Towards a Federated Fuzzy Learning System

Towards a Federated Fuzzy Learning System

Anna Wilbik
Department of Data Science
and Knowledge Engineering
Maastricht University,
Maastricht, The Netherlands
Email: a.wilbik@maastrichtuniversity.nl

Paul Grefen
School of Industrial Engineering
Eindhoven University of Technology and
Atos Digital Transformation Consulting,
Eindhoven, The Netherlands
Email: P.W.P.J.Grefen@tue.nl

***Abstract*—The abundant availability of data allows the construction of predictive systems that support decision makers in business and society. A problem arises if an organization does not have a large enough data set by itself to construct a system of adequate quality. In this case, data across organizations has to be used, which introduces risks of data sharing. To overcome these risks, federated learning is getting increasingly popular to enable automated learning in distributed networks of autonomous partners without sharing raw data. So far, only crisp systems have been used in this context. The use of a fuzzy inference system can bring advantages to deal with vagueness and uncertainty in predictive systems. Therefore, in this paper we explore the (hopefully) happy marriage of federated learning and fuzzy inference mechanisms. We show that it is indeed possible to build a fuzzy inference model in a federated learning setting, resulting in a Federated Fuzzy Learning System ($\mathtt{F^2LS}$). We also show that this combination brings advantages to decision making that cannot be achieved with either mechanism in isolation.**

I. Introduction

Currently, more and more data is being gathered in business processes or automatically produced in the context of the internet-of-things (IoT). For instance, in the healthcare domain the amount of collected data is being doubled every two years [1]. Many organizations want to use this data to improve their decision making. One popular way to achieve this is by building predictive models from the data that inform decision makers of expected future situations by analyzing patterns in past situations.

Building high-quality predictive models requires a historic data set of adequate size to learn from. Sometimes, however, an individual organization does not have enough data to build a predictive model of sufficient quality by itself. Obtaining additional data from other parties may be impossible because of competitive threats or privacy regulations, e.g. the EU General Data Protection Regulation (GDPR) [2]. In such situations, federated learning can offer a solution by using data across organizations for building models without introducing problems related to threats or regulations.

Federated learning enables a collaboration between multiple parties to jointly train a machine learning model without exchanging the local data [3]. Because the data are not exchanged between parties, it is considered a privacy preserving approach. The collaboration in learning is considered successful, if for at least one party the performance of the federated model is better than the performance of the local model [4].

Federated learning has been successfully applied in various domains. Federated learning is used to combine data of mobile devices (so called cross-device federated learning), e.g., on Google Keyboard [5], and data of different organizations (so called cross-silo federated learning), e.g., in healthcare [6] or in finance for transaction fraud detection [7]. We have explored the use of federated learning for processing IoT data to support decision making in business processes, building a concept model [8] and a demonstrator [9].

So far, learning in a federated context has been based on crisp models. In application domains with substantial vagueness or uncertainty, the use of fuzzy inference mechanisms can yield better predictive algorithms, however. With these domains in mind, we have posed a question: Is it possible to build a fuzzy inference system model in a federated setting to bring the advantages of fuzzy reasoning to a federated context? And more detailed: Could fuzzy inference systems deal with heterogeneity of data, by allowing the existence of multiple locally active rules?

In this paper we present our approach to build a fuzzy inference model in a federated learning setting, resulting in a Federated Fuzzy Learning System ($\mathtt{F^2LS}$). The next section discusses the background and related work. Section III explains the basis for our approach. Section IV describes the proposed approach for building an $\mathtt{F^2LS}$. Section V discusses the initial results of applying our approach to test cases. The paper finishes with concluding remarks.

II. Background: Related Work

In this section, we briefly discuss the relevant background and related work on the topics of federated systems in general and federated learning specifically. We address fuzzy inference mechanisms in the next section.

Federated systems. As there are many application contexts in which systems are not tightly integrated, but coupled in a loose fashion, federated systems have been studied for decades. A typical class of systems is the federated database system [10], in which several autonomous systems each manage their own local databases. The contents of these can be combined for purposes at the global level, i.e., the level of the federation. The loose coupling requires specific protocols to use the data at the federation level, e.g., for integrity control [11]. Federated learning systems are federated systems

978-1-6654-4407-1/21/$31.00 © 2021 IEEE

in which specific protocols are used to combine data from federation members for machine learning purposes.

