Self-adaptive Federated Swarm Intelligence

Deyi Zeng 2019051712

Abstract

Federated learning is recently proposed to protect data privacy while training based on all available datasets. In this report, we propose an integration scheme between swarm intelligence and federated learning. Compared to conventional swarm intelligence, we replace the conventional local optimum with a mutation factor. Also, a self-adaptive learning rate was proposed in this algorithm. We apply the federated swarm intelligence to Boston House Price Regression and Handwritten Digits Classification and verify our suggestion of adaptive learning rate.

1 Introduction

Suppose that there are some clients that hold data in their hands. These clients can be hospitals, Internet companies, and others holding data but could not upload the data to the cloud, due to privacy protection policies. This results in data isolation. But these clients want to cooperate to train a generalized model, which is easily completed when centralized learning is allowed. But due to data privacy, they cannot share the raw data with others, but upload some processed characteristics to the server. When the federated average (FPA) was first proposed [1], it suffered a lot from the non- independent identical data. FPA sends the model to clients, and clients could train the models on their local datasets and upload the parameters to server. The server will take weighted average of the parameters to be an updated model. When the data is non independently identically distributed in clients, the models are incompatible and fusion of parameters does not help with improving the accuracy. To solve this problem, two gradient correction models Fedprox [2] and SCAFFOLD [3] are proposed. These two models use correction and regularization terms to control the deviation caused by non independent identical distribution. Meanwhile, FedAMP [4] provides all clients with their personal models. These improve the local accuracy to a certain degreee. Fedfusion [5] decides to extract and combine some features of the neural network to investigate some deep relationships between layers and features.

Particle Swarm Optimization (PSO) [6] plays an important roles in optimization problems. PSO has been applied to solve Traveling Salesman Problems (TSP) [7] [8] .

When the data is non independent identical distributed in clients (For example, if there are four clients 'A','B','C','D' and 5 classes '1','2','3','4','5'. Client A only has samples in class '1', Client B only has samples in class '2', Client C only has samples in class '3', and Client D only has samples in class '4' and '5'.) integrate the model through parameters may not be a wise choice. Maybe, we can try to integrate the model in another way: Integrate the Loss. This formulates our idea. Initialize the model at the server, then sends the model to clients. Clients could use the model to make prediction ad return the client losses back to the server, then the server collects the losses and compute the global losses. Then, the server could use Particle Swarm Optimization to generate a new model. Repeat this process we could train a model without the raw data in clients.

2 Problem Statement

2.1 Regression Task: Boston House Price Prediction

The Boston House Price is correlated to some features. Given four clients each holding some samples, where a sample is characterized by 13 features $x_1, x_2, ..., x_{12}, x_{13}$ and corresponding price y. Clients are not willing to upload the raw data to server or share their data with each other due to protection of privacy. we are going to train the following multiple linear regression model by the federated swarm intelligence scheme and compare it with the benchmark (centralized learning):

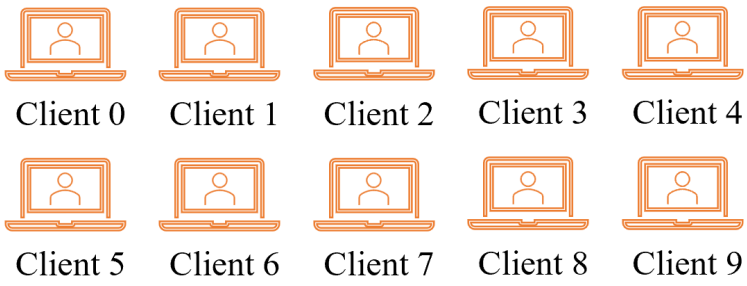
$$y_i = \theta_0 + \sum_{k=1}^{p} \theta_k x_{k,i} \qquad i \in \{1, 2, ..., m-1, m\}$$

2.2 Classification Task: MNIST Handwritten Digits Classification

Given some clients which hold the samples separately. The server want to fit a classification model such that the model can predict the samples, without getting in touch with the raw data. The samples are non-independently identically distributed. In this case, the Federated Average (FedAVG)

algorithm does not perform well. We would like to use Federated Swarm Intelligence for solving this problem. Minimize the Classification Error is our goal.

But we should notice that these samples are privately available to 10 clients. Under this circumstance, server is not allowed to request raw data from clients.



2.3 Frequent Asked Question

Q: What is raw data?

A: The raw data is the dataset. For example, in the Boston House Price, the matrix of features X and matrix of labels y are raw data. But if some θ is given, and you compute the MSE on your dataset. Then this MSE is not a raw data. It can be uploaded to the server.

Q: What is non independent identical distribution?

A: If all the clients hold balanced dataset, it is called independent identical distribution. If clients hold unbalanced dataset, it is called non independent identical distribution. For example, in Figure 1 all clients hold the same number of samples in each class. But in Figure 2, client 0 only has samples in class 0, and client 1 only has samples in class 1..... This kind of distribution is unbalanced. Therefore, we say that it is non independent identical distribution.

	No. of '0's	No. of '1's	No. of '2's	No. of '3's	No. of '4's	No. of '5's	No. of '6's	No. of '7's	No. of '8's	No. of '9's
Client 0	40	40	40	40	40	40	40	40	40	40
Client 1	40	40	40	40	40	40	40	40	40	40
Client 2	40	40	40	40	40	40	40	40	40	40
Client 3	40	40	40	40	40	40	40	40	40	40
Client 4	40	40	40	40	40	40	40	40	40	40
Client 5	40	40	40	40	40	40	40	40	40	40
Client 6	40	40	40	40	40	40	40	40	40	40
Client 7	40	40	40	40	40	40	40	40	40	40
Client 8	40	40	40	40	40	40	40	40	40	40
Client 9	40	40	40	40	40	40	40	40	40	40

Figure 1: Independent Identical Distribution in Clients

	No. of '0's	No. of '1's	No. of '2's	No. of '3's	No. of '4's	No. of '5's	No. of '6's	No. of '7's	No. of '8's	No. of '9's
Client 0	400	0	0	0	0	0	0	0	0	0
Client 1	0	400	0	0	0	0	0	0	0	0
Client 2	0	0	400	0	0	0	0	0	0	0
Client 3	0	0	0	400	0	0	0	0	0	0
Client 4	0	0	0	0	400	0	0	0	0	0
Client 5	0	0	0	0	0	400	0	0	0	0
Client 6	0	0	0	0	0	0	400	0	0	0
Client 7	0	0	0	0	0	0	0	400	0	0
Client 8	0	0	0	0	0	0	0	0	400	0
Client 9	0	0	0	0	0	0	0	0	0	400

Figure 2: Non Independent Identical Distribution in Clients

3 Data Processing

3.1 Boston House Price

Dataset Portal: https://storage.googleapis.com/tensorflow/tf-keras-datasets/boston_housing.npz.

