# COMPARING FEDERATED STOCHASTIC GRADIENT DESCENT AND FEDERATED AVERAGING FOR PREDICTING HOSPITAL LENGTH OF STAY

Mehmet Yigit Balik

Aalto University, Espoo, Finland

## **ABSTRACT**

Predicting hospital length of stay (LOS) reliably is an essential need for efficient resource allocation at hospitals. Traditional predictive modeling tools frequently have difficulty acquiring sufficient and diverse data because healthcare institutions have privacy rules in place. In our study, we modeled this problem as an empirical graph where nodes are the hospitals. This modeling approach facilitates collaborative model training by modeling decentralized data sources from different hospitals without extracting sensitive data outside of hospitals. A local model is trained on a node (hospital) by aiming the generalized total variation minimization (GTVMin). Moreover, we implemented and compared two different federated learning optimization algorithms named federated stochastic gradient descent (FedSGD) and federated averaging (FedAVG). Our results show that federated learning enables accurate prediction of hospital LOS while addressing privacy concerns without extracting data outside healthcare institutions.

*Index Terms*— Federated Learning, Networks, Personalized Machine Learning, Length of Stay

## 1. INTRODUCTION

In hospitals, efficiently managing resources and managing patient flow are increasingly complicated tasks. Managing how long patients stay in the hospital is seen as a highly effective strategy for smartly handling hospital resources during times of crisis [1]. Predicting the length of stay (LOS) of a patient is crucial for meeting the rising demand for healthcare services by making the best use of available resources [2]. The LOS indicates the time frame that time spent by a patient starting from admission to discharge [3].

The LOS is a complicated factor influenced by many aspects, including the patient's medical condition and social circumstances [4], which makes it hard to predict by humans. Research exploring the use of machine learning (ML) methods for LOS in hospitals has become more prevalent in the literature. Many researchers studied predicting LOS by applying centralized ML approaches, where all data is stored in a single environment. In their study, Alsinglawi

et al. [5] constructed a predictive framework utilizing ML regression models to forecast LOS for patients hospitalized with cardiovascular conditions in the Intensive Care Unit. Mekhaldi et al. [6] applied ML methods to predict LOS in hospitals. Their methodology includes data preprocessing, model validation, and comparison of outcomes using two different ML regression models. Turgeman et al. [7] utilized administrative hospital data to develop a tree-based model aimed at predicting LOS for patients admitted with congestive heart failure upon admission. Tsai et al. [8] conducted a comparative analysis between artificial neural networks and linear regression to assess their performance in predicting the length of stay for heart failure inpatients. However, while ML techniques show promise, they often require large amounts of data. Given the limited availability of medical datasets and their sensitive nature, achieving optimal performance can be challenging. Furthermore, conducting the training process using all data simultaneously proves to be both expensive and ineffective. Additionally, consolidating data from all hospitals overlooks the possibility that certain hospitals may exhibit similar patterns, while others may not. In this context, federated learning presents an alternative. Federated learning can improve model accuracy while preserving data privacy by training ML models on decentralized data from various hospitals.

In this study, we structured the problem as an empirical graph where each hospital is a node on the graph. This method allows different hospitals to work together on training local models without needing to share sensitive patient data. We trained a model for each hospital using a method called generalized total variation minimization (GTVMin). Additionally, we tested two different ways of model training, called federated stochastic gradient descent (FedSGD) and federated averaging (FedAVG), and compared their performances.

This paper is outlined as follows: Section 2 explains the dataset used, empirical graph construction, and formulation of the GTVMin instance for FedSGD and FedAVG. Section 3 discusses data preparation, feature selection, local model choice, and implemented federated learning algorithms. Section 4 showcases the performances of applied algorithms. Finally, our key findings are summarized in the final section.