Federated learning. Federated learning is defined as "a machine learning setting where multiple entities (clients) collaborate in solving a machine learning problem, under the coordination of a central server or service provider. Each client's raw data is stored locally and not exchanged or transferred; instead, focused updates intended for immediate aggregation are used to achieve the learning objective" [3].

The greatest advantage of the federated learning is improving the model quality, by using data available to all parties involved in a federation, yet keeping raw data at its source. This is achieved by communicating only model vectors, thereby improving the data privacy of federation members. There are still many challenges in federated learning [3], such as improving efficiency, guaranteeing the privacy of members, providing robustness to the attacks, ensuring fairness among members and addressing bias in learning.

Federated learning has been proposed for several learning paradigms [12]. A popular paradigm is supervised learning for regression problems (cf. [13], [14], [15]). We adopt this paradigm in our approach towards an $\texttt{F}^2\texttt{LS}$. Several machine learning models were adapted to learn in the federated setting, such as neural networks [13], [16], tree based models [14], [17], SVMs [18] and linear models [15], but not fuzzy inference systems.

Many authors (cf. [4], [19], [12]) mention heterogeneity of data (or non-IID data) as one of the major challenges in federated learning. Several methods try to solve this issue by adjusting the relative weights of the data of members better, e.g., using adaptive weighting [20], [21], regularization [19] or clustering [22].

III. GROUNDWORK: BASIS FOR AN $\texttt{F}^2\texttt{LS}$

In this section, we lay the groundwork for the development on an $\texttt{F}^2\texttt{LS}$. First, we discuss the learning process followed by a federated learning system. Then, we discuss the inference mechanism we use in learning.

Federated learning process. In the basic federated learning process [3] a server orchestrates the training process, by repeating the following steps:

1) Client selection: The server selects the clients for the next training iteration. This selection may depend on satisfying some eligibility requirements by the clients.
2) Broadcast: The selected clients download the current model weights and a training program from the server.
3) Client computation: Each selected device locally computes an update to the model by executing the training program. In case of Federated Averaging [13] it is based on a Stochastic Gradient Descent (SGD) mechanism.
4) Aggregation: The server collects the updates from the clients and aggregates them.
5) Model update: The server updates the federated model based on the aggregated update computed from the clients that participated in the current round.

Fuzzy inference mechanism. In this paper we consider a Takagi-Suegno fuzzy inference system [23] (TSFIS). This is a rule-based system with J rules. The j-th rule takes the form:

$$\begin{aligned} &\text{if } x_1 \text{ is } A_{j1} \text{ and } \dots \text{ and } x_n \text{ is } A_{jn} \\ &\text{then } y_j = a_{j0} + a_{j1}x_1 + \ldots + a_{jn}x_n \end{aligned} \tag{1}$$

The degree of fulfilment of the j-th rule is calculated as:

$$B_j(x) = \prod_{n=1}^{N} \mu_{A_{jn}}(x) \tag{2}$$

The output is a weighted average of the outputs of the individual rules:

$$f(x, \theta) = \frac{\sum_{j=1}^{J} B_j(x) y_j}{\sum_{j=1}^{J} B_j(x)} \tag{3}$$

There are several methods how to learn a TSFIS from the data, e.g., [24], [25], [26]. Traditionally a TSFIS is developed in two steps. In the first step, known as structure identification, a suitable number of rules and a proper partition of the feature space is determined. This can be done by using a clustering algorithm, e.g., the fuzzy c-means clustering method [27]. In this case each cluster forms one rule. The fuzzy sets A_{ji} of the rule antecedents are defined by a Gaussian membership function, where the mean c is the cluster center and the standard deviation σ is interfered from the partition matrix. The second step, known as parameter identification, updates the parameters and tunes the system. To find the parameters of the rules consequent one may use the least squares method [24] or the gradient decent method [25].

