SAPIENZA
UNIVERSITÀ DI ROMA

Optimal Client Selection in Federated Learning based on the Power-Of-Choice Strategy

Department of Computer, Control and Management Engineering (DIAG)

Master's degree (M.Sc.) in Data Science

Candidate

Giovanni Giunta

ID number 1177327

Thesis Advisor

Prof. Ioannis Chatzigiannakis

Co-Advisor

Prof. Andrea Vitaletti

Academic Year 2022/2023

Thesis defended on the 17th of January 2024
in front of a Board of Examiners composed by:

Prof. Ioannis Chatzigiannakis (chairman)
Prof. Danilo Avola
Prof. Stefano Faralli
Prof. Filomena Maggino
Prof. Juri Marcucci
Prof. Iacopo Masi
Prof. Veronica Piccialli

Optimal Client Selection in Federated Learning based on the Power-Of-Choice Strategy
Master's thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: gio.giunta.86@gmail.com

To my family

Contents

Abstract

This thesis presents a novel approach to federated learning, especially in environments where data is non-identically distributed (non-IID) and highly sparse among several different clients. We propose a method that combines the Power-Of-Choice selection strategy with a client clustering approach. This method can be particularly effective in large client networks with an average Hellinger Distance between local datasets above 85%. Under these circumstances, it can enhance the learning efficiency of a given deep neural network, and accelerate its convergence. This study also introduces a few variants to the main implementation, that were designed to reduce the amount of communication messages and communication overhead between the central server and the pool of available clients, although this comes at the cost of a reduced performance in terms of accuracy and loss for the trained model.

The introduction to this thesis provides a comprehensive overview of Federated Learning, delving into its unique challenges, particularly in handling non Independent and Identically Distributed (non-IID) data. It introduces the most common Federated Learning algorithms and explains in detail the centralized dataset employed, and the method used to federate it. It also discusses the deep neural network chosen to train the final model and describes FedLab, the selected Federated Learning framework that facilitates communication between the central server and the available pool of devices. Chapter 1 introduces the Power-Of-Choice strategy for client selection, highlighting its strengths also through a convergence analysis. Chapter 2 explores a clustered approach to Federated Learning, presenting various clustering techniques and an illustrative example that serves as the foundation for our final model: the Heterogeneity-Aware Clustered Client Selection (HACCS). Chapter 3 builds upon the previous chapters by proposing a novel strategy that merges the approaches of Chapters 1 and 2, enabling cluster and client selection in Federated Learning using the Power-Of-Choice strategy. This novel approach aims to strike a balance between efficient learning and fair data representation, addressing the challenges of non-IID data scenarios. The final chapter presents empirical results validating the effectiveness of the proposed strategies across a range of Federated Learning scenarios. These results provide tangible evidence of the impact of our work. The conclusion summarizes our findings and suggests avenues for future research.

Chapter 1

The Power-Of-Choice Client Selection Strategy

1.1 Advantages of a Biased Client Selection

Traditional federated learning systems usually employ partial client selection strategies such as the Uniformly-at-Random Federated Averaging (FedAvg). This conventional approach, while ensuring a democratic and unbiased participation of clients, does not account for data skewness or data heterogeneity in client data distributions. This selection strategy limits the model's efficiency because each client's contribution to the learning process varies significantly in a federated learning environment, where data is distributed across a wide range of clients.

The Power-of-Choice is a new type of partial client selection strategy that introduces a bias when selecting clients. In this thesis, we are going to tailor the Power-Of-Choice to direct this bias towards clients that, at any given point in the learning process,[1] exhibit higher local losses. Clients with higher local losses are targeted because they are likely to provide more substantial updates to the global model,[2] thereby accelerating the learning process. Moreover, Power-Of-Choice has been shown to enhance the overall performance of the model under specific circumstances that will be discussed in section 1.4. The underlying theoretical framework to the Power-Of-Choice strategy can be found in [Mitzenmacher, 2001], where the Author explores the advantages of applying power of two (or more than two) choices to randomized load balancing scenarios, an approach that now finds extensive application in queuing systems.

