Asynchronous Online Federated Learning for Edge Devices with Non-IID Data

데이터 응용 및 관리 연구실
학부생 연구원 김민선

BACKGROUND

Fed-AVG Algorithm

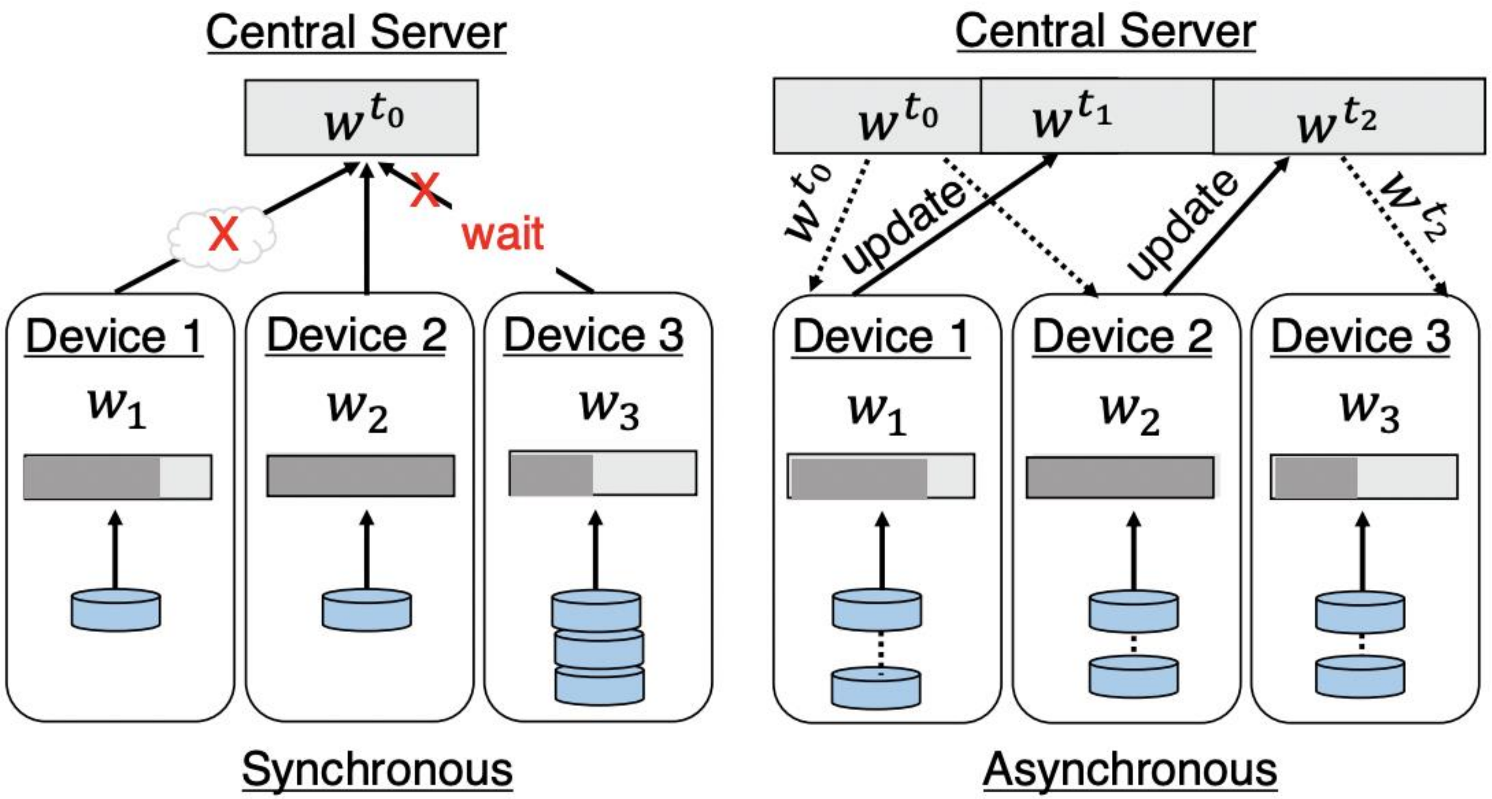
$$f_k(w_k) \overset{def}{=} \frac{1}{n_k} \sum_{i \in \mathcal{D}_k} \ell_i(x_i, y_i; w_k). \tag{1}$$