We show the formulas for the gradient descent method, as they are used later in this paper. At step $k+1$ and for training data pair (x^m, y^m), the parameters of the consequent for the j-th rule are:

$$a_{j,i}(k+1) = a_{j,i}(k) - \lambda e_m(k) \cdot \frac{B_j(x^m, k)}{\sum_{j=1}^{J} B_j(x^m, k)} \cdot x_j^m \tag{4}$$

for i=1,..,n.

$$a_{j,0}(k+1) = a_{j,0}(k) - \lambda e_m(k) \cdot \frac{B_j(x^m, k)}{\sum_{j=1}^{J} B_j(x^m, k)} \tag{5}$$

where $e_m(k)$ is the error in approximating the output, so $e_m(k) = y^m - f(x^m, \theta(k))$

IV. METHOD: CONSTRUCTING AN $\texttt{F}^2\texttt{LS}$

The algorithm for training a fuzzy inference system in a federated setting is depicted in Algorithm 1. This algorithm follows both the two-step process of training the fuzzy inference model and the general federated learning process discussed in the previous section.

In the first stage of the algorithm (structure and rule antecedent identification), the server requests each client to cluster their local data and return to the server the cluster centers and the standard deviations. Next, similar clusters are merged (i.e., cluster centers that are close enough are averaged). For this purpose we use agglomerative hierarchical clustering with a predefined threshold. Also, two clusters from the same client cannot be merged. The number of merged clusters determines the number of rules in the $\mathtt{F^2LS}$. For each cluster, one rule will be formed. The the fuzzy sets in the rule antecedents are defined by the corresponding cluster as Gaussian membership with averaged cluster center $\bar{c}$ and averaged standard deviation $\bar{\sigma}$ as parameters.

In the second stage of the algorithm (rule consequent identification), we use the stochastic gradient descent algorithm in a federated setting. It means that each client selected in each round receives a federated model, runs E training passes of the stochastic gradient descent algorithm to find consequent parameters on a training batch of local data, and then returns the updated parameters to the server. The server updates the parameters of the rule consequent of the federated model as the weighted average of parameters returned by the clients in this round. The weights are dependent on the size of local data, such that large data sets have more influence than small data sets.

Algorithm 1: FedFIS algorithm

```
Server executes:
  initialize empty FIS
  // structure and rule antecedent identification
  [c_k, σ_k] ← Cluster(k)
  [c, σ] ← Merge(c_k, σ_k)
  add rules for each cluster (c, σ)
  // rule consequent identification
  for each round t=1,2,... do
  m ← max(C · K, 1)
  S_t ← (random set of m clients)
  for each client k ∈ S_t in parallel do
    w^k_{t+1} ← ClientUpdate(k, w_t)
  w_{t+1} ← Σ_{i=1}^{K} (n_k/n) w^k_{t+1}

Cluster(k): // Run on client k
m ← FindNumberOfClusters(P_k)
[c, σ] ← FCM(P_k,m)

ClientUpdate(k, w): // Run on client k
  B ← (Split P_k into batches of size B)
  for each local epoch i from 1 to E do
    for batch b ∈ B do
      w ← w − η∇ℓ(w; b)
  return w to server
```

For clustering the local data of each client, we use fuzzy c-means clustering [27]. The parameter, the number of clusters, is determined by the cluster correlation validity index based on Spearman correlation [28]. The standard deviation σ is calculated from memberships values of data points belonging to each cluster, using the definition of the Gaussian membership function.

The cluster merging algorithm uses hierarchical clustering of cluster centers to identify the clusters that can be merged based on their proximity. We use farthest neighbour clustering to make sure that all centers in the cluster are close to each other. Only clusters from different clients can be merged, and the server needs to define the maximal distance as the merge criterion. After merging the new cluster centers and standard deviations are the means of the appropriate parameters of the merged clusters.

The client update process follows the stochastic gradient descent method, with the formulas to update the parameters shown in (4)-(5). Those equations show the parameter updates just for one single data point. We follow the advice from [25] and postpone the update of the parameters until all data from the training batch has been seen and the parameter update is the mean of the individual updates for each data point in the local training batch.