In the linear regression problem without regularization, we are not required to standardize the features. In the Boston House Price Dataset, we have 506 samples, and we divide them into training dataset containing 404 samples and test datasets containing 102 samples. For each sample, there are 13 features. To simulate the cases the server cannot access data from clients and data is separately held by several clients, we just divide the training dataset into 4 datasets. So in this case, 4 clients hold 101 samples each.

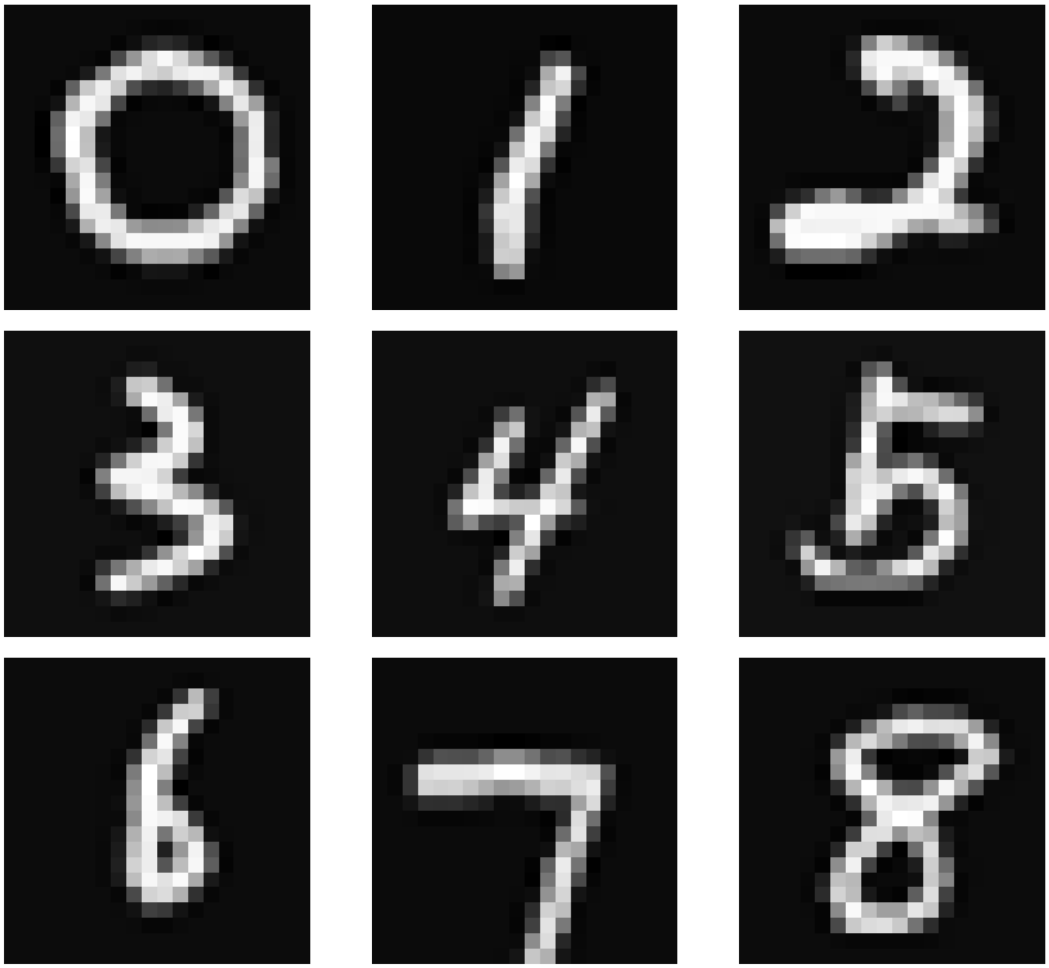
	0	1	2	3	4	5	6	7	8	9	10	11	12
0	1.23247	0.0	8.14	0.0	0.5380	6.142	91.7	3.9769	4.0	307.0	21.0	396.90	18.72
1	0.02177	82.5	2.03	0.0	0.4150	7.610	15.7	6.2700	2.0	348.0	14.7	395.38	3.11
2	4.89822	0.0	18.10	0.0	0.6310	4.970	100.0	1.3325	24.0	666.0	20.2	375.52	3.26
3	0.03961	0.0	5.19	0.0	0.5150	6.037	34.5	5.9853	5.0	224.0	20.2	396.90	8.01
4	3.69311	0.0	18.10	0.0	0.7130	6.376	88.4	2.5671	24.0	666.0	20.2	391.43	14.65
...	...	...	...	...	...	...	...	...	...	...	...	...	...

3.2 Handwritten Digits Classification

Dataset Portal: https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz.

We have a total of 5000 samples, 500 samples for each class. Each sample is a length-400 vector. We choose 100 samples from each class to form the test dataset. To simulate a worst case in Federated Learning case, that is, there are ten clients who hold only one class of samples. That is: client A only holds 400 '0's, while client B only holds 400 '1's.....

According to this, we create ten clients and allocate 400 samples with the same labels to the same client, as displayed in Figure 2.



4 Model Overview

Server initializes the model . Then sends the model to all clients (Figure 3). Clients compute the losses at local space (Figure 4). Clients then send client losses back to the server (Figure 5). The server computes the global loss from client losses and implement swarm intelligence (Figure 6). Server sends the new model to clients (Figure 3), which forms a loop. For details, refer to the Algorithm Chapter.

