Semi-Asynchronous Energy-Efficient Federated Prototype Learning for Client-Edge-Cloud Architectures

***Abstract*—**

***Index Terms*—Federated Prototype Learning, Hierarchical Architecture, Heterogeneous Models, Asynchronous Communication, Energy Efficiency**

I. Introduction

THIS

II. Related Work

III. Motivation

IV. Method

A. Proposed Algorithm New Version

TABLE I
SYMBOL TABLE

Symbol	Description
J	Number of classes
T	Global communication rounds
E	Edge communication rounds
K	Local train epochs
L	Number of edge servers
B	The buffer of the cloud server with a static length
N^l	The number of clients on the l-th each edge server
$\mathcal{N}_j^l$	The number of instances belonging to class j on edge l that have already participated in computation
$\mathcal{N}_j^{l,\text{prev}}$	Previous number of instances belonging to class j on edge that have already participated in computation l
$\mathcal{N}_j^{l,i}$	The number of instances belonging to class j on client i on edge l
S^l	Set of clients participating in training in the l-th edge server
$\bar{C}_j$	Aggregated prototype of class j in the cloud edge server
C_j^l	Aggregated prototype of class j from the l-th edge server
$C_j^{l,\text{prev}}$	Previous version of aggregated prototype of class j from the l-th edge server in the cloud server
$c_{i,j}^l$	Aggregated prototype of class j from the client i in the l-th edge server
$c_{i,j}^{l,\text{prev}}$	Previous version of the aggregated prototype of class j from client i in the l-th edge server in the edge server
$D_{i,j}^l$	A subset of the local dataset D_i^l of the i-th client in the l-th edge server, containing training instances of class j.
$\mathcal{X}_i^l$	The feature and label set of the i-th client in the l-th edge server, containing all features of D_i.
G^l	The global classifier in the l-th edge server.

The following algorithm demonstrates how to calculate the factorial of a number.

Algorithm 1 Hierarchical Federated Prototype Learning -Part 1

procedure CLOUD SERVER EXECUTES
  Initialize weights for clients with heterogeneous models.
  All edge servers execute in parallel.
  for $t = 1, \dots, T$ **do**
    Clear the buffer B
    while B is not full **do** ▷ Async process
      Receive a triple $(C^l, \mathcal{N}^l)$ from one edge server.
      Populate B with the received triple.
    end while
    $\bar{C}, G \leftarrow \text{CloudUpdate}(B)$
    Send $\bar{C}, G$ to edge servers participating in the current global aggregation.
    These edge servers re-execute.
  end for
end procedure
procedure EDGE SERVER EXECUTES
  Receive $\bar{C}, G$ from the cloud server
  Choose a set of clients S^l to train in parallel.
  for $e = 1, \dots, E$ **do** ▷ E now is static 1
    Send $\bar{C}, G$ to client $i \in S^l$
    for each client i in parallel **do**
      $c_i^l \leftarrow \text{ClientUpdate}(i, \bar{C}, G)$
    end for ▷ Wait for all clients
    $(C^l, \mathcal{N}^l) \leftarrow \text{EdgeAggregate}(\{c_i^l\}_{i \in S^l})$
    $\bar{C} \leftarrow \text{EdgeUpdate}(\bar{C}, C^l)$ ▷ not used now
  end for
  Send a triple $(C^l, \mathcal{N}^l)$ to the cloud server
end procedure

Algorithm 2 Hierarchical Federated Prototype Learning -Part 2

procedure CLOUDUPDATE(B)
  for $j = 1, \ldots, J$ **do**
    $\hat{C}_j \leftarrow \sum_{l=1}^{L} \mathcal{N}_j^{l,\text{prev}} \cdot \bar{C}_j$ ▷ Extend the aggregated prototypes $\bar{C}$
    for $(C^l, \mathcal{N}^l) \in B$ **do**
      $\hat{C}_j \leftarrow \hat{C}_j + \mathcal{N}_j^l \cdot C_j^l$
      if $C_j^{l,\text{prev}}$ is not empty **then**
        $\hat{C}_j \leftarrow \hat{C}_j - \mathcal{N}_j^{l,\text{prev}} \cdot C_j^{l,\text{prev}}$
      end if
      $\mathcal{N}_j^{l,\text{prev}} \leftarrow \mathcal{N}_j^l$
    end for
    $\bar{C}_j \leftarrow \frac{\hat{C}_j}{\sum_{l=1}^{L} \mathcal{N}_j^{l,\text{prev}}}$
  end for
  $C^{l,\text{prev}} \leftarrow C^l$ for $l \in B$
  for $j = 1, \ldots, J$ **do**
    $\mu_j \leftarrow \bar{C}_j$
    $\Sigma_j = \frac{1}{\mathcal{N}_j - 1} \sum_{l=1}^{L} \mathcal{N}_j^{l,\text{prev}} (C^{l,\text{prev}} - \mu)(C^{l,\text{prev}} - \mu)^\top$
  end for
  Generate samples $X \sim \mathcal{N}(\mu, \Sigma)$ using the multivariate normal distribution
  Use the generated samples X to train the global classifier G
  return $\bar{C}, G$
end procedure
procedure EDGEAGGREGATE($l, \{c_i^l\}_{i \in S^l}$)
  for $j = 1, \ldots, J$ **do**
    $\hat{C}_j^l \leftarrow \mathcal{N}_j^l \cdot C_j^l$
    for each c_i^l **do**
      $\hat{C}_j^l \leftarrow \hat{C}_j^l + \mathcal{N}_j^{l,i} \cdot c_{i,j}^l$
      if $c_{i,j}^{l,\text{prev}}$ is not empty **then**
        $\hat{C}_j^l \leftarrow \hat{C}_j^l - \mathcal{N}_j^{l,i} \cdot c_{i,j}^{l,\text{prev}}$
      else
        $\mathcal{N}_j^l \leftarrow \mathcal{N}_j^l + \mathcal{N}_j^{l,i}$
      end if
    end for
    $C_j^l \leftarrow \frac{\hat{C}_j^l}{\mathcal{N}_j^l}$
  end for
  $c_i^{l,\text{prev}} \leftarrow c_i^l$ for $i \in S^l$
  return $(C^l, \mathcal{N}^l)$
end procedure
procedure CLIENTUPDATE($i, \bar{C}, G$)
  Receive $\bar{C}, G$ from the edge server
  for $k = 1, \ldots, K$ **do**
    DVFS to be implemented...
    for batch $(x, y) \in D_i$ **do**
      Compute client prototypes by Eq.?.
      Compute loss by Eq.? using client prototypes and the global classifier G.
      Update client model according to the loss.
    end for
  end for
  return c_i^l
end procedure

V. EXPERIMENTS

VI. CONCLUSION