V. Results: Testing an $\mathtt{F^2LS}$

We have tested the proposed $\mathtt{F^2LS}$ on two small data sets from the UCI repository [29]. The goal of these experiments is to verify whether one can train a fuzzy inference system in a federated setting. As a success criterion we use the one proposed by Li et al. [4], in which a federated model should improve the performance for at least one party.

A. Experimental setup

For the experiments we use the Auto MPG data set [30] and Wine Quality data set [31] from the UCI repository [29].

The Auto MPG data set concerns city-cycle fuel consumption in miles per gallon. It contains 398 observations described by 8 independent variables. We have removed observations with missing values and standardized this data set.

The Wine Quality data set concerns wine quality and results of physico-chemical tests. We have used a data set related to red wine only. In this data set 1599 wines are described by 11 features describing among others acidity, sugar levels, pH, alcohol levels. We have standardized this data set and removed the outliers, resulting in 1537 observations.

For the federated learning setting, we have divided the data among three clients. We also have considered three scenarios. In the first scenario, the data was divided equally. In the second scenario, the split was 30% - 35% - 35% . In the third scenario, the split was most skewed: 20% - 40% - 40%. Moreover, each client used only 80% of the data for training and the remaining 20% for testing purposes. The quality measures mean squared error (MSE) and mean absolute error (MAE) are shown for the test set.

For the gradient descent method, we have used the following parameters. The step size λ was set to 0.01. The stopping criterion was the biggest change of a parameter of a federated TSFIS in consequent parameters smaller than 0.0001 or 5000 iterations. The parameters for the federated learning were as follows. The fraction of clients that performed computation in each round C was set to 1, meaning that we used all 3 clients. The number of training passes that each client makes over its local data set E was set to 1. B, the local mini-batch size used for the client updates, was set to ∞ meaning that the whole training set was used in each round.

B. Numerical results

We have calculated MSE and MAE on the test sets available to each client, for both the local and federated models. Each experiment was repeated 20 times with random partition of the data. The means and standard deviations of the errors are shown in Table I for the Auto MPG data set and in Table II for the Wine Quality data set.

TABLE I
MEAN AND STANDARD DEVIATION OF MSE AND MAE OF LOCAL (LM) AND FEDERATED (FM) MODELS OF 20 REPETITIONS – AUTO MPG DATA SET

	$\mathcal{P}_1$	$\mathcal{P}_2$	$\mathcal{P}_3$
Scenario 1: equal partition			
MSE lm	0.185 (0.08)	0.156 (0.048)	0.149 (0.071)
MSE fm	0.168 (0.066)	0.158 (0.054)	0.138 (0.063)
MAE lm	0.306 (0.062)	0.297 (0.049)	0.287 (0.059)
MAE fm	0.294 (0.055)	0.29 (0.037)	0.28 (0.056)
Scenario 2: 30% - 35% - 35%			
MSE lm	0.171 (0.083)	0.171 (0.057)	0.166 (0.068)
MSE fm	0.151 (0.078)	0.141 (0.03)	0.143 (0.042)
MAE lm	0.299 (0.062)	0.303 (0.052)	0.302 (0.051)
MAE fm	0.285 (0.063)	0.287 (0.031)	0.282 (0.047)
Scenario 3: 20% - 40% - 40%			
MSE lm	0.161 (0.085)	0.173 (0.063)	0.165 (0.055)
MSE fm	0.133 (0.078)	0.165 (0.086)	0.154 (0.042)
MAE lm	0.282 (0.059)	0.304 (0.049)	0.304 (0.046)
MAE fm	0.259 (0.065)	0.293 (0.061)	0.290 (0.043)

TABLE II
MEAN AND STANDARD DEVIATION OF MSE AND MAE OF LOCAL (LM) AND FEDERATED (FM) MODELS OF 20 REPETITIONS – WINE QUALITY DATA SET