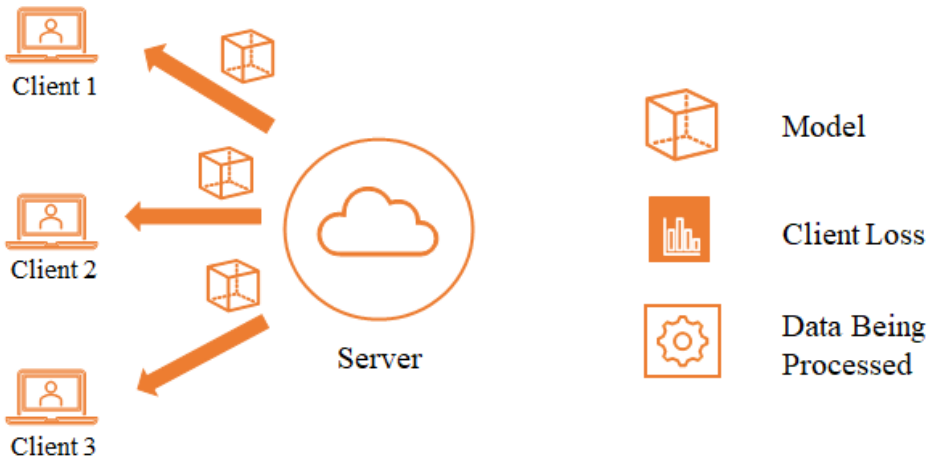


Figure 3: Server Sends Model to Clients

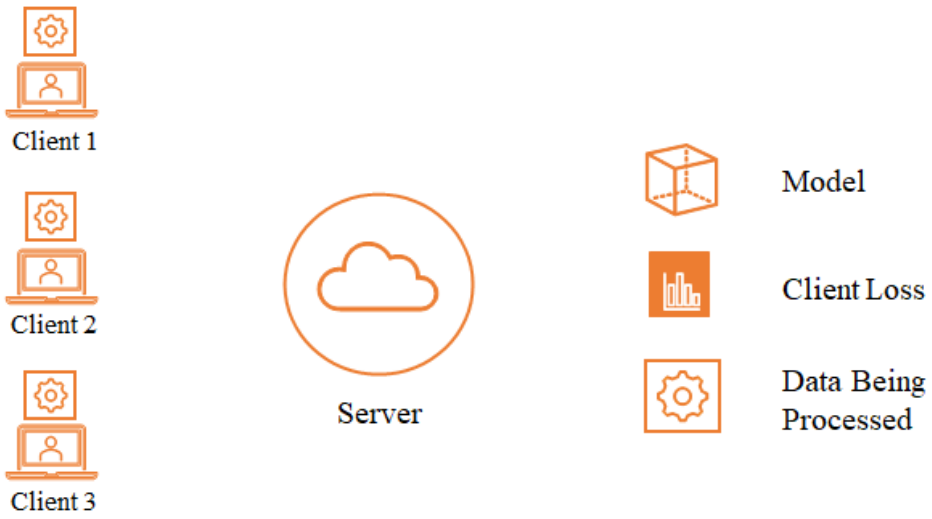


Figure 4: Clients compute Client Losses

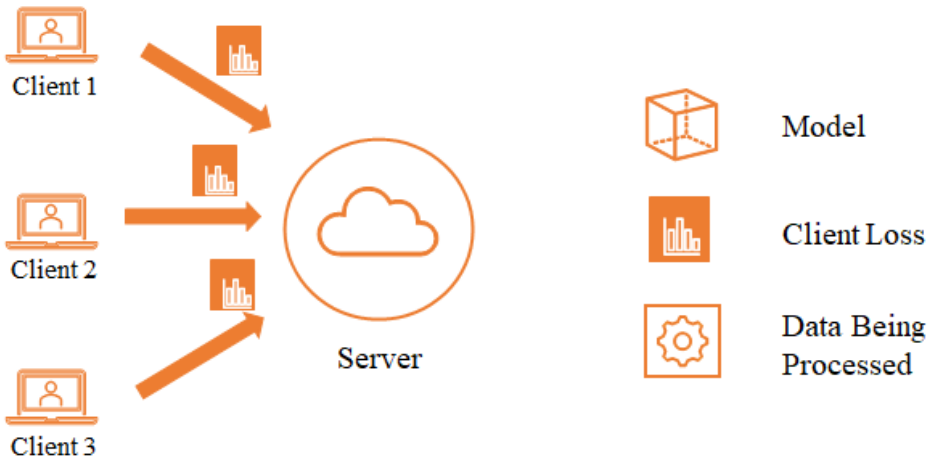


Figure 5: Clients Send Client Losses to Server

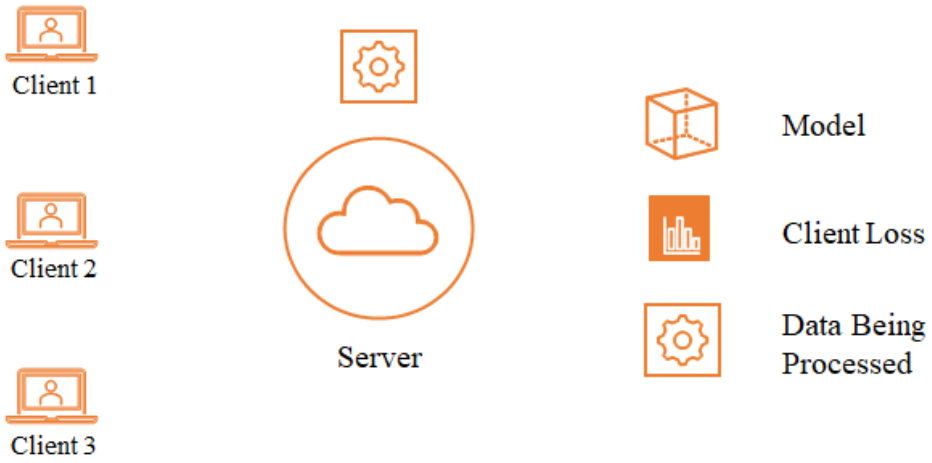


Figure 6: Server Computes Global Loss and Implements Swarm Intelligence

5 Modeling of Loss Function

5.1 Regression Model

We only require one linear model for this problem.