$$F(w) = \sum_{k=1}^{K} \frac{n_k}{N} f_k(w). \tag{2}$$

$$w_* = \arg\min F(w). \tag{3}$$

INTRODUCTION

ASO-Fed



SYNCHRONIZED FEDERATED OPTIMIZATION

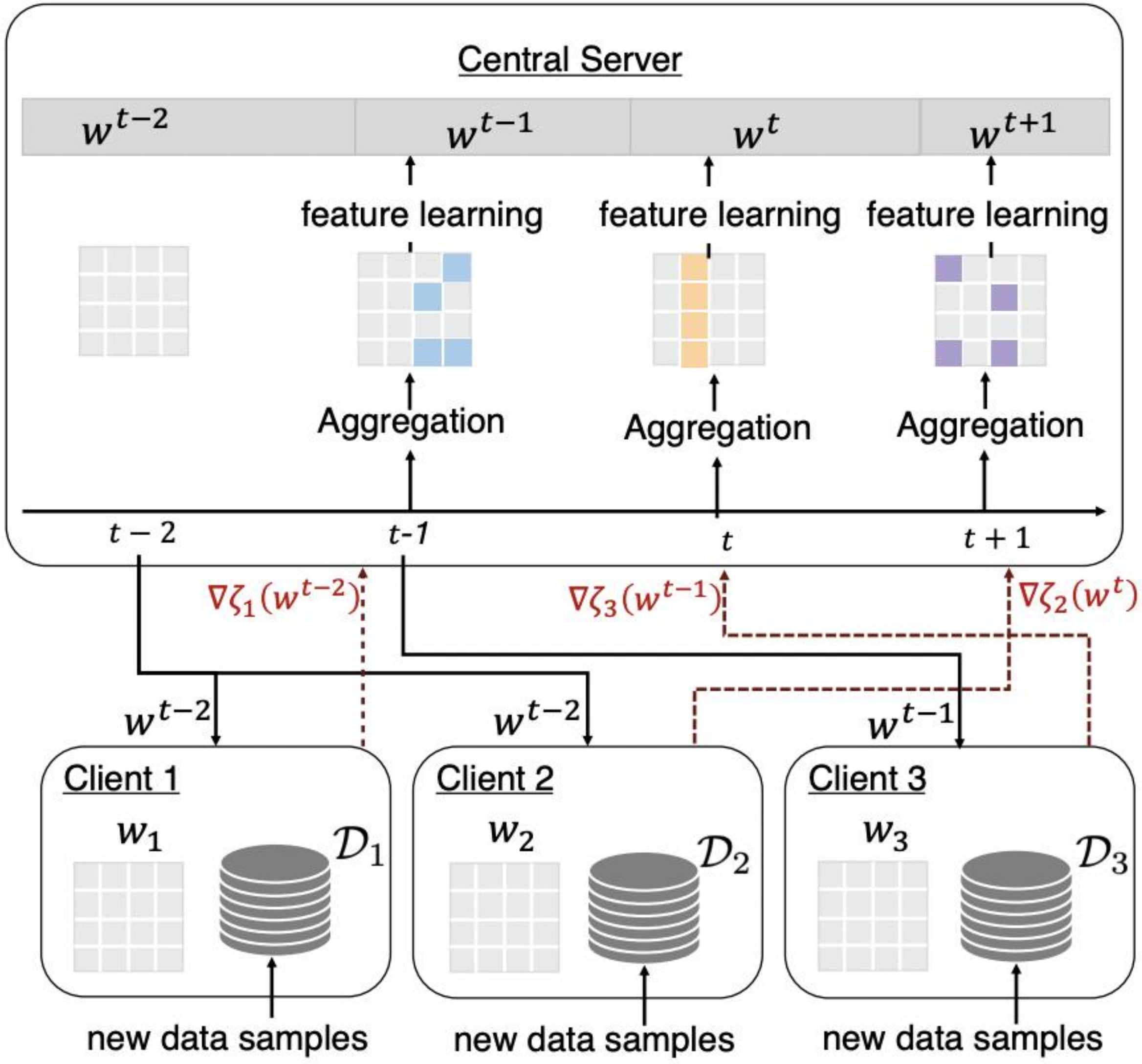
1. *Large epochs -> towards the local objective optima*
2. *Continuous data -> the increase of local gradient variations*

