Unsupervised Federated Learning for Unbalanced Data

Mykola Servetnyk
Institute of Electronics
National Chiao Tung University
Hsinchu, Taiwan
rusly.eic04g@nctu.edu.tw

Carrson C. Fung
Institute of Electronics
National Chiao Tung University
Hsinchu, Taiwan
c.fung@ieee.org

Zhu Han
Dept. of Electrical and Computer Engineering
University of Houston
Houston, USA
zhan2@uh.edu

GLOBECOM 2020 - 2020 IEEE Global Communications Conference | 978-1-7281-8298-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/GLOBECOM42002.2020.9348203

Abstract—**This work considers unsupervised learning tasks being implemented within the federated learning framework to satisfy stringent requirements for low-latency and privacy of the emerging applications. The proposed algorithm is based on Dual Averaging (DA), where the gradients of each agent are aggregated at a central node. While having its advantages in terms of distributed computation, the accuracy of federated learning training reduces significantly when the data is nonuniformly distributed across devices. Therefore, this work proposes two weight computation algorithms, with one using a fixed size bin and the other with self-organizing maps (SOM) that solves the underlying dimensionality problem inherent in the first method. Simulation results are also provided to show that the proposed algorithms' performance is comparable to the scenario in which all data is uploaded and processed in the centralized cloud.**

Index Terms—**Federated learning, unsupervised learning, dual averaging algorithm, gradient weighting, distributed optimization, self-organizing maps.**

I. Introduction

Traditionally, machine learning, which consists of training and inference phases, is done at the centralized cloud computing, which can sustain more heavy computations than edge devices. However, nowadays imbuing edge devices with intelligence gives ability to train models locally, while providing privacy, security, regulatory and economic benefits. Recently proposed federated learning [1], [2] allows mobile edge devices, with limited computational resources, such as mobile phones and IoT devices, to learn a global model for prediction, while only perform training with local data. Given the fact that prediction is done at the edge, stringent requirements in low-latency applications can be met.

Clustering is one of the fundamental tasks in data analysis and signal processing, and plays an important role in various applications such as social network analysis, image and video processing and autonomous driving. These tasks are usually carried out at central servers and require the collected data to be transmitted from individual nodes/agents. While previous works on federated learning have only considered supervised learning with neural networks, the goal of this work is to propose a framework that supports unsupervised learning under the federated learning architecture.

Despite its advantages, there are few challenges that must be overcome before federated learning can be deployed in a large scale. The first challenge is unpredictable users, or processing agents, behaviors such as the unreliable device connectivity, interrupted execution, difference in convergence time for local training, which was considered in [3]–[6]. The second problem is the imbalance in collected data across the network, i.e., particular sensors may have limited observations of certain events, which imposes significant statistical challenges in the learning process. Several works have analyzed and attempted to resolve the problem. Reference [6] has proposed to learn separate models for each node through a multi-task learning framework. However, this approach creates additional communication bandwidth overhead, which the authors attempt to resolve. Reference [7] has shown that accuracy of federated learning reduces significantly, by up to 55%, for neural networks trained for highly skewed non-IID data. To resolve this issue, small subsets of data are globally shared among all edge devices. Experimental results have shown that sharing only 5% of local data can increase prediction accuracy

This work is partially supported by Ministry of Science and Technology Grant 108-2221-E-009-126, Ministry of Education project Trusted Intelligent Edge/Fog Computing Technology RSC 108B568 and Industry-Academia R&D Cooperation between Foxconn Inc. and National Chiao Tung University.

978-1-7281-8298-8/20/$31.00 ©2020 IEEE

by 30%. However, the approach is not applicable for unsupervised learning due to the absence of data labels.