	$\mathcal{P}_1$	$\mathcal{P}_2$	$\mathcal{P}_3$
Scenario 1: equal partition			
MSE lm	0.636 (0.098)	0.675 (0.106)	0.659 (0.095)
MSE fm	0.625 (0.097)	0.655 (0.092)	0.624 (0.083)
MAE lm	0.618 (0.047)	0.638 (0.049)	0.638 (0.044)
MAE fm	0.614 (0.049)	0.628 (0.045)	0.626 (0.042)
Scenario 2: 30% - 35% - 35%			
MSE lm	0.638 (0.126)	0.647 (0.087)	0.622 (0.085)
MSE fm	0.612 (0.121)	0.625 (0.078)	0.613 (0.096)
MAE lm	0.617 (0.056)	0.625 (0.040)	0.623 (0.047)
MAE fm	0.607 (0.056)	0.613 (0.041)	0.614 (0.049)
Scenario 3: 20% - 40% - 40%			
MSE lm	0.619 (0.141)	0.653 (0.084)	0.671 (0.100)
MSE fm	0.599 (0.132)	0.643 (0.080)	0.654 (0.092)
MAE lm	0.614 (0.055)	0.636 (0.042)	0.636 (0.047)
MAE fm	0.599 (0.06)	0.630 (0.044)	0.628 (0.042)

The mean of the errors shows that the federated learning setting is successful, as all parties on average improve their performance quality. However among the 20 repetitions, there are a few cases in which the federated model didn't outperform any of the local models. Further research is required to learn in which cases joining a federation is beneficial for a party.

C. An example of local and federated models

Here we present one of the $\texttt{F}^2\texttt{LS}$ instances created for the Auto MPG data set, as well as a local model.

In the first stage (structure and rule antecedent identification), we have used the FCM clustering algorithm and cluster correlation validity index for each of the clients and their local training data set. For client 1 and client 2, 6 clusters were created, and for client 3, 8 clusters were created. The cluster centers of the clients are shown using a parallel coordinates plot [32] in Figure 1. The color indicates the client id.

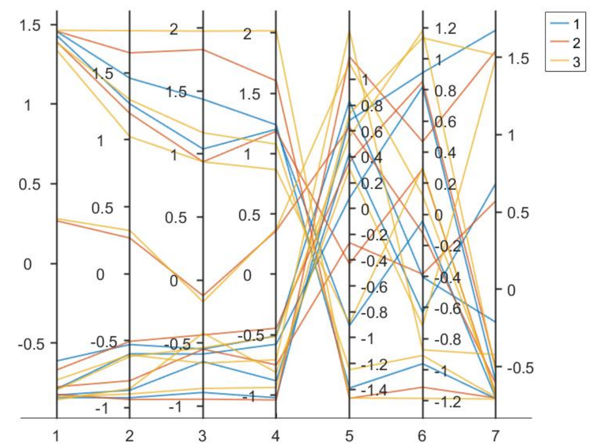

Fig. 1. Cluster centers for the clients 1, 2 and 3

After merging, 9 clusters remained. Note that all except one cluster center from set 3 were merged with each other. The centers of clusters of the federated model are depicted in Figure 2. The label "4" in the plot legend denotes a center of a merged cluster.

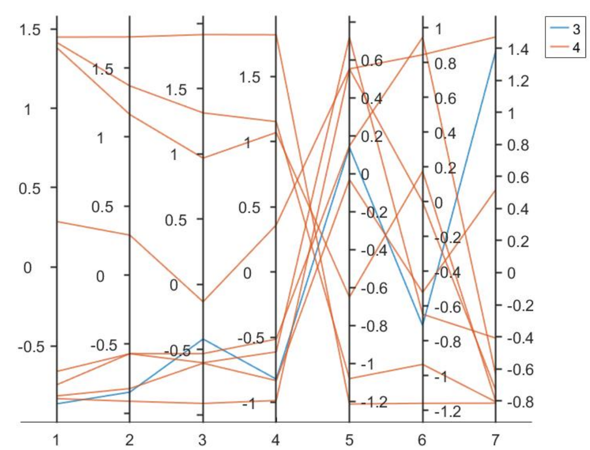

Fig. 2. Cluster centers after merging.

The local data and the initial clustering were used to obtain the local models. The MSE values of local and federated models are shown in Table III.