Algorithm 1 Algorithm for FedAvg

Input: K indexed by k, local minibatch size B, local epochs E and learning rate η.
Central Server:
for global iterations $t = 1, 2, ..., T$ **do**
    Server chooses a subset S_t of K devices at random
    for each client $k \in S_t$ in parallel **do**
        $w_k^{t+1} \leftarrow \text{ClientUpdate}(k, w^t)$
    $w^{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{N} w_k^{t+1}$
ClientUpdate(k, w^t)**:**
device k updates w^t for E epochs of SGD on f_k with η
return w_k^{t+1} to server

PROPOSED METHOD

1. *Decay Coefficient*
2. *Global feature representation (Dynamic Step Size)*



CENTRAL SERVER

Overall Training Procedure

$$
\begin{aligned}
w^{t+1} &= w^t - \frac{n'_k}{N'}(w_k^t - w_k^{t+1}) \\
&= w^t - \frac{n'_k}{N'}(w_k^t - (w_k^t - \eta_k^t \nabla \zeta_k(w^t))) \\
&= w^t - \eta_k^t \frac{n'_k}{N'} \nabla \zeta_k(w^t).
\end{aligned} \tag{4}
$$

Feature Representation Learning on Server

$$
\alpha_{(1)}^{t+1}[i,j] \leftarrow \frac{\exp(|w_{(1)}^{t+1}[i,j]|)}{\sum_j \exp(|w_{(1)}^{t+1}[i,j]|)}, \tag{5}
$$

$$
w_{(1)}^{t+1}[i,j] = \alpha_{(1)}^{t+1}[i,j] * w_{(1)}^{t+1}[i,j]. \tag{6}
$$

LOCAL CLIENTS

Gradient-Based Update

$$s_k(w_k) = f_k(w_k) + \frac{\lambda}{2}||w_k - w||^2. \tag{7}$$

Local Update with Decay Coefficient

$$\nabla\zeta_k \leftarrow \nabla s_k - \nabla s_k^{(\text{pre})} + h_k^{(\text{pre})}, \tag{8}$$

$$h_k^{(\text{pre})} = \beta h_k^{(\text{pre})} + (1-\beta)\nabla s_k^{(\text{pre})}. \tag{9}$$

Model update

$$\begin{aligned} w_k^{t+1} &= w_k^t - \eta_k^t \nabla\zeta_k(w^t) \\ &= w_k^t - \eta_k^t\Big(\nabla f_k(w_k^t) - \nabla s_k^{(\text{pre})} + h_k^{(\text{pre})} + \lambda(w_k^t - w^t)\Big). \end{aligned} \tag{10}$$

Algorithm 2 Algorithm for ASO-Fed

```
Input: Multiple related learning clients distributed at client devices, regularization parameter λ, multiplier r_k, learning rate η̄, decay coefficient β.
Initialize: h_k^pre = h_k = 0, v_k = 0
Procedure at Central Server
for global iterations t = 1, 2, ..., T do
    /* get the update on w^t */
    compute w^t                                  ▷ [Eq.(4)]
    update w^t with feature learning             ▷ [Eq.(5) - Eq.(6)]
end for
Procedure of Local Client k at round t
receive w^t from the server
Compute ∇s_k
Set h_k^(pre) = h_k
Set ∇ζ_k ← ∇s_k − ∇s_k^(pre) + h_k^(pre)        ▷ [Eq.(7) -Eq.(10)]
Update w_k^{t+1} ← w_k^t − r_k^t η̄ ∇ζ_k
Compute and update h_k = βh_k + (1 − β)v_k
Update v_k = ∇s_k(w^t; w_k^t)
upload w_k^{t+1} to the server
```