In this work, the unsupervised learning scheme under the federated learning framework is presented, where processing agents in the network store non-identically distributed data sets. To the best of the authors' knowledge, this problem has not yet been explored in literature. The proposed algorithm relies on modifying the dual averaging (DA) algorithm [8], which is based on weighted gradient aggregation at the central node. However, proper weights need to be determined to reflect the non-IID nature of the observed data at the agents. Two methods are proposed to overcome this problem. The first scheme uses fixed size bins over the data to determine how much data is in a certain bin. However, the number of weights that needs to be computed grows exponentially with respect to the dimension of the data. The second method which exploits the mapping property of self-organizing maps (SOM) is proposed to tackle the curse of dimensionality problem.

Notations: Uppercase (lowercase) bold face letters indicate matrices (column vectors). Superscript ^{H} denotes Hermitian, ^{T} denotes transposition. $\mathbf{1}_M$ denotes an $M \times 1$ vector, containing 1 in all of its entries. $\langle \cdot, \cdot \rangle$ denotes the inner product operator. $\|\mathbf{a}\|$ denotes the ℓ_2 norm of $\mathbf{a}$.

II. System model & problem formulation

A. Federated learning framework

A typical federated learning procedure is shown in Fig. 1. At the initial step, a subset of agents is selected and a pretrained model is downloaded to the agents (top left). Next, the agents compute an updated model based on their local data (top right). After training, the model updates are sent from the selected agents back to the server (bottom right) and the server aggregates these models to construct an improved global model by simple averaging or other techniques (bottom left). This process continues until convergence or until the desired level of prediction accuracy is reached. Notice that in practical implementations of big models, whole model is not transmitted to the server but rather, only structural updates are sent.

B. System Model and Problem formulation

Assume the network consists of a set $\mathcal{J} = \{1, ..., J\}$ of data collecting and processing agents. The jth agent can send information about certain parameters to the centralized server. Parameter update is done at the server and is based on a convex combination of parameters that are received. To model this weighting process, let $\mathbf{w} \in \mathbb{R}^J$ be a vector containing nonnegative weights that is associated with how much emphasis should be placed on the estimated parameter for each agent, with $[\mathbf{w}]_j > 0$ denoting that a connection exists between the jth node and centralized server. It is assumed that $\mathbf{w}$ is a stationary stochastic vector, so that $\mathbf{w}^T \mathbf{1}_J = 1$. It is shown in Sec. IV that proper weighting is crucial to guarantee good performance and proper determination of it is described in the sequel.

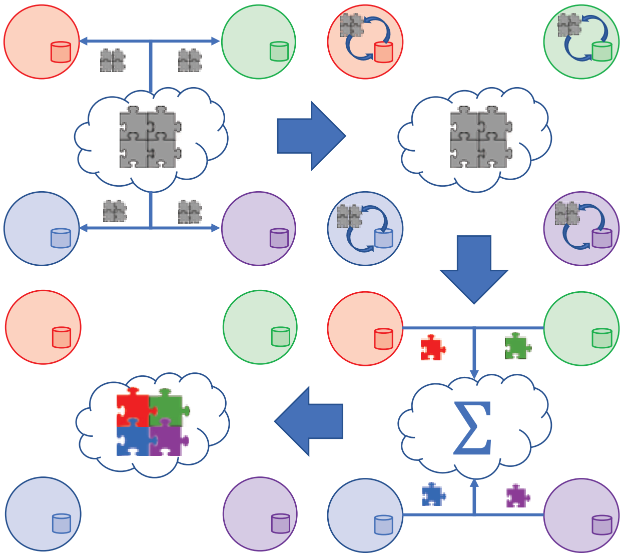


Fig. 1. Unsupervised federated learning procedure.

Assume that sensor node j observes a set of data $\mathcal{X}_j := \{\mathbf{x}_{jn}, n \in \mathcal{N}_j\}$ with $\mathcal{N}_j = \{1, ..., N_j\}$ being a set of neighboring nodes, where $\mathbf{x}_{jn} \in \mathbb{R}^q$ denotes the nth observation at the jth node and q denotes the data dimension. Each observation is assumed to be drawn from one class $\mathcal{C}_k$ with $k \in \{1, ..., K\} \triangleq \mathcal{K}$, where K denotes total number of classes and assumed to be known *a priori*, or can be estimated by various algorithms [9], e.g. the elbow method, average silhouette method or gap statistic method.

