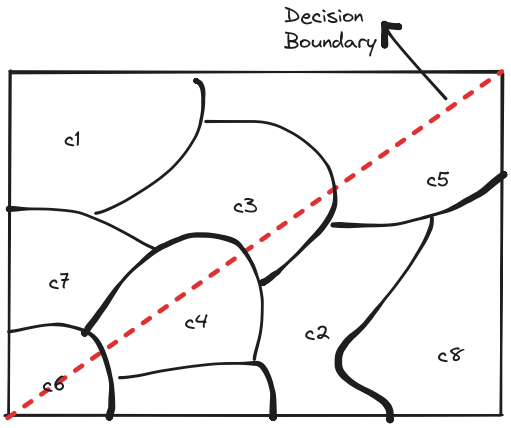
Active Client Selection for Federated Learning

Introduction

Previously, we proposed a novel algorithm, Federated Proximal Sketching (FPS), to reduce communication overhead in noisy, band-limited wireless channels. We now focusing on reducing computational costs by introducing an adaptive client selection strategy. In federated learning, particularly when data is heterogeneously distributed across clients, only a subset of clients significantly contributes to the global model's learning process. To address this, we propose an active client selection strategy that prioritizes clients based on the uncertainty in their data distribution, thereby reducing overall computational complexity.

Drawing inspiration from the field of active learning, we employ entropy as an information-theoretic measure of uncertainty. At each epoch of federated learning, our client selection strategy involves choosing clients with the highest measured uncertainty. We call this approach Uncertainty-based Active Client Selection for Federated Learning (UACS-FL).

To demonstrate the efficacy of our proposed approach (UACS-FL) consider the following illustration of how data is distributed across different clients. From Figure below it is clear that only a fraction of clients (c_3, c_4, c_5 and c_6) contribute significantly to the learning of decision boundary in red.



In order to choose the clients that contribute significant information at each epoch we look to compute the sample entropy at each client c_i. At client c_i, let us denote the dataset as $\mathcal{D}_i$.

The entropy-based metric for a sample x defined as:

$$H(x) = -\sum_{\ell} P_\theta(y_g|x) \log P_\theta(y_g|x)$$

where y_ℓ ranges over all possible labels. We can calculate the entropy metric for client c_i as follows:

$$\alpha_i = \frac{-\sum_{n=1}^{|\mathcal{D}_i|} \sum_g P_\theta(y_\ell^{i,n}|x^{i,n}) \log P_\theta(y_\ell^{i,n}|x^{i,n})}{|\mathcal{D}_i|}. \qquad \text{(Compute sample entropy)}$$

Experimental Objectives

We consider two variants of our proposed approach:

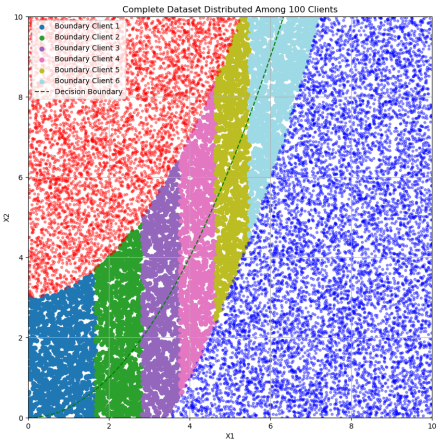
1. Epsilon greedy parameter $\epsilon = 0.1$. In our plots we label this as Entropy+Eps-Greedy.
2. Epsilon greedy parameter $\epsilon = 0 \implies$ Pure exploitation. In our plots we label this as Entropy Based.
3. We compare our proposed approach against the following baseline client selection strategies:

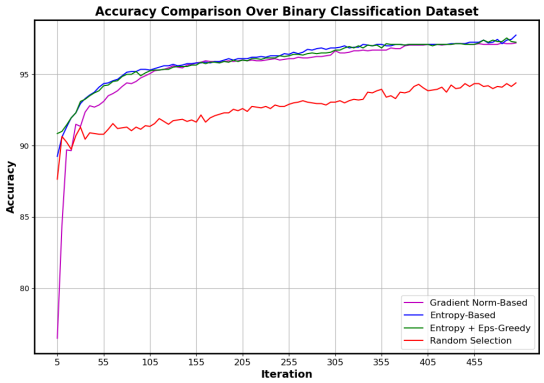
- Random Client Selection (RCS). In our plots we label this as Random Selection.
- Gradient Norm-Based Selection (GNCS): Choosing clients with the largest gradient norm values. In our plots we label this as Gradient Norm-Based.

We demonstrate the performance of our algorithm along with other baselines on three different data distribution settings over a binary classification task. In all three experiments, the number of clients that contribute to efficient learning of the decision boundary is very less ($6 - 10$) compared to the total number of clients (100). In all data distribution plots, dotted green represents the decision boundary.

Experiment 1

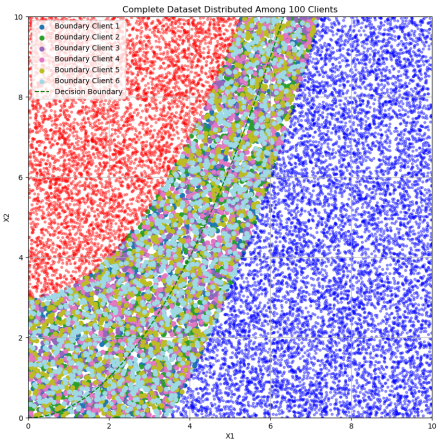
Of the 100 clients only 6 clients have samples around the decision boundary. The distribution of samples at each of these 6 clients is shown below. The remaining 94 clients have access to samples corresponding to only one class (either red or blue).