#### 2. PROBLEM FORMULATION

## 2.1. Dataset Definition

In our study, we used *Microsoft Predicting Length of Stay Dataset* [9]. The dataset contains 100,000 data points representing patients' information (e.g. gender, glucose level, body-mass index, pulse, etc.) and patients' LOS. Table 1 shows the details of the used data fields in this study. Section 3 will discuss the applied prepossessing techniques on the features. A local dataset  $\mathcal{D}^{(i)}$  defined as follows:

$$\mathcal{D}^{(i)} = \{ \{ \mathbf{x}^{(i,1)}, y^{(i,1)} \}, ..., \{ \mathbf{x}^{(i,m_i)}, y^{(i,m_i)} \} \}$$
 (1)

Where  $\mathbf{x}^{(i,r)}$  and  $y^{(i,r)}$  indicate features and the label of the  $r^{th}$  data point in the local dataset  $\mathcal{D}^{(i)}$ . Finally,  $m_i$  indicates the size of the local dataset  $\mathcal{D}^{(i)}$ , which might differ between different nodes. We can also represent features as a feature matrix and labels as a label vector as follows:

$$\mathbf{X}^{(i)} = (\mathbf{x}^{(i,1)}, ..., \mathbf{x}^{(i,m_i)})^T$$
, and  $\mathbf{y}^{(i)} = (y^{(i,1)}, ..., y^{(i,m_i)})^T$ 

Data Field	Type	Description	
rcount	Categorical	Number of readmissions	
		within last 180 days	
gender	String	Gender of the patient - M	
		or F	
hemo	String	Flag for blood disorder	
		during encounter	
hematocrit	Float	Average hematocrit value	
		during encounter (g/dL)	
neutrophils	Float	Average neutrophils	
•		value during encounte	
		(cells/microL)	
sodium	Float	Average sodium value du	
		ing encounter (mmol/L)	
glucose	Float	Average glucose value	
		during encounte	
		(mmol/L)	
bloodureanitro	Float	Average blood urea ni	
		trogen value during en	
		counter (mg/dL)	
creatinine	Float	Average creatinine value	
		during encounter (mg/dL)	
bmi	Float	Average BMI during en	
		counter (kg/m <sup>2</sup> )	
pulse	Float	Average pulse during en	
		counter (beats/m)	
respiration	Float	Average respiration during	
		encounter (breaths/m)	
n_conditions	Integer	Number of clinical condi	
		tions that the patient has	
lengthofstay (label)	Integer	Length of stay for the en	
		counter	
facid (node ID)	Integer	Facility ID at which the	
		encounter occurred	

Table 1: Description of Data Fields

## 2.2. Empirical Graph

The empirical graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is an undirected weighted graph with nodes  $\mathcal{V} = \{1, ..., n\}$  where each node has a local dataset formulated as (1). Each node has a local model parameterized by  $\mathbf{w}^{(i)}$ . An undirected edge  $(i, j) \in \mathcal{E}$  denotes the similarity between local datasets  $\mathcal{D}^{(i)}$  and  $\mathcal{D}^{(j)}$  and has the weight  $A_{i,j} \in \{0,1\}$ .

We measure the discrepancy between two local datasets  $\mathcal{D}^{(i)}$  and  $\mathcal{D}^{(j)}$  using a discrepancy measure  $D(\mathbf{w}^{(i)}, \mathbf{w}^{(j)})$  (such as Euclidean distance, Kullback–Leibler divergence or Wasserstein distance), which accepts local model weights as its parameters. Given a minimum node degree d, we connect this node to other d nodes with the smallest discrepancy and set the corresponding edge weight to 1. During the construction of the empirical graph  $\mathcal{G}$ , we learn local model parameters  $\mathbf{w}^{(i)}$  by minimizing a local loss function  $L_i(\mathbf{w}^{(i)})$  on local dataset  $\mathcal{D}^{(i)}$  without sharing information between nodes. The motivation behind our approach is to learn weights that represent the local dataset in the best way. Then we can use these weights as the representation of the corresponding local dataset and measure dissimilarity between nodes.

## 2.3. Formulation of GTVMin Instance

Our main objective is to learn the best local model parameters  $\hat{\mathbf{w}}^{(i)}$  by minimizing their local loss as well as enforcing the small total variation [10]. In this paper, we implemented two different federated optimization algorithms.