The goal is to assign each observation point to a particular cluster and estimate the cluster centroid. Presented below considers hard assignment scheme, i.e., each point only belongs to one cluster; however, it can easily be cast into a soft clustering scheme where each point is assigned to a cluster with a certain probability. Denote the centroid of cluster $\mathcal{C}_k$ as $\mathbf{m}_k \in \mathbb{R}^q$, and the membership label as $\mu_{jnk} \in \{0, 1\}$, so $\mu_{jnk} = 1$ if $\mathbf{x}_{jn}$ is assigned to $\mathcal{C}_k$ and $\mu_{jnk} = 0$, otherwise. The clustering problem can be formulated as

$$
\begin{aligned}
&\min_{\mu_{jnk},\mathbf{m}_k} \sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}}\sum_{n\in\mathcal{N}_j} \frac{1}{2}\mu_{jnk}\|\mathbf{m}_k-\mathbf{x}_{jn}\|^2\\
&\quad\text{s.t.} \sum\nolimits_{j\in\mathcal{J}}\mu_{jnk}=1,\ k\in\mathcal{K}, n\in\mathcal{N}_j,\\
&\qquad\mu_{jnk}\in\{0,1\}, j\in\mathcal{J}, k\in\mathcal{K},\ n\in\mathcal{N}_j,
\end{aligned}
\tag{1}
$$

where the Euclidean distance between cluster centroids and corresponding assigned data points is minimized and the constraints describe hard assignment.

III. Proposed algorithm

A. Algorithm description

Prior to clustering, all agents conduct simple data analysis and send statistics to server, that decides which weight each agent will have in a certain region of the data space. Next, network runs an iterative algorithm similar to the k-means algorithm, which consists of the following steps. First, cluster initialization using k-means++ [10] is done. This is followed by an iterative algorithm consisting of two steps. The first step is data labeling, where data points are assigned to specific cluster at each node. Inspired by simulated annealing, the proposed algorithm uses distance perturbation, which may avoid local minima and results in better clustering performance. The second step computes data centroids via the proposed DA based method, where weighted gradients are combined at the central node. Each step is described in the sequel.

B. Data labeling step

A simple way to solve (1) for a suboptimal μ_{jnk} is to employ the following procedure. After all agents have updated their respective centroids, each data point from the locally observed dataset at each agent is assigned to a specific cluster based on the distance from that agent's centroid, i.e. $\mu_{jnk}=1$ if $k=\arg\min\ \|\mathbf{x}_{jn}-\mathbf{m}_{jk}^{(t_1)}\|$ and $\mu_{jnk}=0$ otherwise. However, inspired by [11], it is modified as

$$
\mu_{jnk}=\begin{cases}1, \text{if } k=\arg\min\,(1+\frac{\xi}{t_1})\|\mathbf{x}_{jn}-\mathbf{m}_k^{(t_1)}\|,\\ 0, \text{otherwise},\end{cases}
\tag{2}
$$

where ξ is random variable drawn from uniform distribution $\xi\sim\mathcal{U}(0;\xi_{\max})$ between 0 and $\xi_{\max}$. $\frac{\xi}{t_1}\|\mathbf{x}_{jn}-\mathbf{m}_{jk}^{(t_1)}\|$ in (2) can be interpreted as a perturbation term on the distance between centroid $\mathbf{m}_{jk}$ and point $\mathbf{x}_{jn}$. Setting $\xi_{\max}$ to a large value leads to stronger perturbation and as a result, there is a higher chance to avoid local minima in (1), but at the cost of requiring more iterations for the algorithm to converge.