LOCAL CLIENTS

Dynamic Learning Step Size

$$w_k^{t+1} = w_k^t - \eta_k^t \nabla \zeta_k(w^t) \tag{10}$$

$$w_k^{t+1} = w_k^t - r_k^t \bar{\eta} \nabla \zeta_k(w^t). \tag{11}$$

$$r_k^t = \max\{1, \log(\bar{d}_k^t)\}, \text{ where } \bar{d}_k^t = \frac{1}{t} \sum_{\tau=1}^{t} d_k^\tau$$

EXPERIMENT

TABLE IV.1

PREDICTION PERFORMANCE COMPARISON. BOLD NUMBERS ARE THE BEST PERFORMANCE, NUMBERS WITH UNDERLINES ARE THE SECOND BEST VALUES. IMPROV.(1) SHOWS THE PERCENTAGE IMPROVEMENT OF ASO-FED OVER FEDAVG. IMPROV.(2) SHOWS THE PERCENTAGE IMPROVEMENT OF ASO-FED OVER THE BEST BASELINE RESULTS.

Method	FitRec				Air Quality		ExtraSensory				Fashion-MNIST
	MAE ↓ (Speed)	SMAPE↓ (Speed)	MAE↓ (HeartRate)	SMAPE↓ (HeartRate)	MAE↓	SMAPE↓	F1↑	Precision↑	Recall↑	BA↑	Accuracy↑
FedAvg	13.61	0.78	13.72	0.78	44.30	0.44	0.66	0.87	0.55	0.77	0.87
FedProx	14.21	0.82	14.53	0.83	44.30	0.44	0.67	0.82	0.57	0.77	0.88
FedAsync	13.56	0.78	13.67	0.78	37.98	0.43	0.72	0.84	0.65	0.82	0.90
Local-S	12.76	0.75	13.27	0.76	36.72	0.56	0.65	0.72	0.61	0.79	0.89
Global	12.95	0.78	12.79	0.79	37.61	0.44	**0.77**	**0.92**	0.66	0.83	0.92
ASO-Fed(-D)	12.46	0.74	12.51	0.75	37.13	0.43	0.76	0.88	0.69	**0.85**	0.94
ASO-Fed(-F)	12.62	0.76	12.71	0.76	37.72	0.43	0.75	0.86	0.68	0.84	0.94
ASO-Fed	**12.31**	**0.73**	**12.36**	**0.74**	**36.71**	**0.42**	**0.77**	0.88	**0.70**	**0.85**	**0.95**
improv.(1)	9.55%	6.41%	9.91%	5.13%	17.13%	2.32%	16.66%	1.15%	27.27%	10.39%	9.19%
improv.(2)	3.52%	2.67%	3.36%	2.63%	0.03%	4.54%	0.00%	-4.34%	6.06%	2.41%	3.26%

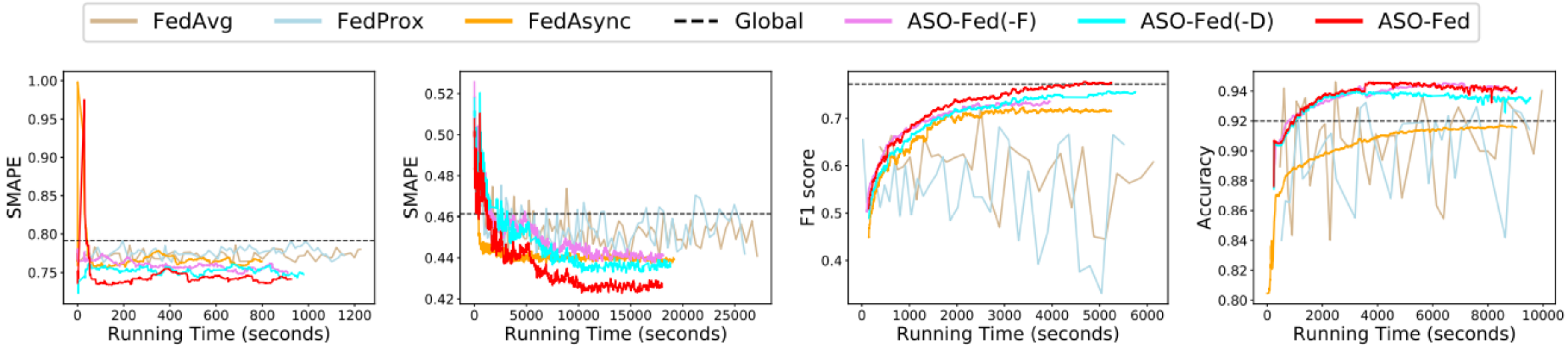
Datasets:

1. FitRec

2. Air Quality

3. ExtraSensory

4. Fashion MNIST



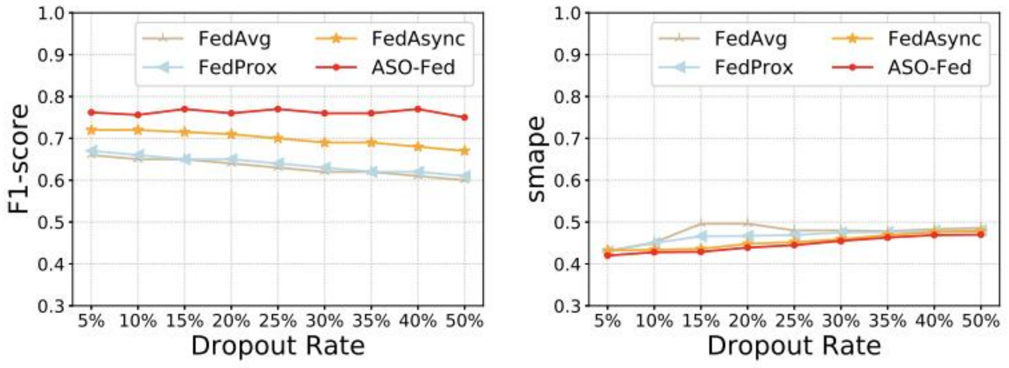
(a) FitRec (SMAPE ↓) (b) Air Quality (SMAPE ↓) (c) ExtraSensory (F1-score ↑) (d) Fashion-MNIST (Accuracy ↑)

Fig. 3. Test set performance vs. running time for four datasets. Lower SMAPE value indicates better model performance. For the synchronized federated frameworks, we plot results of *FedAvg* and *FedProx* at every 10 global iterations.

EXPERIMENT

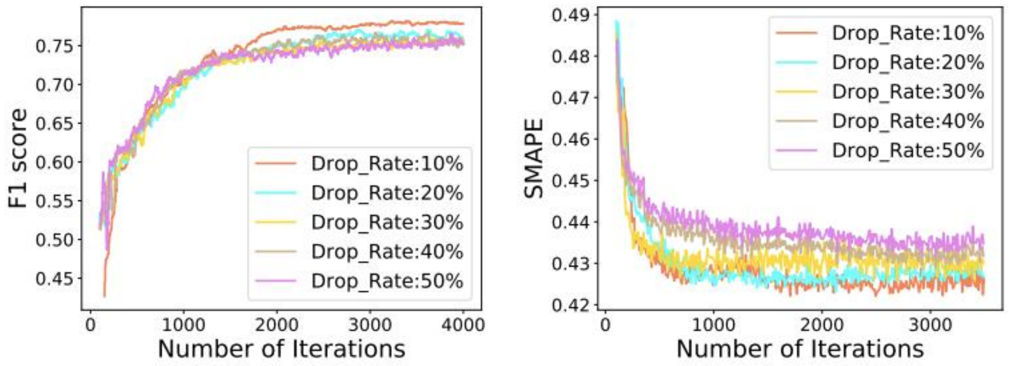
TABLE VI.1
COMPUTATION TIME (IN MINUTES) TO REACH TARGET TEST PERFORMANCE. THE NETWORK DELAY OF EACH CLIENT WAS SET TO BE A RANDOM VALUE BETWEEN $10 \sim 100$ SECONDS.