[1] By definition, a partial client selection strategy selects without replacement a new set of clients at each training round.

[2] If the local loss is high for a specific client, it means that its local data is still relatively unknown to the model. This is something we can exploit.

1.2 Power-Of-Choice Algorithm and Pseudo Code

When the training process starts, the server first asks each client $k \in K$ its current size p_k of local data. These requests can be performed by means of HTTP messages, with p_k inserted in the payload of each message. As a result, we assume that only data collected by the start of the training process will be included. The actual training process consists of n rounds, every round involving j training epochs to be performed by a fixed number $m = C|K|$ of selected clients.[3]

Each round t begins with the server sampling a set of candidate clients, denoted as $A(t)$, from the entire pool of available clients K, hence $A(t) \subseteq K$. The clients are sampled with probability proportional to their share of data.[4] The number of clients sampled, represented by d, is a crucial parameter as it determines the breadth of the selection pool. Note that d is equal to the cardinality of the set $A(t)$, hence $d = |A(t)|$. The server then sends the current global model parameters $w(t)$ to each of the d candidate clients, initiating the second phase of the process. During this phase, each client in set $A(t)$ computes its current local loss $F_k(w(t))$ based on the received model and its local data.[5] All these local losses are then sent back to the server.

The final step in this strategy is the selection of clients for the actual model update. From the candidate set $A(t)$, the server selects m clients, where m is the second key parameter of the Power-Of-Choice. This selection is based on the received local loss values: the server sorts the local losses in descending order and picks the m clients who presented the highest local loss, with ties broken at random. These selected clients constitute the active client set $S(t)$, and are subsequently involved in the training process for the current round. This approach ensures that the most promising clients in terms of local loss will be the ones to actually contribute to the global model updates. Algorithm 1 provides the pseudo code to implement the Power-Of-Choice strategy used in this analysis.

1.3 Role of Parameters d and m

The key parameters in the Power-of-Choice strategy are d and m, each playing a distinct role in shaping the client selection process. On the one hand, the parameter d determines the size of the candidate clients set. A larger d implies a broader range of clients being considered for selection. However, using a large d also introduces some communication overhead, as it requires sharing the model weights with the

[3] Note that while the number m of clients to be trained per round is fixed, the actual m clients involved in each round's training process change.

[4] In other words, the more data a client contains, the more the probability of being selected.

[5] Remember that the local loss is a measure of how well the global model performs on each client's local dataset.

Algorithm 1 Power-of-Choice Algorithm

Input:
d *Number of candidates to sample*
m *Number of clients to be selected for training*
p *Set of* p_k *for each* $k \in K$
Output:
$S(t)$ *Set of clients to be trained at round* t
Step 1: Candidate Selection
$A(t) \leftarrow$ Server samples d clients from K with probability p
Step 2: Model Sharing
for each client i in $A(t)$ **async do**
    Server sends model parameters $w(t)$ to client i
Step 3: Local Loss Computation
Server initializes an empty dictionary D
for each client i in $A(t)$ **async do**
    $F_i \leftarrow$ Client i computes its local loss based on $w(t)$
    Client i sends F_i to the server
    $D[i] \leftarrow$ Server stores F_i in D
Step 4: Client Selection
$S(t) \leftarrow$ Server selects the top m clients with the highest local loss from D
Return $S(t)$

candidate clients and receiving their local loss values at each round t.

On the other hand, m dictates the number of clients eventually chosen from the list of d selected candidates to participate in the model update. Balancing d and m is crucial, as it affects not only the efficiency and speed of convergence, but also the practical feasibility of the strategy in different federated learning environments. However, it is worth mentioning that in most research papers, such as [Wolfrath et al., 2022], the choice of a huge value for d is usually explored without concern. In fact, a large d decreases the efficiency of the model mostly in terms of communication messages and communication overhead between the server and its clients. It is m that actually exploits the computational power of the chosen clients, and is therefore treated more carefully. We will see in section 1.4 that striking a good balance between d and m is also critical to achieve good performances in terms of convergence and accuracy of the model.