C. DA-based centroid computation step

A modified DA subgradient based method is proposed herein. Given μ_{jnk} from step 2, at (inner iteration) time step t_2 of the algorithm, each node calculates a gradient $\mathbf{g}_{jk}^{(t_2)}$ of local objective function $\sum_{n\in\mathcal{N}_j}\frac{1}{2}\|\mathbf{m}_k-\mathbf{x}_{jn}\|^2$ as

$$
\mathbf{g}_{jk}^{(t_2)}=\sum_{n\in\mathcal{N}_j}\mu_{jnk}(\mathbf{m}_k^{(t_2)}-\mathbf{x}_{jn}).
\tag{3}
$$

Next, the DA generates a sequence of iterates $\{\mathbf{m}_k^{(t_2)},\mathbf{z}_{jk}^{(t_2)}\}_{t_2=1}^{\infty}$ at the central node using the update equations

$$
\mathbf{z}_k^{(t_2+1)}=\mathbf{z}_k^{(t_2)}+\sum_{j\in\mathcal{J}}[\mathbf{w}]_j\mathbf{g}_{jk}^{(t_2)},
\tag{4a}
$$

$$
\mathbf{m}_k^{(t_2+1)}=\arg\min\,\langle\mathbf{z}_k^{(t_2+1)},\mathbf{m}_k\rangle+\alpha^{(t_2)}\|\mathbf{m}_k\|^2,
\tag{4b}
$$

where $\mathbf{z}_k^{(t_2)}$ can be seen as accumulated gradient for cluster k at iteration t_2. First term on the right-hand side of (4b) is the first-order approximation of the objective and the second is a regularization term to turn the problem into strictly convex form, which prevents the solution from being unbounded. $\alpha^{(t_2)}$ is nonincreasing positive sequence and acts as regularization parameter and has to be carefully selected. Notice that (4a) is different from that of the conventional DA algorithm in which weighting of $\mathbf{g}_{jk}^{(t_2)}$ is now performed. The proposed method can also be viewed as a combination of stochastic gradient descent (SGD) with minibatch gradient aggregation [12]. Similar to the latter, the data at each agent can be viewed as randomly selected data subset and the gradient is computed based on this subset. Hence the data at the agent is similar to a minibatch. In addition, the gradients from different agents are convexly combined at the central node during the centroid update step, which is the main difference between the proposed method and existing techniques. A data driven approach is used to compute the weights used in the above convex combination and it is described in the following section.

D. Weight computation via bin method

Consider the scenario shown in Fig. 2, where 3 sensor nodes observe data points from 16 clusters indicated by different colors. It is clear that each agent in the figure only has partial observations of the entire dataset. It is intuitive to weigh the gradients $\mathbf{g}_{jk}^{(t_2)},\forall j$ that correspond to each cluster by the number of points that are observed at agent j. In particular, it can be seen that agent 1 will have the highest weight for the cluster at the bottom left of the grid in Fig. 2, while agents 2 and 3 will

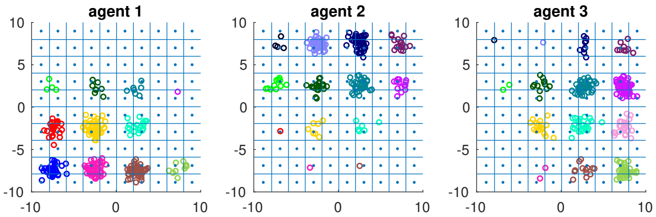


Fig. 2. Unbalanced data sets observed at different agents.