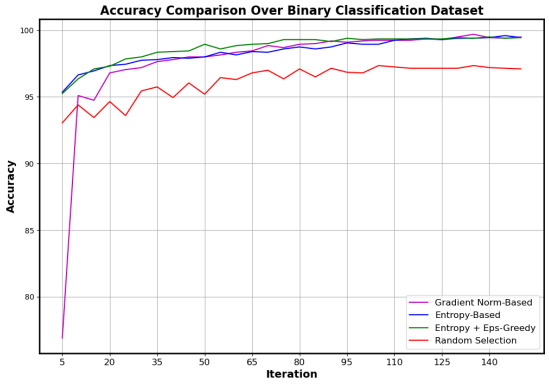




Experiment 2

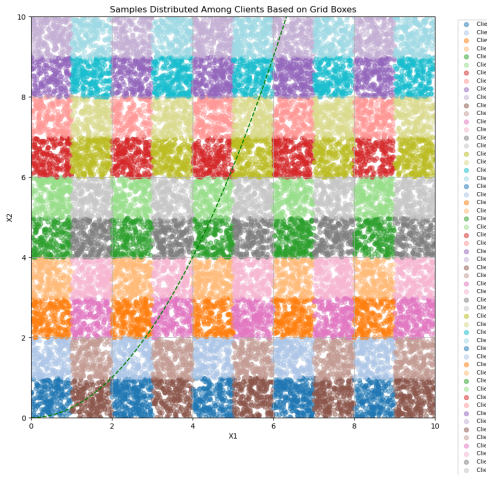
Similar to the previous setting, we distribute data around the decision boundary to only 6 clients, with the difference being that the boundary clients now have a more uniform spread of samples along the entire decision boundary.

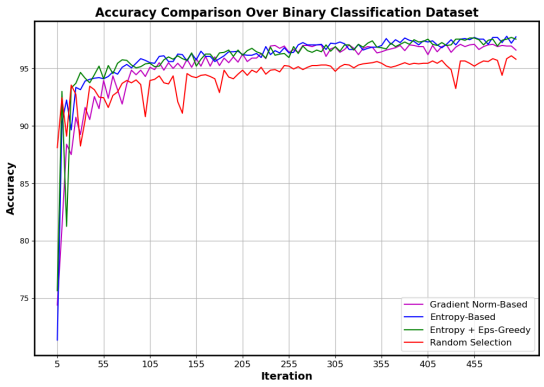




Experiment 3

Lastly, this setup is very similar to experiment 1 where clients around the boundary have samples confined box. Moreover, the remaining clients that are away from the decision boundary also have a restrictive sample space.





Remarks:

- In all three data distribution scenarios, random client selection consistently underperforms in comparison to the proposed method.
- Furthermore, while the gradient norm-based client selection achieves comparable results, the proposed method demonstrates superior performance, notably achieving faster convergence.

Algorithm

We now describe our UACS-FL method. The algorithm is presented below. UACS-FL is based on FedAvg framework and uses AL metric (entropy) to select clients.

Objective

$$\min_{x} \left[f(x) := \sum_{i=1}^{n} \mathbf{w}_i f_i(x) \right],$$

To keep it simple we choose $\mathbf{w}_i = \frac{1}{n} \, \forall i \in [n]$.

Notation

- $\mathbf{w}_t$: Global model parameters at round t.
- $\mathbf{w}_i^{(t+1)}$: Local model parameters of client i after round $t+1$.
- α_i: Selection metric for client i.
- n: Total number of clients.
- m: Number of clients selected per round.
- D_i: Local dataset of client i.

Initialization

1. **Server** initializes the global model parameters $\mathbf{w}_0$.
2. **Server** sets the number of selected clients m per round.

Federated Learning Round

Active Client Selection

- **For each client** $i \in \{1, 2, \ldots, n\}$:
  - **Client** receives global model parameters $\mathbf{w}_t$ from the server.
  - **Client** computes the selection metric α_i based on its local model (see, compute sample entropy equation.).
  - **Client** sends α_i to the server.
- **Server** selects a set S_t of m clients based on epsilon greedy approach:
  - **Step 1:** Generate a random number $r \in [0, 1]$

- **Step 2:**
  If $r \leq \epsilon$ (exploration step):
  - Randomly select m clients from the set $\{1, 2, \ldots, n\}$ without considering α_i.
    Else $r > \epsilon$ (exploitation step):
  - Select the top m clients with the highest α_i values.

Client-Side Computation

For each client $i \in S_t$:
- **Client** updates its local model using its local dataset D_i to compute the local model parameters $\mathbf{w}_i^{(t+1)}$.

Server-Side Aggregation

- **Server** aggregates the parameters from the selected m clients:

$$\mathbf{w}_{t+1} = \frac{1}{m} \sum_{i \in S_t} \mathbf{w}_i^{(t+1)}$$

- **Server** broadcasts the updated global model parameters $\mathbf{w}_{t+1}$ to all clients.