Linear Model

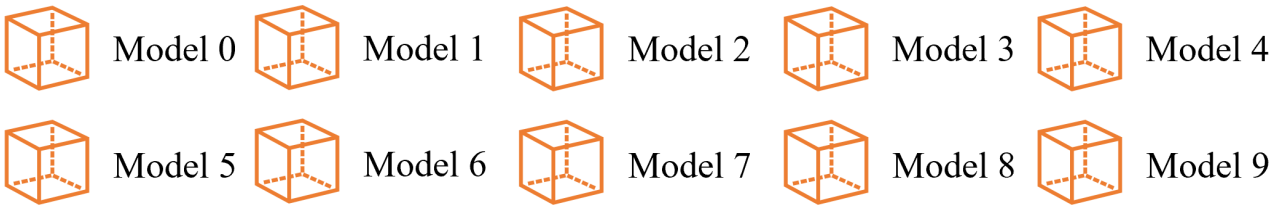
$$y_i = \theta_0 + \sum_{k=1}^{p} \theta_k x_{k,i} \qquad i \in \{1, 2, ..., m-1, m\} \tag{1}$$

In this linear regression model, we can define the global loss and client loss (also known as local loss).

$$\begin{aligned}
Global\ Loss: \quad MSE(\theta) =& \frac{1}{J} \sum_{i=1}^{m} (\theta_0 + \sum_{k=1}^{p} \theta_k x_{k,i} - y_i)^2 \\
=& \sum_{j=1}^{J} \frac{l_j}{J} \frac{1}{l_j} \sum_{i \in C_j} (\theta_0 + \sum_{k=1}^{p} \theta_k x_{k,i} - y_i)^2 \\
=& \sum_{j=1}^{J} \frac{l_j}{J}\ MSE_j(\theta) \\
Client\ Loss: \quad MSE_j(\theta) =& \frac{1}{l_j} \sum_{i \in C_j} (\theta_0 + \sum_{k=1}^{p} \theta_k x_{k,i} - y_i)^2
\end{aligned}$$

5.2 Classification Model

We may use logistic regression here. We could train 10 models, and use one versus all scheme. That is, we could train 10 models. The first model predicts whether a sample is '0', and returns the probability to be '0'. The second model predicts whether a sample is '1', and returns the probability to be '1'.....Same for the remaining models. Then for a specific sample, it has 10 probabilities given by 10 models. We select the highest probability and the model generating this probability is its predicted label. This is the 'one-versus-all' technique.



Model y predicts the probability that the sample is in the class y

For example, for a sample X_i, the model 0 predicts $P(y_i = 0|X_i) = 0.30$, the model 1 predicts $P(y_i = 1|X_i) = 0$, the model 2 predicts $P(y_i = 2|X_i) = 0$, the model 3 predicts $P(y_i = 3|X_i) = 0$, the model 4 predicts $P(y_i = 4|X_i) = 0.99$, the model 5 predicts $P(y_i = 5|X_i) = 0.98$, the model 6 predicts $P(y_i = 6|X_i) = 0$, the model 7 predicts $P(y_i = 7|X_i) = 0.24$, the model 8 predicts $P(y_i = 8|X_i) = 0$, the model 9 predicts $P(y_i = 9|X_i) = 0.4$. Then $\hat{y}_i = 4$. The y_i is predicted to be 4, since it has the highest probability among ten models.

For each model, we should train them separately. Now we define the model, global loss and client loss for a single model (Binary Cross Entropy):

$$
\begin{aligned}
&y_i^{(t)} = 1, \quad \text{if} \quad t = y_i \\
&y_i^{(t)} = 0, \quad \text{if} \quad t \neq y_i \\
&P(y_i = t|X_i) = P(y_i^{(t)} = 1|X_i) = h(X_i, \theta^{(t)}) = \frac{1}{1 + e^{-(\theta_0^{(t)} + \sum_{k=1}^{p} \theta_k^{(t)} x_{k,i})}} \\
&y_i = argmax(P(y_i = t|X_i)), t \in \{0, 1, ..., 8, 9\}, i \in \{1, 2, ..., m-1, m\}
\end{aligned}
\tag{2}
$$

$$
\begin{aligned}
Global\ Loss: \quad BCE(\theta^{(t)}) &= -\sum_{i=1}^{m} [y_i^{(t)} \cdot ln(h(X_i, \theta^{(t)})) + (1 - y_i^{(t)}) \cdot ln(1 - h(X_i, \theta^{(t)}))] \\
&= -\sum_{j=1}^{J} \sum_{i \in C_j} [y_i^{(t)} \cdot ln(h(X_i, \theta^{(t)})) + (1 - y_i^{(t)}) \cdot ln(1 - h(X_i, \theta^{(t)}))] \\
&= \sum_{j=1}^{J} BCE_j(\theta^{(t)}) \\
Client\ Loss: \quad BCE_j(\theta^{(t)}) &= -\sum_{i \in C_j} [y_i^{(t)} \cdot ln(h(X_i, \theta^{(t)})) + (1 - y_i^{(t)}) \cdot ln(1 - h(X_i, \theta^{(t)}))]
\end{aligned}
$$

5.3 Computation of Global Loss

In the above formula, we transform the computation of global loss as a linear combination of local loss. So the idea comes up. In the Particles Swarm Algorithm, fitness(in these problems Global Loss) is required. Then in the federated Swarm Intelligence, the server could send the parameters θ to clients, and the local clients could compute their client loss and return these back to the server. The server can integrate the client losses to have the global loss.

6 Modeling of Federated Swarm Intelligence

We may not use the conventional particle swarm algorithm. Some modification is required. In this problem, we drop out the local optimum: 'pbest'. But we introduce a new random factor(mutation factor), which describes that the motion of a particle may not rigorously follow the direction to gbest. They have their own thinking pattern. This enlarges the search area for solutions.