TABLE III
MSE LOCAL (LM) AND FEDERATED (FM) MODEL

	$\mathcal{P}_1$	$\mathcal{P}_2$	$\mathcal{P}_3$
MSE lm	0.261	0.274	0.071
MSE fm	0.191	0.234	0.063

We have generated the surface plots of the local and federated models. In Figure 3, we show the plots only for the first two variables. By comparing the surfaces, one can notice that the federated model is not the average of the local models, but a smart combination of them.

We can also look at the rules that are generated. As the federated model has 9 rules, we will focus only on two of them. In the first case, the definition of the antecedent is a result of a cluster merge. This rule concerns observations with very low values for the first 4 variables, and very high values for the remaining 3.

For the federated model the rule is as follows: if x_1 is very small and x_2 is very small and ... and x_7 is very large then y = - 0.105*x_1 + 0.252*x_2 - 0.134 *x_3 - 0.489*x_4 + 0.001*x_5 + 0.321*x_6 + 0.1470*x_7 + 0.001.

For the local model of client 1 the rule is as follows: if x_1 is very small and x_2 is very small and ... and x_7 is very large then y = -0.238*x_1 + 0.17*x_2 - 0.325*x_3 - 0.761*x_4 + 0.144*x_5+ 0.4*x_6 - 0.128*x_7 + 0.0375.

For the local model of client 2 the rule is as follows: if x_1 is very small and x_2 is very small and ... and x_7 is very large then y = -0.086*x_1 - 0.147*x_2 - 0.195*x_3 -0.254*x_4 + 0.143*x_5 + 0.372*x_6 + 0.171*x_7 + 0.055.

For the local model of client 3 the rule is as follows: if x_1 is very small and x_2 is very small and ... and x_7 is very large then y = -0.068*x_1 - 0.165*x_2 - 0.153*x_3 - 0.41*x_4 + 0.043*x_5 + 0.368*x_6 + 0.188*x_7 + 0.1157.

The second rule of the federated model originates from a cluster that was not merged (represented by a blue line in Figure 2). This rule in the federated model is: if x_1 is very small and x_2 is very small and x_3 is small ... and x_7 is very large then y = 0.051*x_1 - 0.032*x_2 - 0.149*x_3 - 0.178*x_4 + 0.155*x_5 - 0.154*x_6 - 0.047*x_7 -0.022.

The rule with the same antecedent exists also in the local model of client 3. It is: if x_1 is very small and x_2 is very small and x_3 is small ... and x_7 is very big then y = 0.022*x_1 + 0.002*x_2 - 0.036*x_3 - 0.052*x_4 + 0.126*x_5 - 0.125*x_6.

One can see that obtaining a federated rule is not simply a matter of averaging the corresponding local rules, but requires a more complex aggregating procedure.

In this case the constructed federated model outperforms all local models, and improves MSE by at least 15%. The federated model managed to find additional patterns that are not present in individual local models.

VI. Conclusion

In this paper, we have proposed an approach for building an $\mathtt{F^2LS}$, using a Takagi-Sugeno fuzzy inference system in a federated setting. The $\mathtt{F^2LS}$ approach integrates the best of two worlds: federated learning to deal with privacy-preserving data integration and learning and fuzzy inference to deal with uncertainty and vagueness in the contents of the learning process. We have shown that on average a federated model outperforms corresponding local models. We have presented a prototype approach in this paper, which requires further testing. In this testing, we will put an emphasis on cases with heterogeneous data.

There are also possibilities for further improvement of our approach. During our experiments, we have observed a few cases where the federated model was not better than any of the local models. In future work, we want to minimize this risk, for instance by looking at antecedent identification. Currently antecedents of the federated model are generated by merging clusters obtained by clustering algorithm on local data. In future work, we plan to investigate the possibility of using a federated version of the fuzzy c-means algorithm to arrive at an $\mathrm{F^2{}_CLS}$ variation of our system. Also more detailed privacy considerations pose an interesting question for further research, e.g., how much information is released by sharing the cluster centers with other parties.

References

[1] I. for Health Technology Transformation, “Transforming health care through big data: Strategies for leveraging big data in the health care industry,” 2013.

[2] “Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data (...) (general data protection regulation) oj l 119, 4.5.2016, p. 1–88.”