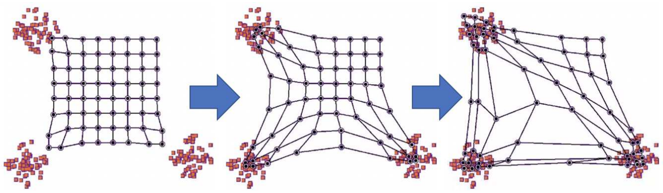


Fig. 3. Self-organizing maps training.

have zero weight as this cluster is not observed by those agents. Unfortunately, weights cannot be computed in this fashion because the data labels remain unknown before clustering is done. Therefore, it is proposed to assign the weights by dividing the data space into a grid with uniform-sized bins as shown in Fig. 2 and calculate the number of points falling into a particular *bin* (a region of the grid) at each node. The number of points for each bin is sent to the central server from each agent so that the weight for each particular bin at each agent can be computed as the number of points falling in that bin at that agent divided by the total number of points in that bin from all agents. This guarantees that $\mathbf{1}_J^T\mathbf{w} = 1$ so that the algorithm will converge, which will be shown in later publication. The above approach, however, suffers from the curse of dimensionality. For instance, in Fig. 2, the data dimension $q = 2$ and the number of bins required equals 10^q.

Algorithm 1: DA-based unsupervised federated learning.

```
Result: $\mu_{jnk}$, $\mathbf{m}_k$, $k \in \mathcal{K}$, $j \in \mathcal{J}$, $n = 1...N_j$
1. Each node $j$ : Train SOM $\mathbf{P}_j$ and send it to server;
2. Server combines received models and send it to nodes;
3. Each node calculates number of points falling in the bin and sent it to server to determine gradient weight $\mathbf{w}$
4. Initialize $\mathbf{m}_k^{(0)}$ via $k$-means++, send it to nodes;
$t_1 = 0$;
while $\forall j \in \mathcal{J} : \sum_k \sum_n \mu_{jnk}^{(t_1)} - \mu_{jnk}^{(t_1-1)} \neq 0$ do
    5. Each node $j$: assign labels $\mu_{jnk}^{(t_1)}$ based on (2).
    $\mathbf{z}_k^{(0)} = 0$, select $\alpha^{(t_2)}$; $t_2 = 0$;
    while $\|\mathbf{m}_k^{(t_2)} - \mathbf{m}_k^{(t_2-1)}\| \leq \epsilon_{glo}$ do
        6. Each node $j$: send $\mathbf{g}_{jk}^{(t_2)}$ to server; $\mathbf{z}_k^{(t_2+1)}$ by (4a);
        7. Server computes $\mathbf{z}_k^{(t_2+1)}$ by (4a), $\mathbf{m}_k^{(t_2+1)}$ by (4b) and send it back to nodes;
    $t_1 = t_1 + 1$;
```

E. Weight computation via self-organizing maps

Self-organizing maps (SOM) [13], which is often used to produce a low-dimensional representation of the input space of training samples, can be used to resolve the curse of dimensionality problem. SOM consists of set of neurons $\mathcal{M}$ with its coordinate vector $\mathbf{p}_m \in \mathbb{R}^q, m \in \mathcal{M} = \{1, \ldots, M\}$. Hence, the entire SOM is parameterized by $\mathbf{P}_j \triangleq [\mathbf{p}_1, \cdots, \mathbf{p}_M]$. Typically, training process consists of the following steps. 0) All neurons are initialized with small values of their weights. 1) For each data point, the neurons compute the distance to the data point and the closest neuron is declared as the winner. 2) The winning neuron determines the neighborhood of excited neurons and these neurons adjust their individual weights towards the data point. 3) Neurons decrease neighborhood radius and learning rate. These steps are repeated at each sensor until convergence and data map is obtained. Example of evolution of SOM training is shown on Fig. 3. After $\mathbf{P}_j$ has been constructed from the local data observed at the jth agent, the proposed algorithm will send this back to the central server to form the aggreated SOM $\mathbf{P} \triangleq [\mathbf{P}_1, \cdots, \mathbf{P}_j]$. $\mathbf{P}$ will then be sent back to each node and each vector inside $\mathbf{P}$ will be used as a centroid for the bin similar to the one discussed in Sec. III-D. The number of points falling into each bin is then calculated and the corresponding weights for the gradients in (4a) can be found. The overall scheme is summarized in Algorithm 1. Due to space limitation, only a brief discussion concerning the convergence of the proposed algorithm is given herein. The convergence rate of the centroid computation step (inner loop) is $O(\frac{\gamma}{T})$, where γ is a parameter that depends on the regularization parameter $\alpha^{(t_2)}$ and Euclidean distance between the initialized and optimal variable. T denotes the total number of iterations in the inner loop. Global convergence of the algorithm (outer loop) is similar to those of standard k-means, and probabilistic convergence upper bound can be computed based on [14], [15].

