics, the communication rounds in federated learning are interlaced with the gradient dynamics that locally learn the weights of the clients. Hence, the number of communications with the server is approximated by the terminal time T in continuous-time dynamics. Let $(w_t^{0, w_0, \Lambda, \mu})_{t \in [0, T]}$ be the solution of (36) with initial weight w_0 , control strategy $\Lambda = (\Lambda_t)_{t \in [0, T]}$, and the flow of probability measures $\mu = (\mu_t)_{t \in [0, T]}$. We fix T and define the running cost f_t as the risk of the server model on distribution D^* at time t , that is

$$f_t(w) = \mathcal{L}_{D^*}(w) = \mathbf{E}_{(x, y) \sim D^*} [\ell(h(x; w), y)]. \quad (37)$$

This definition for the running cost emphasizes our desire that we want the model to perform well not only at the end but also during the course of federated learning. This is especially the case in infinite-horizon learning applications, where learning is always happening due to various reasons such as the non-stationarity of the local datasets. For example, the data distribution on mobile phones may change over time as the users' behavior and habits change.

Hence, the expected payoff becomes

$$J^\mu(x, \Lambda) := \mathbf{E} \left[\int_0^T f_t(w_t^{0, x, \Lambda, \mu}) + g(w_T^{0, x, \Lambda, \mu}, \mu_T) \right], \quad (38)$$

where g is the cost of the final model when the federated learning process is concluded. This is a generic formulation that can be specialized to various scenarios. For example, if we are only concerned with the performance of the final trained model, we can simply set $g = f_T$ and $f_t = c$ for $t \in [0, T]$ where the constant c shows the relative importance of learning in the fewest federated update rounds. Having the payoff function, it is straightforward to construct the Hamiltonian and the HJB equation. These steps are the same as Equations (9) and (27), therefore, we do not repeat them here.

Inspired by Definition 1, the equilibrium of this game is a set of strategies $\{u_1, u_2, \dots, u_p\}$ for the local clients where no learner is willing to change its strategy unilaterally. In the special case of Equation (35) where the control strategy is modifying the learning rate multiplicatively by time-dependent functions $\{(\Lambda_{1,t})_{t \in [0,T]}, (\Lambda_{2,t})_{t \in [0,T]}, \dots, (\Lambda_{p,t})_{t \in [0,T]}\}$, at the Nash equilibrium, changing the control function $(\Lambda_{k,t})_{t \in [0,T]}$ by the k -th local learner increases the cost. Notice that this equilibrium concerns a more general objective than the final model learned by the federated learning process and is consequently defined in a space different from the hypothesis family $\mathcal{H}$. It is defined in $([0, T] \rightarrow \mathcal{U})^k := ([0, T] \rightarrow \mathcal{U}) \times ([0, T] \rightarrow \mathcal{U}) \times \dots \times ([0, T] \rightarrow \mathcal{U})$, the space of functions that control the schedule of learning for every local learner such that the transient and asymptotic performance of the aggregated model (output of the federated learning at time t during training) is optimal in the sense of the generic cost defined by (38).

It was shown in this section that constructing the dynamical system of federated learning as a combination of local learning dynamics and federated aggregation, results in a system of equations similar to those that are derived in mean-field games. This result highlights the bridge we aimed to build between federated learning and mean-field games.

4 Conclusion

Federated learning as a distributed and privacy-preserving training method is currently receiving considerable attention due to the omnipresence of the edge devices such as mobile phones and the amount of data they create on daily basis. In addition to the complexity of the theoretical analysis of local learning methods which is the case in any non-distributed deep learning algorithm, the distributed learning and aggregation of the model updates make federated learning twofold complicated for theoretical analysis. To abstract the high-level and dynamical aspect of federated learning from the subtleties of learning the local models, we formalized every local learning process as a dynamical system. The aggregation that is performed at each round of federated learning is modeled as a mean-field effect where each local learner is affected by an average (mean-field) effect of the other learners. Therefore, we showed that the entire process can be modeled as a mean-field game, a topic that is studied in physics, game theory, probability theory, and optimal control. We hope mean-field games opens up a new set of theoretical tools to understand the behavior of distributed learning algorithms, especially, federated learning.

References

- [1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.
- [2] Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014.
- [3] Jean-Michel Lasry and Pierre-Louis Lions. Mean field games. *Japanese journal of mathematics*, 2(1):229–260, 2007.
- [4] Ioannis Karatzas and Steven Shreve. *Brownian motion and stochastic calculus*, volume 113. springer, 2014.
- [5] Huy  n Pham. *Continuous-time stochastic control and optimization with financial applications*, volume 61. Springer Science & Business Media, 2009.
- [6] J Frehse and A Bensoussan. Nonlinear elliptic systems in stochastic game theory. *Journal f  r die reine und angewandte Mathematik*, 1984(350):23–67, 1984.
- [7] Aad W Van der Vaart. *Asymptotic statistics*, volume 3. Cambridge university press, 2000.
- [8] Pierre Cardaliaguet. Notes on mean field games. Technical report, Technical report, 2010.
- [9] Ren   Carmona, Fran  ois Delarue, et al. *Probabilistic Theory of Mean Field Games with Applications I-II*. Springer, 2018.
- [10] Xuanqing Liu, Si Si, Qin Cao, Sanjiv Kumar, and Cho-Jui Hsieh. Neural sde: Stabilizing neural ode networks with stochastic noise. *arXiv preprint arXiv:1906.02355*, 2019.