Then we may give the formula or iteration here:

$$
\begin{aligned}
&\theta_s = \theta_s + v_s \\
&v_s = c_1 \cdot v_s + c_2 \cdot rand_1^{p+1} \cdot (gbest - \theta_s) + c_3 \cdot rand_2^{p+1} \\
&\text{where } c_1 \geq 0, \quad c_2 \geq 0, \quad c_3 \geq 0 \\
&gbest = argmin(Global\ Loss(\theta_s)) \quad s \in \{1, 2, ..., S-1, S\} \\
&rand_1^{p+1} \in [0, 1]^{p+1} \\
&rand_2^{p+1} \in [-1, 1]^{p+1}
\end{aligned}
\tag{3}
$$

We can have one more step, to transform the problem into the following form:

$$
\begin{aligned}
&\theta_s = \theta_s + v_s \\
&v_s = \alpha(w_1 \cdot v_s + w_2 \cdot rand_1^{p+1} \cdot (gbest - \theta_s) + (1 - w_1 - w_2) \cdot rand_2^{p+1}) \\
&\text{where} \quad \alpha \geq 0, \quad 0 \leq w_1 \leq 1, \quad 0 \leq w_2 \leq 1 \\
&gbest = argmin(Global\ Loss(\theta_s)) \quad s \in \{1, 2, ..., S-1, S\} \\
&rand_1^{p+1} \in [0, 1]^{p+1} \\
&rand_2^{p+1} \in [-1, 1]^{p+1}
\end{aligned}
\tag{4}
$$

During the design of the algorithm part, we will give a design self-adaptive α. This is the reason why we want to draw the α out here. Indeed, it represents the learning rate or stride of the update.

7 Algorithm

The server should initialize a particle swarm of coefficients θ and velocity v, where the number of particles is denoted by S. The velocity should be initialized as a zero vector. This is because any other initialization misleads evolution.

Then, the server sends these particles θ to the clients. You need to notice that, the θ is usually a matrix recording S number of particles. For example, if you use 20 particles and 35 features, you

will have a matrix of shape 36×20. In the Boston House Price case, there are 13 features and the population of particles are 20. So the θ matrix has a shape 14×20. In the MNIST Classification case, there are 400 pixels (features), and we use 20 particles. Then the θ matrix has a shape 401×20.

When clients receive the θ, they can compute the corresponding mean square error(MSE) and binary cross entropy (BCE) locally, and send it back to the server.

The server collects client loss from the clients and computes the global loss. Then, the server could compute the global loss of each particle.

gbest can be obtained from the global loss. Then we randomly generate two random numbers rand1 and rand2.

We introduce the learning rate α here. Since the loss function, MSE and BSE (Binary Crossentropy) are convex functions. There is only one minimum. If the gbest does not change after several epochs, then update $\alpha = 0.5\alpha$, while gbest changes then update $\alpha = 2\alpha$. This is because if the gbest does not change for several epochs, the search space is far from the optimum. And if the gbest changes, it means that the search space can be enlarged and we can approach the minimum at a faster speed.

Update the velocity, and update the particles. Then send the new particles θ to each clients. Clients send their client losses back to server, and the server computes global losses of each particle. For a specific particle θ_s, the server compares the new global loss and old global loss. If the new global loss is lower than the old global loss, it implies that this specific particle has been improved. Then we should renew the particle θ_s. If the new global loss is higher than the old global loss. This implies that this specific particle has not been improved. Then the particle should not be changed.

After this, server send the updated particles to clients again. This forms the loop.

Symbol	Interpretation	Symbol	Interpretation
α	Learning Rate	MSE	Mean Square Error
θ	Parameters of Model	BCE	Binary Crossentropy
$\theta^{(t)}$	Parameters of Model t	v	Velocity of Motion
J	Number of Clients	l_j	Number of Samples in Clients j
S	Population of Particles	$gbest$	Global Optimum

Algorithm 1 Federated Swarm Intelligence: Linear Regression

Initialize: α, θ, v, J, S,l_j, max_epoch

for epoch $\leftarrow 0 :$ max_epoch **do**
  MSE_old = 0
  for $j \leftarrow 1 : J$ **do**
    send θ to clients
    clients **return** MSE(j)
    MSE_old = MSE_old + l_j/J * MSE(j)
  end for
  if MSE_old doesn't change for several steps **then**
    $\alpha = \alpha/2$
  else if MSE_old changes in last step **then**
    $\alpha = \alpha * 2$
  end if
  gbest = θ(argmin(MSE_old))
  $v = \alpha(w_1 * v + w_2 * rand_1^{p+1}(gbest - \theta) + (1 - w_1 - w_2) * rand_2^{p+1})$
  temp = $\theta + v$
  MSE_new = 0
  for $j \leftarrow 1 : J$ **do**
    send temp to clients
    clients **return** MSE(j)
    MSE_new = MSE_new + l_j/J * MSE(j)
  end for
  for $s \leftarrow 1 : S$ **do**
    if MSE_new(s) $<$ MSE_old(s) **then**
      $\theta(s)$ = temp(s)
    end if
  end for
end for

Algorithm 2 Federated Swarm Intelligence: Multi-class Classification

Initialize: α, $\theta^{(t)}$, v, J, S, max_epoch

for t $\leftarrow$ 0 : 9 **do** ▷ Digits 0 to 9
  for epoch $\leftarrow$ 0 : max_epoch **do**
    BCE_old = 0
    for $j \leftarrow 1 : J$ **do**
      send $\theta^{(t)}$ to clients
      clients **return** BCE(j)
      BCE_old = BCE_old + BCE(j)
    end for
    if BCE_old doesn't change for several steps **then**
      $\alpha = \alpha/2$
    else if BCE_old changes in last step **then**
      $\alpha = \alpha * 2$
    end if
    gbest = $\theta^{(t)}$(argmin(BCE_old))
    $v = \alpha(w_1 * v + w_2 * rand_1^{p+1}(gbest - \theta^{(t)}) + (1 - w_1 - w_2) * rand_2^{p+1})$
    temp = $\theta^{(t)} + v$
    BCE_new = 0
    for $j \leftarrow 1 : J$ **do**
      send temp to clients
      clients **return** BCE(j)
      BCE_new = BCE_new + BCE(j)
    end for
    for $s \leftarrow 1 : S$ **do**
      if BCE_new(s) $<$ BCE_old(s) **then**
        $\theta^{(t)}(s)$ = temp(s)
      end if
    end for
  end for
end for