Method	FitRec	Air Quality	ExtraSensory	FMNIST
FedAvg	20.42	460.02	104.86	160.72
FedProx	19.26	439.95	99.45	160.36
FedAsync	15.41	326.45	87.97	151.72
ASO-Fed(-D)	16.31	332.74	95.77	158.83
ASO-Fed(-F)	**15.17**	320.92	**65.87**	150.54
ASO-Fed	15.43	**319.41**	87.40	**150.46**



(a) ExtraSensory (F1-socre ↑) (b) Air Quality (SMAPE ↓)

Fig. 4. Performance comparison of federated approaches as dropout rate of clients increases. ASO-Fed has better performance than the other federated frameworks.



(a) ExtraSensory (F1-score ↑) (b) Air Quality (SMAPE ↓)

Fig. 5. The performance of ASO-Fed with clients periodically dropping out.

EXPERIMENT

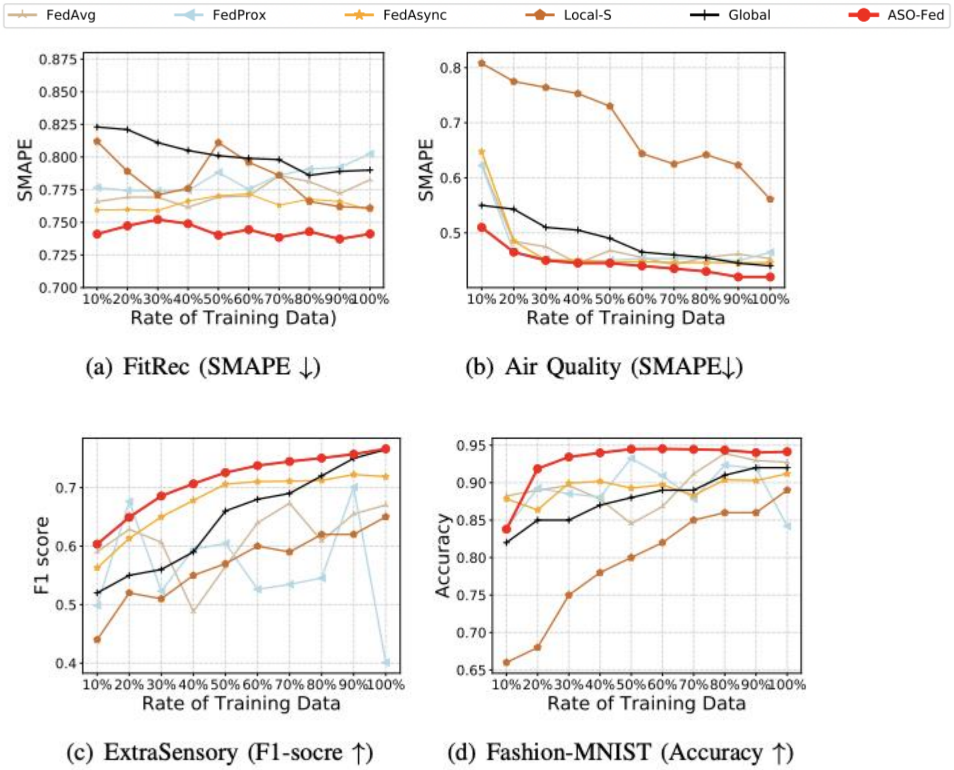


Fig. 6. Average performance comparison (SMAPE, F1, accuracy) on four datasets as training data increases.