IV. Numerical results

A. Simulation Setup

In the following figures, the proposed algorithms a labeled as *UFLBin* and *UFLSOM*. The centralized means algorithm is used as the performance benchmar with all using the initialization and perturbation tec niques described in Sec. II-B. In addition, uniform grad ent weighting, labeled as *UFLUni* is used for comparis to show the benefit of proper weighting.

Data are generated at random from $K = 16$ classe with vectors from each class generated from a symmetr Gaussian distributions with means $\mathbf{m}_1 = [-7.5, -7.5$ $\mathbf{m}_2 = [-7.5, -2.5]^T$, $\mathbf{m}_3 = [-7.5, 2.5]^T$, $\mathbf{m}_4$ $[-7.5, 7.5]^T$, $\mathbf{m}_5 = [-2.5, -7.5]^T$, $\mathbf{m}_6 = [-2.5, -2.5]$, $\mathbf{m}_7 = [-2.5, 2.5]^T$, $\mathbf{m}_8 = [-2.5, 7.5]^T$, $\mathbf{m}_9 = [2.5, -7.5]^T$, $\mathbf{m}_{10} = [2.5, -2.5]^T$, $\mathbf{m}_{11} = [2.5, 2.5]^T$, $\mathbf{m}_{12} = [2.5, 7.5]^T$, $\mathbf{m}_{13} = [7.5, -7.5]^T$, $\mathbf{m}_{14} = [7.5, -2.5]^T$, $\mathbf{m}_{15} = [7.5, 2.5]^T$, $\mathbf{m}_{16} = [7.5, 7.5]^T$ similar to Fig. 2. Each cluster contains 50 points. The network consists of $J = 5$ sensing agents that are placed randomly and observe data according to energy detector for Rayleigh fading with a probability $P(d_j, \text{SNR}_j) = 1 - \exp(-\frac{\alpha \text{SNR}_j}{d_j})$. d_j is the distance between the jth agent and the data points, and SNR_j is the signal-to-noise ratio in linear scale at the jth agent. α is an observation parameter and is set to 0.025 in the simulations.

B. Simulation Results

Convergence results are shown in Fig. 4. It is clear that the proposed algorithms take relatively the same number of global iterations (indexed by t_1 in Algorithm 1) to converge. It can be observed that the convergence rate is similar to that of a centralized k-means algorithm. Clustering performance vs. SNR is shown in Fig. 5, where four metrics are used, namely, objective value, variation of information (VI), Rand Index (RI) and Adjusted Rand Index (ARI). First metric reflects accuracy of centroid estimation, while the other three describes correctness of data labels. Both UFLBin and UFLSOM perform better than UFLUni and close to k-means in terms of objective. Hence, the proposed algorithms are able to achieve superior performance, while the SOM based method requires less signaling than the bin method.

V. Conclusion

Two DA based unsupervised federated learning schemes are proposed to tackle the problem of non-IID data. The first one uses a bin method that suffers from the curse of dimensionality problem, while the second can overcome this problem, albeit incurring slightly more computations. The proposed methods have been shown to achieve good results compared to uniform gradient weighting and the centralized k-means.

