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 context, 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 (F²LS). 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 (F²LS). 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 F²LS. 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...
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 a fuzzy inference model for regression problems (cf. [13], [14], [15]). We adopt this approach towards an F2LS. 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 F2LS

In this section, we lay the groundwork for the development on an F2LS. 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-Sugeno fuzzy inference system [23] (TSFIS). This is a rule-based system with J rules. The j-th rule takes the form:

\[
\text{if } x_1 \text{ is } A_{j1} \text{ and } \ldots \text{ and } x_n \text{ is } A_{jn} \text{ then } y_j = \mu_{A_{jn}}(x) = a_{j0} + a_{j1}x_1 + \ldots + a_{jn}x_n
\]

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

\[
B_j(x) = \prod_{n=1}^{N} \mu_{A_{jn}}(x)
\]

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

\[
f(x, \theta) = \frac{\sum_{j=1}^{J} B_j(x) y_j}{\sum_{j=1}^{J} B_j(x)}
\]

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 $\sigma$ 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 $(x^n, y^n)$, the parameters of the consequent for the j-th rule are:

\[
a_{j,i}(k+1) = a_{j,i}(k) - \lambda e_m(k) \cdot \frac{B_j(x^n, k)}{\sum_{j=1}^{J} B_j(x^n, k)} \cdot x^n_m
\]

for i=1,...,n.

\[
a_{j,0}(k+1) = a_{j,0}(k) - \lambda e_m(k) \cdot \frac{B_j(x^n, k)}{\sum_{j=1}^{J} B_j(x^n, k)}
\]

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

### IV. Method: Constructing an F2LS

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 $F^2LS$. For
each cluster, one rule will be formed. 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 iden-
tification), 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
$[c_k, \sigma_k] \leftarrow \text{Cluster}(k)$
$[c, \sigma] \leftarrow \text{Merge}(c_k, \sigma_k)$
add rules for each cluster $(c, \sigma)$
// rule consequent identification
for each round $t=1,2,\ldots$ do
$m \leftarrow \max(C \cdot K, 1)$
$S_t \leftarrow$ (random set of $m$ clients)
for each client $k \in S_t$ in parallel do
$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{m} w_{t+1}^k$

Cluster$(k)$: // Run on client $k$
$m \leftarrow \text{FindNumberOfClusters}(P_k)$
$[c, \sigma] \leftarrow \text{FCM}(P_k, m)$

ClientUpdate$(k, w)$: // Run on client $k$
$B \leftarrow$ (Split $P_k$ into batches of size $B$)
for each local epoch $i$ from 1 to $E$ do
for batch $b \in B$ do
$w \leftarrow w - \eta \nabla \ell(w; b)$
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,
determined by the cluster correlation validity index based
on Spearman correlation [28]. The standard deviation $\sigma$ is cal-
culated 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 $F^2LS$

We have tested the proposed $F^2LS$ 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 consump-
tion 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 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 $\lambda$ 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 $\infty$ 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>
<thead>
<tr>
<th>Scenario 1: equal partition</th>
<th>( \mathcal{P}_1 )</th>
<th>( \mathcal{P}_2 )</th>
<th>( \mathcal{P}_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE lm</td>
<td>0.185 (0.08)</td>
<td>0.156 (0.048)</td>
<td>0.149 (0.071)</td>
</tr>
<tr>
<td>MSE fm</td>
<td>0.168 (0.066)</td>
<td>0.158 (0.054)</td>
<td>0.138 (0.063)</td>
</tr>
<tr>
<td>MAE lm</td>
<td>0.306 (0.062)</td>
<td>0.297 (0.049)</td>
<td>0.287 (0.059)</td>
</tr>
<tr>
<td>MAE fm</td>
<td>0.294 (0.055)</td>
<td>0.29 (0.037)</td>
<td>0.28 (0.056)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scenario 2: 30% - 35% - 35%</th>
<th>( \mathcal{P}_1 )</th>
<th>( \mathcal{P}_2 )</th>
<th>( \mathcal{P}_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE lm</td>
<td>0.171 (0.083)</td>
<td>0.171 (0.057)</td>
<td>0.166 (0.068)</td>
</tr>
<tr>
<td>MSE fm</td>
<td>0.151 (0.078)</td>
<td>0.141 (0.03)</td>
<td>0.143 (0.042)</td>
</tr>
<tr>
<td>MAE lm</td>
<td>0.299 (0.062)</td>
<td>0.303 (0.052)</td>
<td>0.302 (0.051)</td>
</tr>
<tr>
<td>MAE fm</td>
<td>0.285 (0.063)</td>
<td>0.287 (0.031)</td>
<td>0.282 (0.047)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scenario 3: 20% - 40% - 40%</th>
<th>( \mathcal{P}_1 )</th>
<th>( \mathcal{P}_2 )</th>
<th>( \mathcal{P}_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE lm</td>
<td>0.161 (0.085)</td>
<td>0.173 (0.063)</td>
<td>0.165 (0.055)</td>
</tr>
<tr>
<td>MSE fm</td>
<td>0.133 (0.078)</td>
<td>0.165 (0.086)</td>
<td>0.154 (0.042)</td>
</tr>
<tr>
<td>MAE lm</td>
<td>0.282 (0.059)</td>
<td>0.304 (0.049)</td>
<td>0.304 (0.046)</td>
</tr>
<tr>
<td>MAE fm</td>
<td>0.259 (0.065)</td>
<td>0.293 (0.061)</td>
<td>0.290 (0.043)</td>
</tr>
</tbody>
</table>

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 \( F_2^2 \) 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](image1)

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.](image2)

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.
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 \( x_3 \) is small and \( x_7 \) is very large then \( y = -0.105x_1 + 0.252x_2 - 0.134x_3 - 0.489x_4 + 0.001x_5 + 0.321x_6 + 0.1470x_7 + 0.001 \).

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 \( x_3 \) is small and \( x_7 \) is very large then \( y = -0.238x_1 + 0.177x_2 - 0.325x_3 - 0.761x_4 + 0.144x_5 + 0.4x_6 - 0.128x_7 + 0.0375 \).

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 \( x_3 \) is large then \( y = -0.238x_1 + 0.177x_2 - 0.325x_3 - 0.761x_4 + 0.144x_5 + 0.4x_6 - 0.128x_7 + 0.0375 \).

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 \( x_3 \) is small and \( x_7 \) is very large then \( y = -0.238x_1 + 0.177x_2 - 0.325x_3 - 0.761x_4 + 0.144x_5 + 0.4x_6 - 0.128x_7 + 0.0375 \).

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 \( x_3 \) is small and \( x_7 \) is very large then \( y = -0.238x_1 + 0.177x_2 - 0.325x_3 - 0.761x_4 + 0.144x_5 + 0.4x_6 - 0.128x_7 + 0.0375 \).

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 \( x_7 \) is very large then \( y = 0.051x_1 - 0.032x_2 - 0.149x_3 - 0.178x_4 + 0.155x_5 - 0.154x_6 - 0.047x_7 - 0.022 \).

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 \( x_7 \) is very big then \( y = 0.051x_1 - 0.032x_2 - 0.149x_3 - 0.178x_4 + 0.155x_5 - 0.154x_6 - 0.047x_7 - 0.022 \).

In this paper, we have proposed an approach for building an \( F^2 \)LS, using a Takagi-Sugeno fuzzy inference system in a federated setting. The \( F^2 \)LS 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 \( F^2 \)c-LS 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.

### VI. CONCLUSION

### REFERENCES

Fig. 3. Surface plots for local and federated models