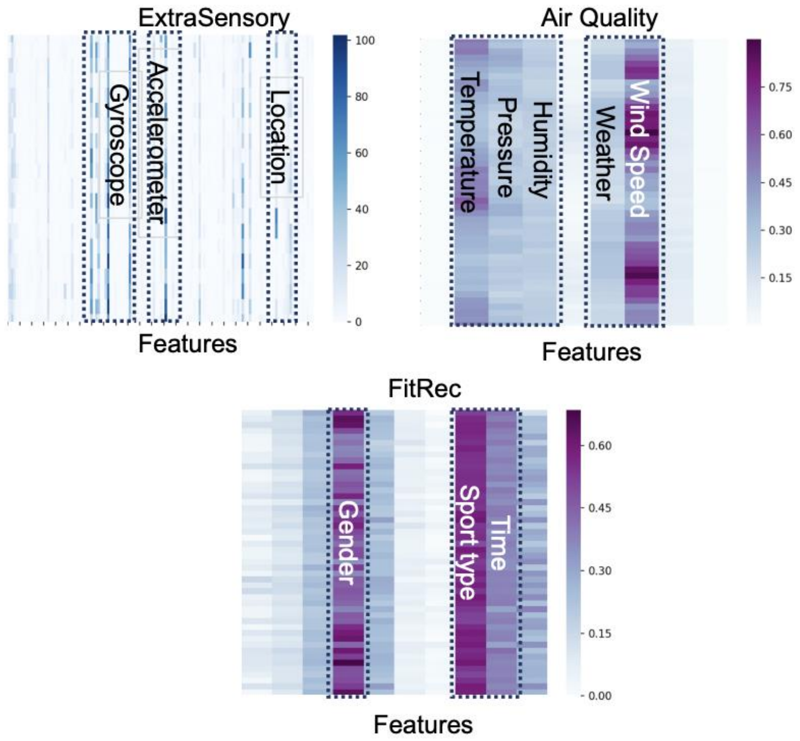
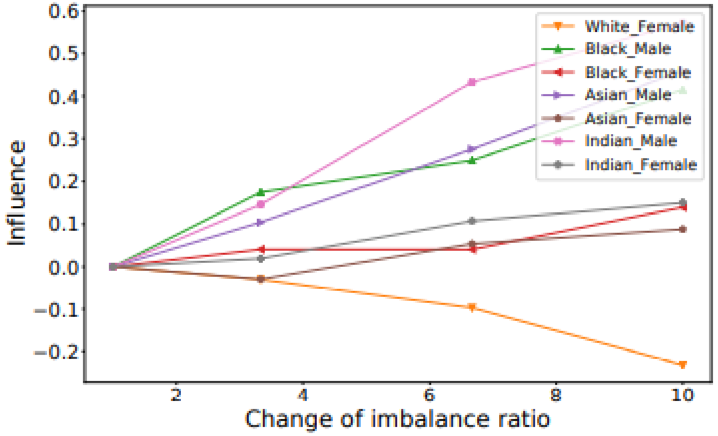


Fig. 7. Feature representation learned on the server of three real-world datasets. Each column is the weights vector within 48 time steps over the input series.

INTUITION

Selective Data Acquistion 알고리즘을 연합 학습에 적용함에 있어, Device간의 상이한 data acquistion cost를 고려하여 모델을 설계할 필요가 존재하며 그에 따른 threshold(learning rate 혹은 imbalance ratio의 limitation) 의 조정도 동적으로 일어나며 client별로 관리되어야할 필요성을 느꼈다.



ALGORITHM 1: Iterative algorithm for Slice Tuner

Input: The slices S, budget B, minimum slice size L, and data acquisition cost function C

```
sizes = SliceSizes(S);
T = 1;
if ∃i sizes[i] < L then
    /* Ensure minimum slice size L */
    num_examples = max(L × 1 − sizes, 0);
    sizes = sizes + num_examples;
    B = B - Σ_i(C(i) × num_examples[i]);
IR = GetImbalanceRatio(sizes);
while B > 0 do
    /* One-shot always uses the entire budget */
    num_examples = OneShot(sizes, B);
    After_IR = GetImbalanceRatio(sizes + num_examples);
    if |After_IR − IR| > T then
        /* Do not make imbalance ratio change exceed T */
        target_ratio = IR + T× Sign(After_IR − IR);
        change_ratio = GetChangeRatio(sizes, num_examples,
          target_ratio);
        num_examples = change_ratio × num_examples;
        After_IR = GetImbalanceRatio(sizes + num_examples);
    CollectData(num_examples);
    sizes = sizes + num_examples;
    B = B - Σ_i(C(i) × num_examples[i]);
    T = IncreaseLimit(T);
    IR = After_IR;
return;
Function GetImbalanceRatio(sizes):
    return max(sizes) / min(sizes);
```

THANK YOU