Adaptive Federated Learning Takeaways

Efficient Resource Utilization (Pages 4-5, 19-20):

- The paper discusses how to make efficient use of resources such as time, energy, and communication bandwidth during the federated learning process. This is crucial for social media platforms where computational resources are distributed across numerous devices with varying capabilities.
- **Page 4-5:** The section on problem formulation addresses the need to optimize resource usage to minimize the loss function while respecting resource constraints.
- **Page 19-20:** Tables III, IV, and V provide details on resource consumption parameters, which can be useful for planning and simulating resource usage in your project.

Handling Non-IID Data (Pages 6-7):

- Federated learning often deals with non-IID (independent and identically distributed) data, which is common in social media data due to diverse user behaviors and content types.
- **Page 6-7:** The paper's algorithm is designed to handle such non-IID data by averaging local updates and performing global aggregations, ensuring that the model can generalize well despite the heterogeneous data.

Scalability (Pages 5, 19-20):

- Scalability is addressed by optimizing the frequency of global aggregations and local updates, which helps in managing large-scale distributed systems like social media platforms.
- **Page 5:** The discussion on the optimization problem involving τ (local update steps between global aggregations) and T (total iterations) is crucial for scaling the federated learning process efficiently.

"We note that instead of transmitting the
entire model parameter vector in every global aggregation
step, one can also transmit compressed or quantized model
parameters to further save the communication bandwidth,
where the compression or quantization can be performed using
techniques described in [19], [20], "

Algorithm 2: Procedure at the aggregator

Input: Resource budget R, control parameter φ, search range parameter γ, maximum τ value $\tau_{\max}$

Output: $\mathbf{w}^{\mathrm{f}}$

Initialize $\tau^* \leftarrow 1$, $t \leftarrow 0$, $s \leftarrow 0$; //s is a resource counter

Initialize $\mathbf{w}(0)$ as a constant or a random vector;

Initialize $\mathbf{w}^{\mathrm{f}} \leftarrow \mathbf{w}(0)$;

repeat

  Send $\mathbf{w}(t)$ and τ^* to all edge nodes, also send STOP if it is set;

  $t_0 \leftarrow t$; //Save iteration index of last transmission of $\mathbf{w}(t)$

  $t \leftarrow t + \tau^*$; //Next global aggregation is after τ iterations

  Receive $\mathbf{w}_i(t)$, $\hat{c}_i$ from each node i;

  Compute $\mathbf{w}(t)$ according to (5);

  if $t_0 > 0$ **then**

    Receive $\hat{\rho}_i$, $\hat{\beta}_i$, $F_i(\mathbf{w}(t_0))$, $\nabla F_i(\mathbf{w}(t_0))$ from each node i;

    Compute $F(\mathbf{w}(t_0))$ according to (2)

    if $F(\mathbf{w}(t_0)) < F(\mathbf{w}^{\mathrm{f}})$ **then**

      $\mathbf{w}^{\mathrm{f}} \leftarrow \mathbf{w}(t_0)$;

    if *STOP flag is set* **then**

      break; //Break out of the loop here if STOP is set

    Estimate $\hat{\rho} \leftarrow \frac{\sum_{i=1}^{N} D_i \hat{\rho}_i}{D}$;

    Estimate $\hat{\beta} \leftarrow \frac{\sum_{i=1}^{N} D_i \hat{\beta}_i}{D}$;

    Compute $\nabla F(\mathbf{w}(t_0)) \leftarrow \frac{\sum_{i=1}^{N} D_i \nabla F_i(\mathbf{w}(t_0))}{D}$, estimate $\hat{\delta}_i \leftarrow \|\nabla F_i(\mathbf{w}(t_0)) - \nabla F(\mathbf{w}(t_0))\|$ for each i, from which we estimate $\hat{\delta} \leftarrow \frac{\sum_{i=1}^{N} D_i \hat{\delta}_i}{D}$;

    Compute new value of τ^* according to (19) via linear search on integer values of τ within $[1, \tau_{\mathrm{m}}]$, where we set $\tau_{\mathrm{m}} \leftarrow \min\{\gamma\tau^*; \tau_{\max}\}$;

  for $m = 1, 2, ..., M$ **do**

    Estimate resource consumptions $\hat{c}_m$, $\hat{b}_m$, using $\hat{c}_{m,i}$ received from all nodes i and local measurements at the aggregator;

    $s_m \leftarrow s_m + \hat{c}_m \tau + \hat{b}_m$;

  if *$\exists m$ such that $s_m + \hat{c}_m(\tau + 1) + 2\hat{b}_m \geq R_m$* **then**

    Decrease τ^* to the maximum possible value such that the estimated resource consumption for remaining iterations is within budget R_m for all m, set STOP flag;

Send $\mathbf{w}(t)$ to all edge nodes;

Receive $F_i(\mathbf{w}(t))$ from each node i;

Compute $F(\mathbf{w}(t))$ according to (2)

if $F(\mathbf{w}(t)) < F(\mathbf{w}^{\mathrm{f}})$ **then**

  $\mathbf{w}^{\mathrm{f}} \leftarrow \mathbf{w}(t)$;

"

Explanation:

This algorithm is designed for a central server in a federated learning system to efficiently manage the training process by coordinating updates from multiple edge devices.

Algorithm 3: Procedure at each edge node i

Initialize $t \leftarrow 0$;
repeat
    Receive $\mathbf{w}(t)$ and new τ^* from aggregator, set $\widetilde{\mathbf{w}}_i(t) \leftarrow \mathbf{w}(t)$;
    $t_0 \leftarrow t$; //Save iteration index of last transmission of $\mathbf{w}(t)$
    if $t > 0$ **then**
        Estimate $\hat{\rho}_i \leftarrow \|F_i(\mathbf{w}_i(t)) - F_i(\mathbf{w}(t))\| \,/\, \|\mathbf{w}_i(t) - \mathbf{w}(t)\|$;
        Estimate $\hat{\beta}_i \leftarrow \|\nabla F_i(\mathbf{w}_i(t)) - \nabla F_i(\mathbf{w}(t))\| \,/\, \|\mathbf{w}_i(t) - \mathbf{w}(t)\|$;
    for $\mu = 1, 2, ..., \tau^*$ **do**
        $t \leftarrow t + 1$; //Start of next iteration
        Perform local update and obtain $\mathbf{w}_i(t)$ according to (4);
        if $\mu < \tau^*$ **then**
            Set $\widetilde{\mathbf{w}}_i(t) \leftarrow \mathbf{w}_i(t)$;
    for $m = 1, 2, ..., M$ **do**
        Estimate type-m resource consumption $\hat{c}_{m,i}$ for one local update at node i;
    Send $\mathbf{w}_i(t)$, $\hat{c}_{m,i}$ $(\forall m)$ to aggregator;
    if $t_0 > 0$ **then**
        Send $\hat{\rho}_i$, $\hat{\beta}_i$, $F_i(\mathbf{w}(t_0))$, $\nabla F_i(\mathbf{w}(t_0))$ to aggregator;
until *STOP flag is received*;
Receive $\mathbf{w}(t)$ from aggregator;
Send $F_i(\mathbf{w}(t))$ to aggregator;

This algorithm outlines the steps each edge node (like a user's smartphone or IoT device) follows to participate in federated learning by locally training a model and communicating updates to the central server