## 2.3.1. FedSGD

The formulation of GTVMin is given as follows [10]:

$$\mathbf{stack}\{\hat{\mathbf{w}}^{(i)}\}_{i=1}^{n} \in \underset{\mathbf{stack}\{\mathbf{w}^{(i)}\}_{i=1}^{n}}{\operatorname{argmin}} \sum_{i \in \mathcal{V}} L_{i}(\mathbf{w}^{(i)}) \\
+\alpha \sum_{(i,j) \in \mathcal{E}} A_{i,j} \|\mathbf{w}^{(i)} - \mathbf{w}^{(j)}\|_{2}^{2} \tag{3}$$

## 2.3.2. FedAVG

We can also achieve same local model weights  $\hat{\mathbf{w}}^{(i)}$  and solve GTVMin as follows [10]:

$$\begin{split} \hat{\mathbf{w}} \in & \underset{\mathbf{w} \in \mathcal{C}}{\operatorname{argmin}} \sum_{i \in \mathcal{V}} L_i(\mathbf{w}^{(i)}) \\ & \text{with } \mathcal{C} = \{ \mathbf{w} = \mathbf{stack} \{ \mathbf{w}^{(i)} \}_{i=1}^n : \mathbf{w}^{(i)} = \mathbf{w}^{(j)} \\ & \text{for any } i, j \in \mathcal{V} \} \end{split} \tag{4}$$

## 3. METHODS

## 3.1. Data Preparation and Feature Selection

Dataset contains data collected from 5 different hospital facilities. Table 2 gives the number of data points in each local

dataset. Each local dataset is divided into training, validation, and test sets randomly using seed 42 with percentages of 70%, 15%, and 15%, respectively. We use standard split to ensure that the model is robust, generalizes well to new data, and avoids overfitting, leading to more reliable and accurate machine learning applications. The validation set is used to tune hyperparameters such as  $\alpha$  value in (3), the minimum node degree d and the learning rate  $\eta$  in (8), (9) and (11).

Local Dataset	Dataset Size
$\mathcal{D}^{(1)}$	30012
$\mathcal{D}^{(2)}$	30035
$\mathcal{D}^{(3)}$	30755
$\mathcal{D}^{(4)}$	4499
$\mathcal{D}^{(5)}$	4699

Table 2: Sizes of Local Datasets

There were 28 data fields including the label and facility ID (i.e. LOS value) in the dataset [9]. However, irrelevant features (admission id, date of discharge, etc.) are dropped and the remaining features are indicated in Table 1. We dropped those features since we aim to infer LOS utilizing the medical information of the patient rather than dates of admission and discharge or ID of the admission. Binary features (gender and hemo) are used without any preprocessing. There are also different binary features indicating the patient's conditions such as depression and asthma in the original dataset. However, we decided to create a new feature named  $n\_conditions$  that represents the number of conditions a patient has and we dropped the binary condition features for the sake of sim-"4", "5+"} is converted one-hot-encoding format where we fixed value "5+" to 5 to eliminate ambiguity. The numerical features (hematocrit, neutrophils, sodium, glucose, bloodureanitro, creatinine, bmi, pulse, and respiration) are normalized as follows:

Let  $x_j^{(i,r)}$  represent the value of the  $j^{th}$  feature in the  $r^{th}$  data point in the local training set of node i. Then, the mean and standard deviation of the j-th feature in the local training set can be denoted as  $\bar{x}_j^{(i)}$  and  $\sigma_j^{(i)}$ , respectively. The normalization of the j-th feature in both the local training and test sets can be expressed as:

For the training set: 
$$x_j^{(i,r)} \leftarrow \frac{x_j^{(i,r)} - \bar{x}_j^{(i)}}{\sigma_j^{(i)}}$$

For the validation/test set: 
$$x_j^{(i,r')} \leftarrow \frac{x_j^{(i,r')} - \bar{x}_j^{(i)}}{\sigma_i^{(i)}}$$

where r' denotes the index of a data point in the local validation/test set of node i.

