Algorithms

1850245174

December 2024

1 Introduction

This paper introduces various algorithms used in our research.

2 Asynchronous Baseline Algorithm

Below is a table of symbols used in the algorithms:

Symbol	**Description**
J	Number of classes
T	Global rounds
E	Edge rounds
K	Local epochs
L	Number of edge servers
B	The buffer of the cloud server with a static length
N^l	Number of clients in the l-th each edge server
$D_{i,j}$	A subset of the local dataset D_i of the i-th client, containing training instances of class j.
$\mathcal{N}_j^l$	Number of clients in edge l containing class j that have participated in aggregation
$\mathcal{N}_j^{l,\mathrm{old}}$	Last number of clients in edge l containing class j that have participated in aggregation
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
$C_j^{l,\mathrm{old}}$	Last version of aggregated prototype of class j from the l-th edge server stored in the cloud server
$c_{i,j}^l$	Aggregated prototype of class j from the client i in the l-th edge server
$c_{i,j}^{l,\mathrm{old}}$	Last version of the aggregated prototype of class j from client i in the l-th edge server stored in the edge server

Table 1: Symbol Table

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 global prototype set $\bar{C}$ for all classes and weights for clients with heterogeneous models
  All edge servers choose a set of clients S^l and execute in parallel
  for $t = 1, \ldots, T$ **do**
    Clear the buffer B
    while The buffer is not full **do**
      Receive a pair $(C^l, \mathcal{N}^l)$ from one edge server
      Fill the buffer B with C^l
    end while
    $\bar{C} \leftarrow \text{CloudUpdate}(B)$
    Send $\bar{C}$ to edge servers participating in the current global aggregation
    These edge servers rechoose S^l and re-execute
  end for
end procedure
procedure EDGE SERVER EXECUTES
  Receive $\bar{C}$ from the cloud server
  for $e = 1, \ldots, E$ **do** ▷ Edge rounds, E is static 1
    Send $\bar{C}$ to client $i \in S^l$
    for each client i in parallel **do**
      $c_i^l \leftarrow \text{ClientUpdate}(i, \bar{C}_i)$
    end for
    $C^l \leftarrow \text{EdgeAggregate}(\{c_i^l\}_{i \in S^l})$
    $\bar{C} \leftarrow \text{EdgeUpdate}(\bar{C}, C^l)$ ▷ Must know client distribution of all edges, not implement now
  end for
  Send a pair $(C^l, \mathcal{N}^l)$ to the cloud server
end procedure

Algorithm 2 Hierarchical Federated Prototype Learning -Part 2

procedure CLOUDUPDATE(buffer)
  for $j = 1, \ldots, J$ **do**
    $\hat{C}_j \leftarrow \sum_{l=1}^{L} \mathcal{N}_j^{l,\text{old}} \cdot \bar{C}_j$ ▷ Extend the aggregated prototypes $\bar{C}$
    for $(C^l, \mathcal{N}^l) \in$ buffer **do**
      $\hat{C}_j \leftarrow \hat{C}_j + \mathcal{N}_j^l \cdot C_j^l$
      if $C_j^{l,\text{old}}$ is not empty **then**
        $\hat{C}_j \leftarrow \hat{C}_j - \mathcal{N}_j^{l,\text{old}} \cdot C_j^{l,\text{old}}$
      end if
      $\mathcal{N}_j^{l,\text{old}} \leftarrow \mathcal{N}_j^l$
    end for
    $\bar{C}_j \leftarrow \frac{\hat{C}_j}{\sum_{l=1}^{L} \mathcal{N}_j^{l,\text{old}}}$
  end for
  $C^{l,\text{old}} \leftarrow C^l$ for $l \in B$
  return $\bar{C}$
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**
      $\hat{C}_j^l \leftarrow \hat{C}_j^l + c_{i,j}^l$
      if $c_{i,j}^{l,\text{old}}$ is not empty **then**
        $\hat{C}_j^l \leftarrow \hat{C}_j^l - c_{i,j}^{l,\text{old}}$
      else
        $\mathcal{N}_j^l \leftarrow \mathcal{N}_j^l + 1$
      end if
    end for
    $C_j^l \leftarrow \frac{\hat{C}_j^l}{N_j^l}$
  end for
  $c_i^{l,\text{old}} \leftarrow c_i^l$ for $i \in S^l$
  return C^l
end procedure
procedure CLIENTUPDATE($i, \bar{C}_i$)
  for $k = 1, \ldots, K$ **do**
    for batch $(x, y) \in D_i$ **do**
      Compute client prototypes by Eq.?.
      Compute loss by Eq.? using client prototypes.
      Update client model according to the loss.
    end for
  end for
  return c_i^l
end procedure

3 Proposed Algorithm