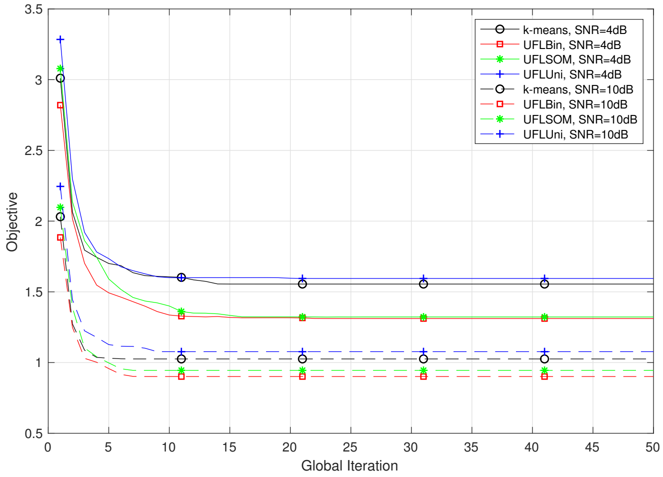


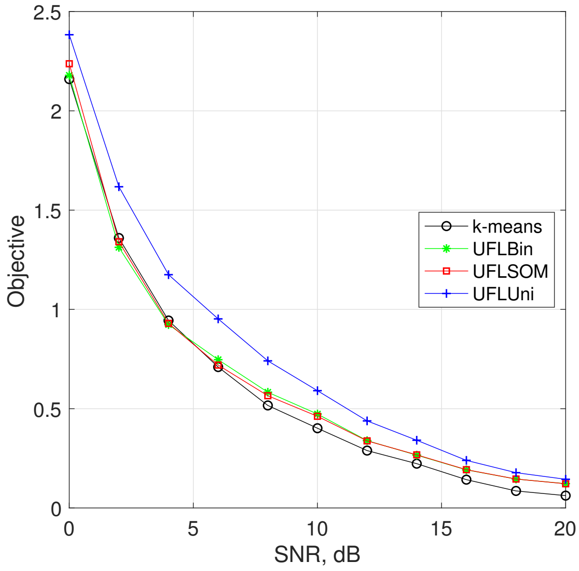
Fig. 4. Convergence of the proposed algorithms.

References

[1] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, Oct. 2016.

[2] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, Oct. 2016.

[3] H. Zhu and Y. Jin, "Multi-objective evolutionary federated learning," *IEEE transactions on neural networks and learning systems*, pp. 1–13, Jun. 2019.