#### 3.2. Local Model

Local linear models (5) are preferred for predicting LOS due to their interpretability, efficiency, and robustness, making them suitable for healthcare analytics. Their transparent interpretations of coefficients allow for understanding the impact of predictor variables on outcomes, crucial in clinical decision-making. Additionally, their computational efficiency and scalability enable handling large datasets and real-time predictions in healthcare settings effectively.

$$\mathbf{y}^{(i)} \approx \mathbf{X}^{(i)} \mathbf{w}^{(i)} \tag{5}$$

We selected Euclidean distance as the variation measure between two local datasets  $\mathcal{D}^{(i)}$  and  $\mathcal{D}^{(j)}$  by projecting the corresponding learned local model parameters to Euclidean space shown in (6). We choose Euclidean distance due to its intuitive interpretation, mathematical simplicity, and compatibility with local linear models, ensuring effective comparison between local models. Its straightforward computation and consideration of both the magnitude and direction of differences make it a suitable and widely accepted metric for assessing dissimilarity between parameter vectors.

$$D(\mathbf{w}^{(i)}, \mathbf{w}^{(j)}) = \|\mathbf{w}^{(i)} - \mathbf{w}^{(j)}\|_{2}$$
 (6)

In the context of predicting LOS with local linear models, the mean squared error (MSE) loss function is well-suited for training. MSE penalizes the squared difference between predicted and actual LOS values, providing a smooth optimization landscape conducive to efficient model training. More importantly, it is a good indication that the model prediction deviates from the ground truth. Hence, we also use MSE as the evaluation metric during the testing.

$$L_i(\mathbf{w}^{(i)}) = \frac{1}{m_i} \|\mathbf{y}^{(i)} - \mathbf{X}^{(i)} \mathbf{w}^{(i)}\|_2^2$$
 (7)

## 3.3. Federated Learning Algorithms

#### 3.3.1. FedSGD

FedSGD employs a gradient descent (GD) approach to solve (3). This algorithm enforces similar weights across nodes by message passing between neighboring nodes where the construction of edges plays an important role [10]. Variation between nodes penalizes weight.

$$\mathbf{w}^{(i,k+1)} \leftarrow \mathbf{w}^{(i,k)} + \eta[-\nabla L_i(\mathbf{w}^{(i,k)}) + 2\alpha \sum_{(i,j)\in\mathcal{E}} A_{i,j}(\mathbf{w}^{(i)} - \mathbf{w}^{(j)})]$$
(8)

Moreover, FedSGD does not operate on the whole training dataset at a time. It randomly samples *b* training data from the training set to represent the whole training data.

#### 3.3.2. FedAVG Version 1 (FedAVGv1)

FedAVGv1 employs a projected GD approach to solve (4) by (i) taking a local gradient step in each node, then (ii) sending updated weights to a center, and finally (iii) calculating the average of updated weights and distributing it to the nodes [10]. The mathematical formulation is given as:

$$\hat{\mathbf{w}}_k^{(i)} \leftarrow \mathbf{w}^{(i,k)} - \eta \nabla L_i(\mathbf{w}^{(i,k)})$$
(local gradient step) (9)

$$\mathbf{w}^{(i,k+1)} \leftarrow \frac{1}{n} \sum_{i' \in \mathcal{V}} \hat{\mathbf{w}}_k^{(i')}$$
(projection) (10)

## 3.3.3. FedAVG Version 2 (FedAVGv2)

In FedAVGv1, the client takes a gradient step and waits for global weights. In some situations, this waiting time can be long and wasted. Hence, a client can use that time efficiently by using a local minimization around global weights [10]. FedAVGv2 follows the procedure (i) local minimization around the global weights, (ii) sending updated weights to a center, and finally (iii) calculating the average of updated weights and distributing it to the nodes [10] to solve (4). The mathematical formulation is given as:

$$\hat{\mathbf{w}}_k^{(i)} \leftarrow \underset{\mathbf{v}}{\operatorname{argmin}} L_i(\mathbf{v}) + (1/\eta) \|\mathbf{v} - \mathbf{w}^{(i,k)}\|_2^2$$
(local minimization around global weights)

$$\mathbf{w}^{(i,k+1)} \leftarrow \frac{1}{n} \sum_{i' \in \mathcal{V}} \hat{\mathbf{w}}_k^{(i')} \tag{12}$$

(update of global weights)

In FedAVG, the edges do not have any effect on the training since this algorithm operates in a server-client paradigm (the empirical graph is assumed to be a star graph where all nodes have only one neighbor which is the server) [10]. Similar weights are enforced by averaging local weights to obtain global weights and distributing these global weights to nodes again.