8 Results

8.1 Boston House Price

Since the α is self-adaptive and not sensitive to the initial value. We will not focus on the α at the first stage. We analyze the performance of the algorithm under different w_1 and w_2. We can use normal equations to solve the centralized linear regression problem as the benchmark. Then compare the results given by Federated Swarm Intelligence with this centralized method. The result is compared in Table 1. We can see that the benchmark value is obtained in several settings: $w_1 = 0, w_2 = 1.0$; $w_1 = 0.2, w_2 = 0.8$;$w_1 = 0.4, w_2 = 0.6$. These three settings eliminate the mutation terms in the modified swarm intelligence. But when $w_1 = 0, w_2 = 0$, the algorithm also performs well. We should observe the evolution plot of these values.

Table 1: test MSE after 10^6 epochs (Benchmark: $MSE = 23.1956$)

w_1 \ w_2	0	0.2	0.4	0.6	0.8	1.0
0	24.7145	33.1310	38.0039	32.6774	39.5274	23.1956
0.2	32.4099	39.0483	41.0442	36.5248	23.1956	
0.4	24.5342	40.3318	34.2974	23.1956		
0.6	40.7602	40.9808	83.6202			
0.8	28.9566	149.6375				
1.0	22332.6867					

Evolution of part of the results is also displayed in Figure 7. The recording begins after 10^5 iterations. We observe that when $w_1 = 0, w_2 = 1.0$, the algorithm has already converged before the recording. And we can also observe a steady decrease when $w_1 = 0, w_2 = 0$. Notice that the algorithm in this setting relies completely on mutation terms. This will be very helpful if the particles all converge to a suboptimum (suboptimum is not necessarily the local optimum), and the algorithm depends on the mutation to find a better value (jump out of the trap).

We also want to know whether our adaptive scheme works. So we compare the adaptive algorithm with fixed algorithm (Figure 8). By setting a fixed α from 0.0005 to 0.1, we discover that an adaptive α performs the best among all. This verifies our design.

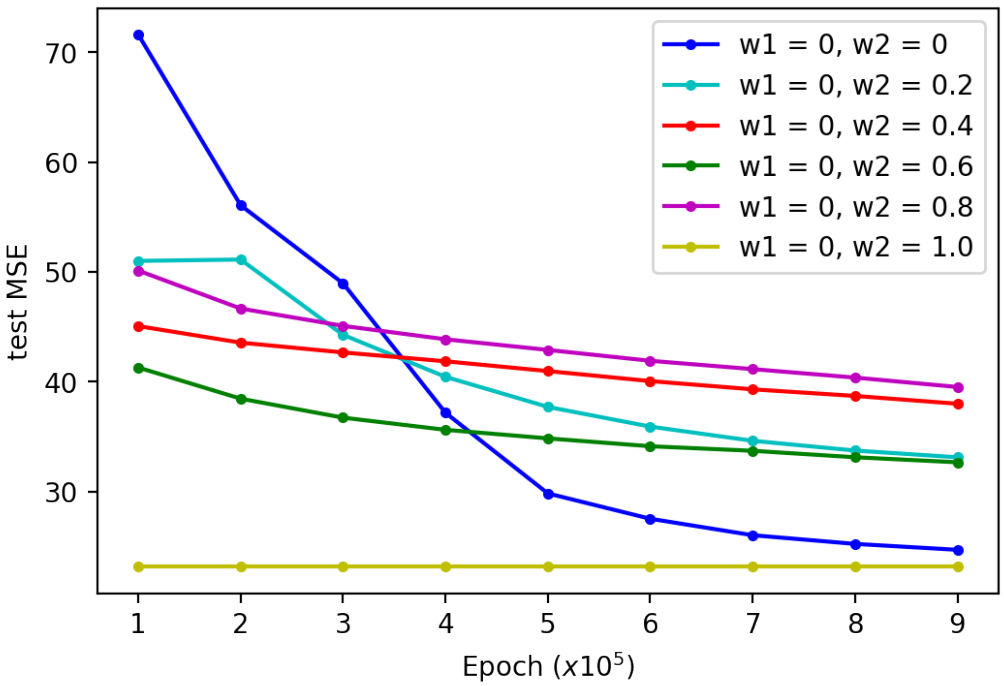


Figure 7: Test MSE versus epoch (Adaptive α)

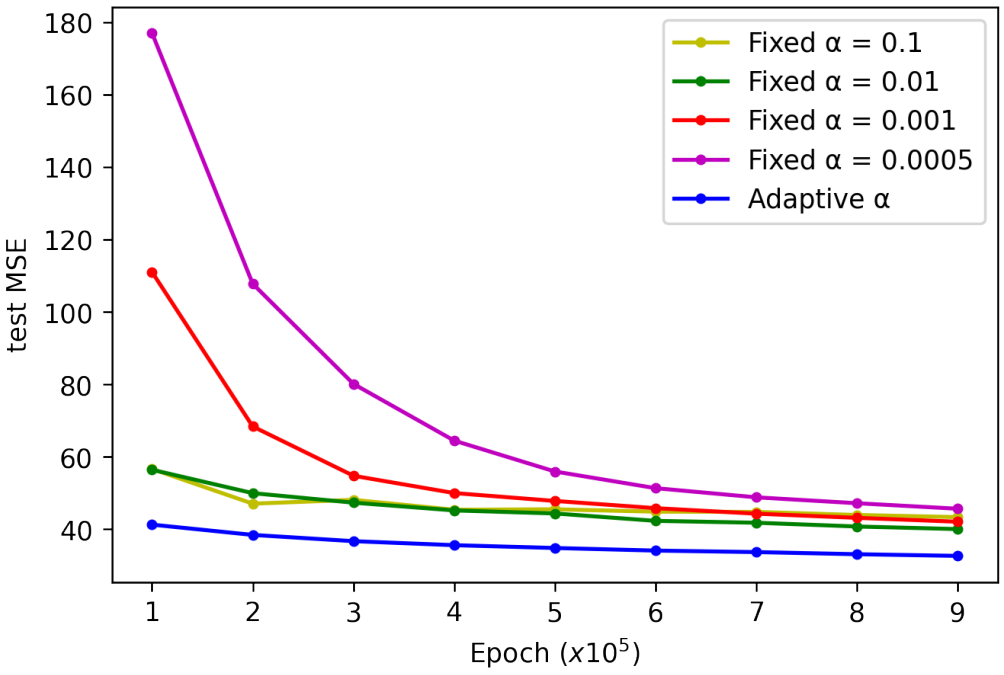


Figure 8: Test MSE versus epoch ($w_1 = 0, w_2 = 0.6$)