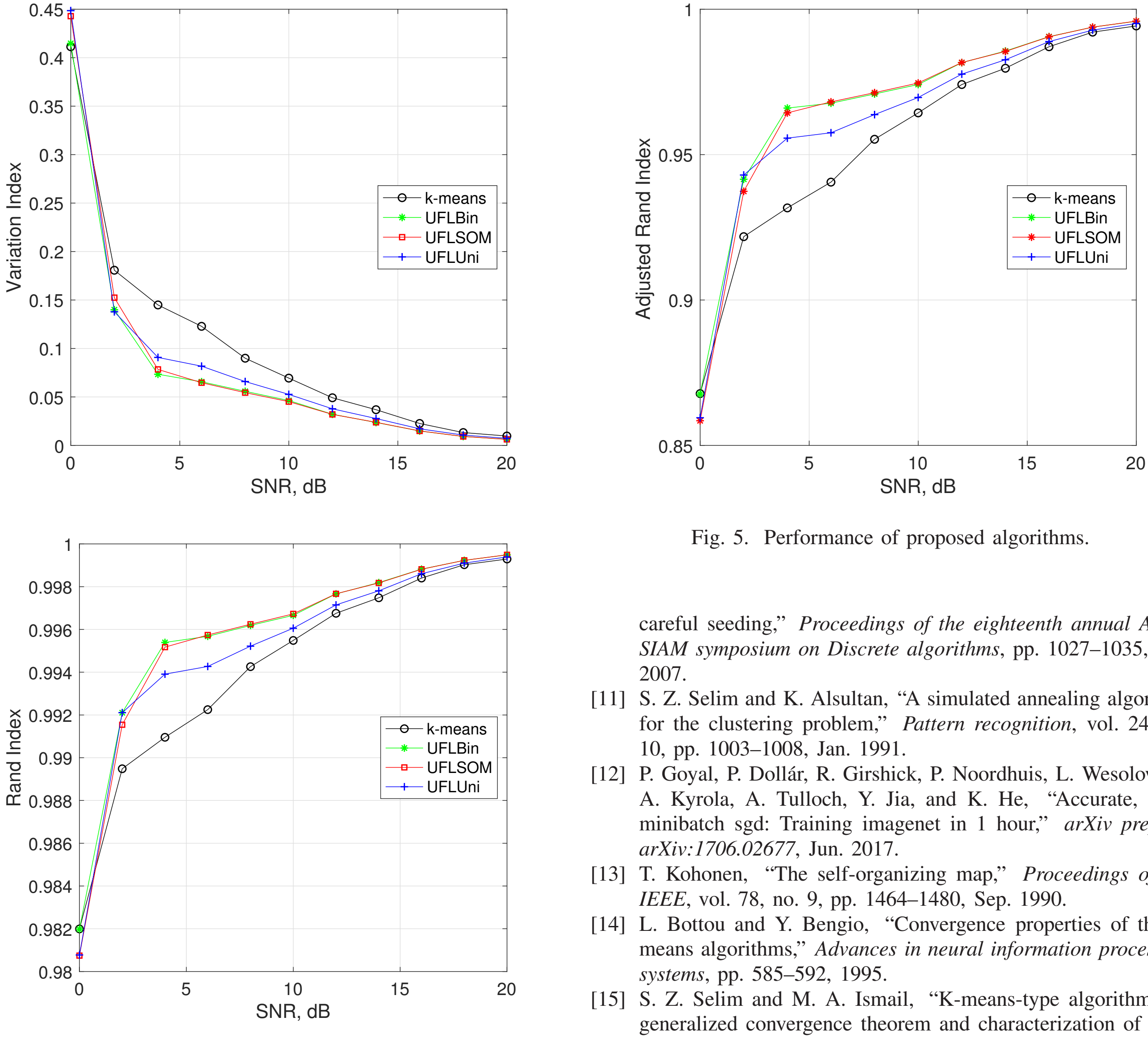


Fig. 5. Performance of proposed algorithms.

[4] P. Blanchard, R. Guerraoui, J. Stainer, et al., "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in Neural Information Processing Systems*, pp. 119–129, Dec. 2017.

[5] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 44(1–25), Dec. 2017.

[6] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Advances in Neural Information Processing Systems*, pp. 4424–4434, Dec. 2017.

[7] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, Jun. 2018.

[8] A. Agarwal, M. J. Wainwright, and J. C. Duchi, "Distributed dual averaging in networks," *Advances in Neural Information Processing Systems*, pp. 550–558, Dec. 2010.

[9] A. Kassambara, *Practical guide to cluster analysis in R: Unsupervised machine learning*, vol. 1, STHDA, Aug. 2017.

[10] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035, Jan. 2007.

[11] S. Z. Selim and K. Alsultan, "A simulated annealing algorithm for the clustering problem," *Pattern recognition*, vol. 24, no. 10, pp. 1003–1008, Jan. 1991.

[12] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, Jun. 2017.

[13] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, Sep. 1990.

[14] L. Bottou and Y. Bengio, "Convergence properties of the k-means algorithms," *Advances in neural information processing systems*, pp. 585–592, 1995.

[15] S. Z. Selim and M. A. Ismail, "K-means-type algorithms: A generalized convergence theorem and characterization of local optimality," *IEEE Transactions on pattern analysis and machine intelligence*, , no. 1, pp. 81–87, Jan 1984.