## 4. RESULTS

Tuning hyperparameters is a crucial step to obtain the most suitable empirical graph and the optimal local model weights. FedSGD algorithm has 3 hyperparameters listed below:

- α value in (3): Determines importance of variation across different model weights.
- The minimum node degree d: Determines the minimum number of neighbors for a node i.

• The learning rate  $\eta$  in (8), (9) and (11): Determines the size of gradient step.

In FedAVG algorithms, we assume a star graph without explicitly constructing the empirical graph. Moreover, the similarity of local model weights is enforced by averaging at a server node without needing  $\alpha$ . Hence, the only hyperparameter for FedAVG algorithms is  $\eta$ . We fixed the maximum number of iterations to 1000 for both algorithms and set batch size b = 512 for the FedSGD algorithm. We did not tune these hyperparameters. These hyperparameters were selected among  $\alpha \in \{1, 0.5, 0.1\}, \eta \in \{0.1, 0.01, 0.001\},\$ and  $d \in \{1, 2, 3, 4\}$ , with the additional constraint that the graph must be connected, based on the lowest error observed on the validation set. For FedSGD, optimal parameters were  $\alpha = 0.1$ ,  $\eta = 0.1$  and d = 2. The constructed empirical graph is shown in Figure 2 (recall that the edge weights are 1). For both FedAVG algorithms, the optimal learning rate obtained is  $\eta = 0.1$ .

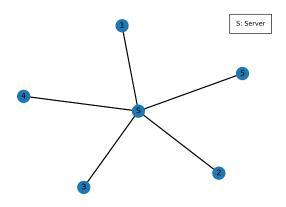


Fig. 1: Assumed Empirical Graph in FedAVG

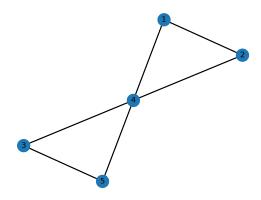


Fig. 2: Constructed Empirical Graph Used in FedSGD

Table 3 and Table 4 depict the training and validation errors

obtained with these optimal hyperparameters, respectively. Although all three algorithms have MSE less than 2, the FedSGD algorithm has outperformed FedAVG algorithms, achieving an MSE of 1.377 on the training set and 1.407 on the validation set, which is approximately 0.4 less error compared to other algorithms. Hence, FedSGD is chosen as more applicable in our study.

	FedSGD	FedAVGv1	FedAVGv2
Node	Training	Training	Training
Noue	MSE	MSE	MSE
1	1.136	1.728	1.828
2	1.123	1.724	1.829
3	1.603	2.104	2.236
4	1.52	1.704	1.767
5	1.503	1.819	1.895
mean	1.377	1.816	1.911

**Table 3**: Training Errors

It is important to note that for each algorithm and each node, training, and validation errors are similar, which indicates the training was done properly without any overfitting.

	FedSGD	FedAVGv1	FedAVGv2
Node	Validation	Validation	Validation
Noue	MSE	MSE	MSE
1	1.142	1.749	1.843
2	1.214	1.81	1.913
3	1.585	2.085	2.216
4	1.577	1.764	1.839
5	1.514	1.763	1.827
mean	1.407	1.834	1.928

Table 4: Validation Errors

Table 5 shows the achieved test errors (on the test set which is neither used for training nor validation see Section 3.1) by FedSGD, FedAVGv1 and FedAVGv2. Again, the FedSGD algorithm has shown way better performance, achieving an MSE of 1.354 compared to FedAVGv1 with 1.798 MSE and FedAVGv2 with 1.897 MSE.

	FedSGD	FedAVGv1	FedAVGv2
Node	Test MSE	Test MSE	Test MSE
1	1.061	1.666	1.767
2	1.097	1.706	1.81
3	1.685	2.224	2.358
4	1.361	1.525	1.592
5	1.565	1.87	1.957
mean	1.354	1.798	1.897

Table 5: Test Errors

#### 5. CONCLUSION

In this paper, we have modeled the problem of predicting hospital LOS as an empirical graph and trained local linear models in nodes to solve GTVMin. The empirical graph is constructed by measuring the Euclidean distance between nodes using the local model weights. As the local model, we selected linear models and MSE as the loss function. We have implemented and compared three different federated learning algorithms, named FedSGD and two different versions of FedAVG, to predict hospital LOS. In the first version of FedAVG (FedAVGv1), a local model takes a single gradient step over the global weights and sends the updated local weights to the center for federated averaging. On the other hand second version (FedAVGv2) local models employ a local minimization around the global weights and then send the updated local weights to the center for averaging.