[3] E. by: Peter Kairouz and H. B. McMahan, “Advances and open problems in federated learning,” *Foundations and Trends® in Machine Learning*, vol. 14, no. 1, pp. –, 2021.

[4] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, and B. He, “A survey on federated learning systems: vision, hype and reality for data privacy and protection,” *arXiv preprint arXiv:1907.09693*, 2019.

[5] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, “Federated learning for mobile keyboard prediction,” *arXiv preprint arXiv:1811.03604*, 2018.

[6] T. M. Deist, A. Jochems, J. van Soest, G. Nalbantov, C. Oberije, S. Walsh, M. Eble, P. Bulens, P. Coucke, W. Dries *et al.*, “Infrastructure and distributed learning methodology for privacy-preserving multicentric rapid learning health care: eurocat,” *Clinical and translational radiation oncology*, vol. 4, pp. 24–31, 2017.

[7] W. Zheng, L. Yan, C. Gou, and F.-Y. Wang, “Federated meta-learning for fraudulent credit card detection,” in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)*, 2020.

[8] P. Grefen, H. Ludwig, S. Tata, R. Dijkman, N. Baracaldo, A. Wilbik, and T. D’hondt, “Complex collaborative physical process management: a position on the trinity of bpm, iot and da,” in *Working Conference on Virtual Enterprises*. Springer, 2018, pp. 244–253.

[9] T. d’Hondt, A. Wilbik, P. Grefen, H. Ludwig, N. Baracaldo, and A. Anwar, “Using bpm technology to deploy and manage distributed analytics in collaborative iot-driven business scenarios,” in *Proceedings of the 9th International Conference on the Internet of Things*, 2019, pp. 1–8.

[10] L. Liu and M. T. Özsu, Eds., *Federated Database Systems*. Springer US, 2009, pp. 1128–1128.

[11] P. Grefen and J. Widom, “Protocols for integrity constraint checking in federated databases,” *Distributed and Parallel Databases*, vol. 5, pp. 327–355, 1997.

[12] S. Ji, T. Saravirta, S. Pan, G. Long, and A. Walid, “Emerging trends in federated learning: From model fusion to federated x learning,” *arXiv preprint arXiv:2102.12920*, 2021.