8.2 Handwritten Digit Classification

Figure 9 is the confusion matrix of the applications of federated swarm intelligence on the MNIST handwritten test dataset. We can observe that the accuracy on digits 5' and '9' are relatively low, where '5's are always recognized as '8's and '9's are recognized as '7'. To separate them apart, we can choose to use a more complicated model(like including higher power and interactive terms into logistic regression) or use a neural network instead. The overall accuracy is 80.2%.

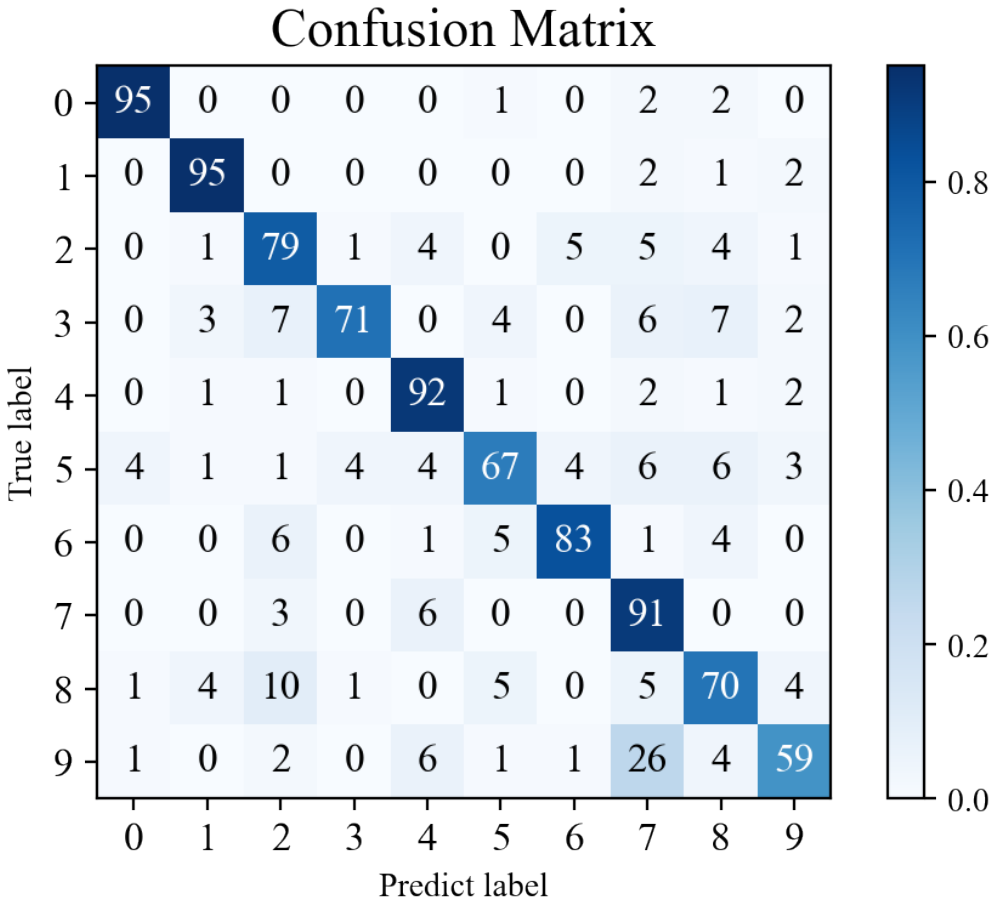


Figure 9: Confusion of Test Dataset (100 samples for each class)

The setting is $w_1 = 0, w_2 = 0.3$ and max epoch = 1500. The training process is displayed in the Figure 10. We can observe that model 0 and model 1 converge swiftly to very small losses. This implies that the model 0 and 1 are versed in distinguishing 0 and 1 from other digits, as also indicated in the confusion matrix (The accuracy of predicting '0' and '1' are both 95%).