Experimental results show that all three algorithms perform accurately and achieve a satisfactory and competitive MSE values on test sets, comparable to existing methods. However, FedSGD outperformed both FedAVGv1 and FedAVGv2, achieving the lowest MSE on training, validation, and test sets. The main reason for FedSGD's superior performance can be attributed to the fact that facilities do not necessarily have similar data. This allows FedSGD to tune the  $\alpha$  parameter, thereby adjusting the strength of information sharing between different facilities. In contrast, FedAVG-based algorithms average local model weights and broadcast them to the nodes, which can have a negative impact when different facilities contain highly dissimilar data.

Although our study presents promising results with proper training without overfitting (the maximum difference between training error and validation error is around 0.1), there are areas for improvement. Firstly, the choice of hyperparameters, such as the learning rate and the minimum node degree, could be further optimized to enhance model performance. Additionally, investigating alternative discrepancy measures and federated learning algorithms could lead to even better results. Finally, extending the study to include more decentralized datasets could provide insights into the applicability of the proposed approach to real life. We leave these improvements for future work.

## 6. REFERENCES

- [1] Z. Wang, J. S. Ji, Y. Liu, R. Liu, Y. Zha, X. Chang, L. Zhang, Q. Liu, Y. Zhang, J. Zeng *et al.*, "Survival analysis of hospital length of stay of novel coronavirus (covid-19) pneumonia patients in sichuan, china," *Medrxiv*, pp. 2020–04, 2020.
- [2] S. Ellahham and N. Ellahham, "Use of artificial intelligence for improving patient flow and healthcare delivery," *J. Comput. Sci. Syst. Biol*, vol. 12, no. 3, 2019.
- [3] O. Khosravizadeh, S. Vatankhah, P. Bastani, R. Kalhor, S. Alirezaei, and F. Doosty, "Factors affecting length of stay in teaching hospitals of a middle-income country," *Electronic physician*, vol. 8, no. 10, p. 3042, 2016.
- [4] S. Shea, R. V. Sideli, W. DuMouchel, G. Pulver, R. R. Arons, and P. D. Clayton, "Computer-generated informational messages directed to physicians: effect on length of hospital stay," *Journal of the American Medical Informatics Association*, vol. 2, no. 1, pp. 58–64, 1995.
- [5] B. Alsinglawi, F. Alnajjar, O. Mubin, M. Novoa, M. Alorjani, O. Karajeh, and O. Darwish, "Predicting length of stay for cardiovascular hospitalizations in the intensive care unit: Machine learning approach," in 2020 42nd annual international conference of the IEEE engineering in medicine & biology society (EMBC). IEEE, 2020, pp. 5442–5445.
- [6] R. N. Mekhaldi, P. Caulier, S. Chaabane, A. Chraibi, and S. Piechowiak, "Using machine learning models to predict the length of stay in a hospital setting," in World conference on information systems and technologies. Springer, 2020, pp. 202–211.
- [7] L. Turgeman, J. H. May, and R. Sciulli, "Insights from a machine learning model for predicting the hospital length of stay (los) at the time of admission," *Expert Systems with Applications*, vol. 78, pp. 376–385, 2017.
- [8] P.-F. J. Tsai, P.-C. Chen, Y.-Y. Chen, H.-Y. Song, H.-M. Lin, F.-M. Lin, and Q.-P. Huang, "Length of hospital stay prediction at the admission stage for cardiology patients using artificial neural network," *Journal of health-care engineering*, vol. 2016, 2016.
- [9] Microsoft, "Predicting hospital length of stay." [Online]. Available: https://microsoft.github.io/ r-server-hospital-length-of-stay/input\_data.html
- [10] A. Jung, "Federated learning," Lecture notes of the course CS-E4740, 2024.