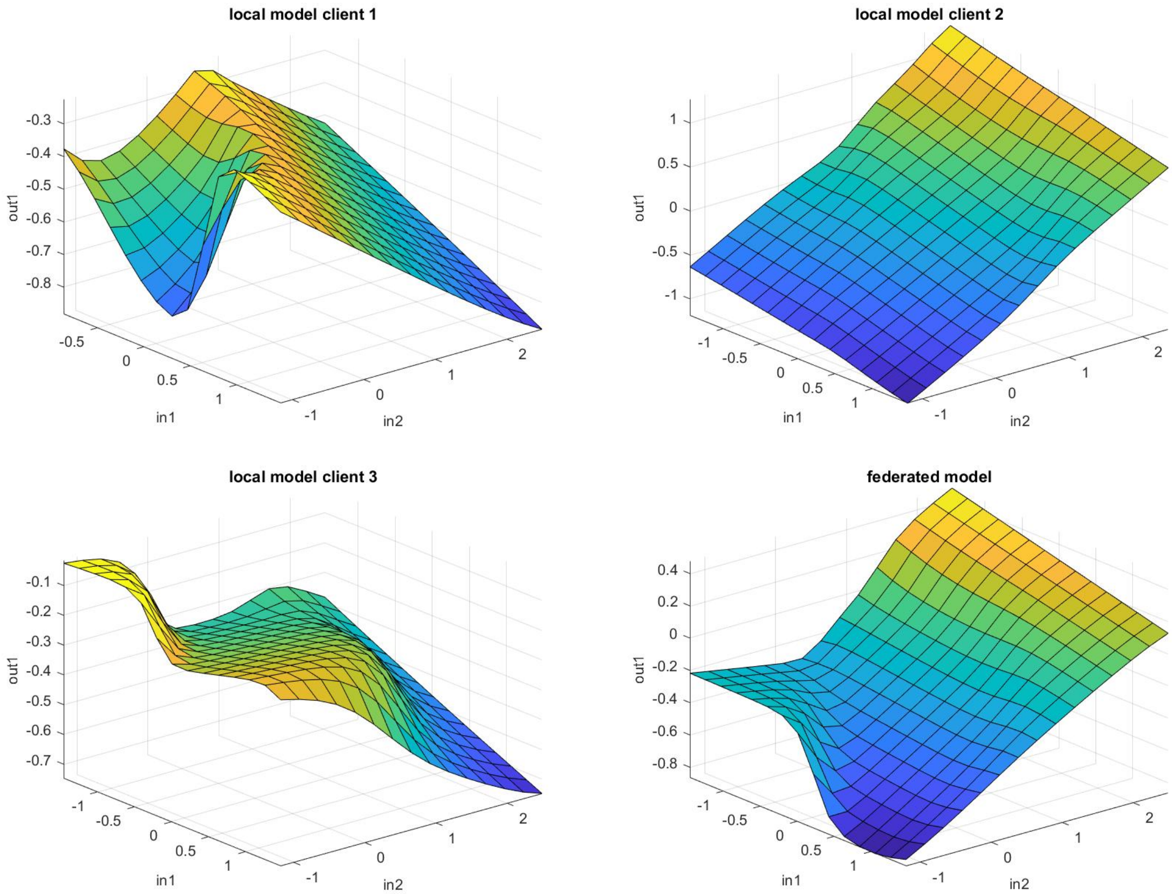

Fig. 3. Surface plots for local and federated models

[13] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[14] Q. Li, Z. Wen, and B. He, "Practical federated gradient boosting decision trees," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 4642–4649.

[15] F. Wang, H. Zhu, R. Lu, Y. Zheng, and H. Li, "A privacy-preserving and non-interactive federated learning scheme for regression training with gradient descent," *Information Sciences*, vol. 552, pp. 183–200, 2021.

[16] L. Zhang, Y. Shi, Y.-C. Chang, and C.-T. Lin, "Hierarchical fuzzy neural networks with privacy preservation for heterogeneous big data," *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 1, pp. 46–58, 2020.

[17] S. Rana, S. K. Gupta, and S. Venkatesh, "Differentially private random forest with high utility," in *2015 IEEE International Conference on Data Mining*. IEEE, 2015, pp. 955–960.

[18] Q. Li, Z. Wen, and B. He, "Adaptive kernel value caching for svm training," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 7, pp. 2376–2386, 2019.

[19] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127*, 2018.

[20] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 10, pp. 4229–4238, 2019.

[21] Y. Yeganeh, A. Farshad, N. Navab, and S. Albarqouni, "Inverse distance aggregation for federated learning with non-iid data," in *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning*. Springer, 2020, pp. 150–159.

[22] C. Briggs, Z. Fan, and P. 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)*. IEEE, 2020, pp. 1–9.

[23] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 1, pp. 116–132, 1985.

[24] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, *Neuro-fuzzy and Soft Computing, a Computational Approach to Learning and Machine Intelligence*, 1997.

[25] K. M. Passino and S. Yurkovich, *Fuzzy control*. MA: Addison-wesley, 1998.

[26] F. Herrera, M. Lozano, and J. L. Verdegay, "Tuning fuzzy logic controllers by genetic algorithms," *International Journal of Approximate Reasoning*, vol. 12, no. 3-4, pp. 299–315, 1995.

[27] J. C. Bezdek, "Models for pattern recognition," in *Pattern Recognition with Fuzzy Objective Function Algorithms*. Springer, 1981, pp. 1–13.

[28] M. Popescu, J. C. Bezdek, T. C. Havens, and J. M. Keller, "A cluster validity framework based on induced partition dissimilarity," *IEEE transactions on cybernetics*, vol. 43, no. 1, pp. 308–320, 2012.

[29] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[30] R. Quinlan, "Auto mpg dataset," 1993. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/auto+mpg

[31] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision support systems*, vol. 47, no. 4, pp. 547–553, 2009.

[32] A. Inselberg, "The plane with parallel coordinates," *The visual computer*, vol. 1, no. 2, pp. 69–91, 1985.