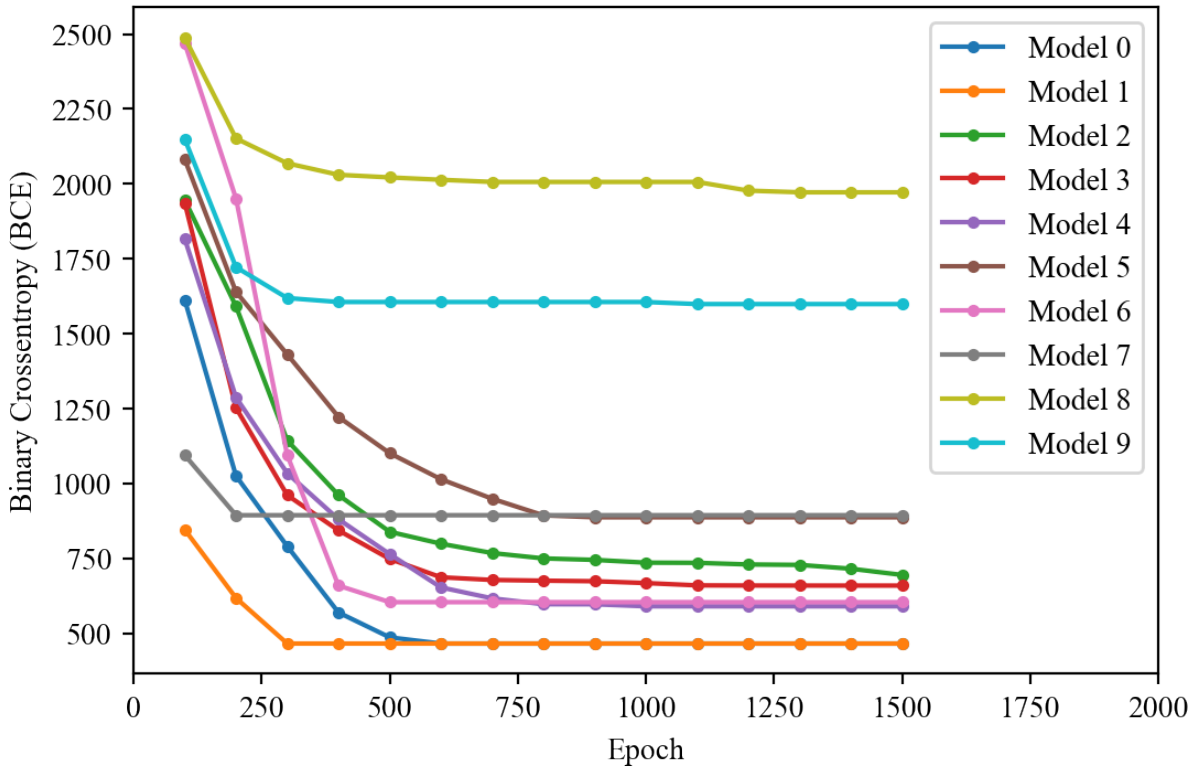


Figure 10: Training Process of Models

9 Conclusion

In the regression model, we prove the effectiveness of Federated Swarm Intelligence. And conduct sensitivity analysis by comparing different combinations of hyper parameters. We prove that the Federated Swarm Intelligence can converge to the Benchmark case. That is, in linear regression task, the result of Federated Swarm Intelligence can be the same as the result obtained when all data can be trained in the same machine. Also, we prove that the self-adaptive scheme of learning rate indeed accelerates the convergence. In the classification model, we apply the Federated Swarm Intelligence to Logistic Regression Model. By using one versus all scheme, we fit a model that predicts the test dataset with a 80.2% accuracy.

The Federated Swarm Intelligence is robust to non independent identical distribution. This is because whatever distribution the clients have, if all the samples are the same, the global loss is radically the same. This is a big difference compared to the conventional federated learning, as in the conventional federated learning, the clients send their personal models to server, which is sensitive to the distribution of data, while in the federated swarm intelligence, the clients only send the client

losses to server, which is robust to the distribution of data.

10 Originality

- We propose a method of implementing the Particle Swarm Algorithm in the Federated Learning Case, and apply it to both regression and classification tasks.
- We design an efficient self-adaptive learning rate in the Particle Swarm Algorithm.
- The designed Federated Swarm Intelligence is robust to non-independent identical distribution.

11 Appendix A: Parameters of Linear Regression Model

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	40.293629	-0.119997	0.057	0.003984	4.126981	-20.500245	3.38025	0.007568	-1.711897	0.334748	-0.01178	-0.902317	0.008719	-0.555843

12 Appendix B: Parameters of Multi-class Classification Model

	Model0	Model1	Model2	Model3	Model4	Model5	Model6	Model7	Model8	Model9
0	-87.974997	1.329816e+24	-115.445438	-577.996080	-73.491337	20.499590	-1.605185e+49	-22.809520	-306.496682	-116.273596
1	-148.059745	-1.610100e+26	-43.721117	-86.679586	-157.529651	-87.300538	-2.559669e+48	-3.275666	88.706794	-13.870228
2	-24.542126	-2.697142e+26	-39.277256	-65.218391	-35.519629	24.407907	-3.607035e+46	-29.810791	6.045927	214.492259
3	38.007672	-5.252096e+25	-79.718558	-398.734940	124.207376	23.100396	3.862628e+49	-2.125386	53.851414	-46.911610
4	264.272959	3.484403e+28	-46.061975	-719.183549	28.037741	44.393563	4.753859e+48	1.478790	87.988012	-38.545421
...	...	...	...	...	...	...	...	...	...	...
396	-229.156734	6.774906e+25	-14.467895	-202.312512	-24.851759	27.593614	2.183863e+50	-24.455009	13.204998	56.344923
397	-129.049576	-1.027579e+26	-12.615918	103.112001	-4.642655	65.365735	1.545500e+47	-9.033208	1.670176	-31.681482
398	-368.190178	2.406200e+24	53.890324	8.283381	-57.903348	-12.463101	2.928140e+52	37.182749	76.067670	-108.638196
399	381.352570	-1.030189e+24	-14.106621	-343.707777	-40.463505	-23.909395	3.143963e+53	-27.063175	87.835068	-43.241226
400	136.456735	1.685648e+26	22.346777	3.448867	-19.502704	42.959241	-2.592308e+48	16.397453	-36.941642	31.008506

401 rows × 10 columns

Full table Portal: https://github.com/StellaVadis/Result-FSI-Classification

References

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[2] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

[3] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[4] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on non-iid data. In *AAAI*, pages 7865–7873, 2021.

[5] Xin Yao, Tianchi Huang, Chenglei Wu, Ruixiao Zhang, and Lifeng Sun. Towards faster and better federated learning: A feature fusion approach. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 175–179. IEEE, 2019.

[6] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm intelligence*, 1(1):33–57, 2007.

[7] Kang-Ping Wang, Lan Huang, Chun-Guang Zhou, and Wei Pang. Particle swarm optimization for traveling salesman problem. In *Proceedings of the 2003 international conference on machine learning and cybernetics (IEEE cat. no. 03ex693)*, volume 3, pages 1583–1585. IEEE, 2003.

[8] Xiaohu H Shi, Yanchun Chun Liang, Heow Pueh Lee, C Lu, and QX Wang. Particle swarm optimization-based algorithms for tsp and generalized tsp. *Information processing letters*, 103(5):169–176, 2007.