1.4 Implementation Considerations

Perhaps the most important aspect of the Power-Of-Choice is the inherent bias introduced by this strategy. By favoring clients with higher local losses, the Power-Of-Choice skews the model towards specific data. This bias, while beneficial for rapid convergence, needs to be managed to ensure that the final model is representative

and fair, especially in non-IID scenarios where data distributions in each client vary substantially. When using the Power-Of-Choice, increasing d while keeping m constant, in fact, leads to faster convergence at the price of a higher bias.[6] The amount of bias is proportional to the gap between d and m. For example, if we set a high value for d, close to $|K|$, while keeping m small, we end up selecting for the training phase the m clients in K that systematically return the highest local loss, hence penalizing the potentially unknown data in the remaining clients. However, by training on clients with a high local loss, the model has the opportunity of focusing its learning on data that guarantees higher margins of improvement. In short, as d increases, we trade off accuracy for training speed.

Such claims have been verified empirically: in Figure 1.1 and 1.2, in fact, we present the results achieved by training our Multilayer Perceptron using the Power-Of-Choice client selection strategy with $m = 3$ and different levels of d, for $n = 150$ rounds and $j = 2$ epochs. The dataset used was Fashion MNIST, federated with a Hellinger distance of 90%, and with $K = 100$.

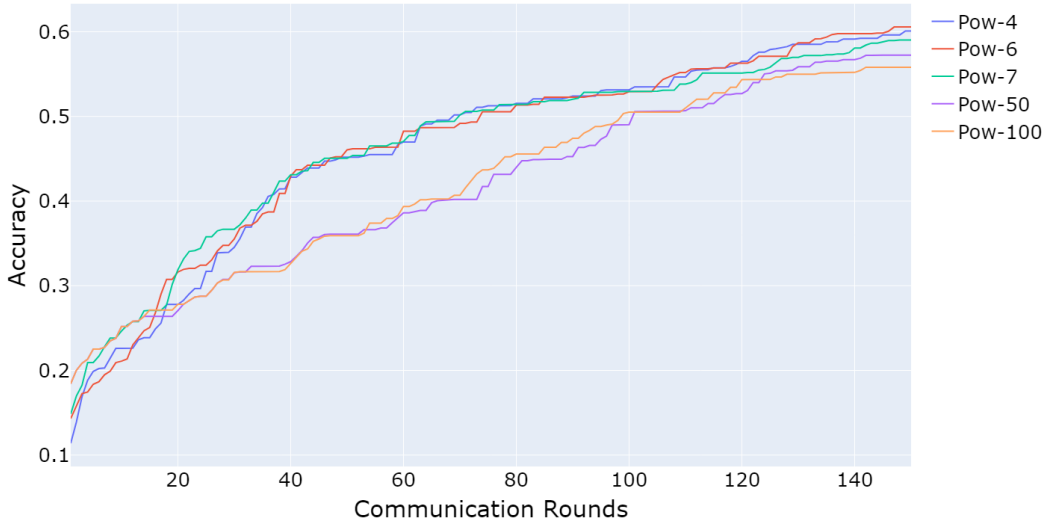


Figure 1.1. Power-Of-Choice - Accuracy - $m = 3$ and various levels of d

In both plots we start with a level of $d = 4$, only one unit above $m = 3$. As we increase d, we gain better performances until we reach $d = 6$. For $d = 6$ and $d = 7$ we have more or less the same results. After that, we stop improving, both in terms of accuracy and loss. What we gain by expanding more and more the pool of candidates is an increasing convergence towards more biased results. This appears even more clear when setting $d = 50$ or $d = 100$.

[6] This can be demonstrated by means of a convergence analysis. See section 1.5.2.

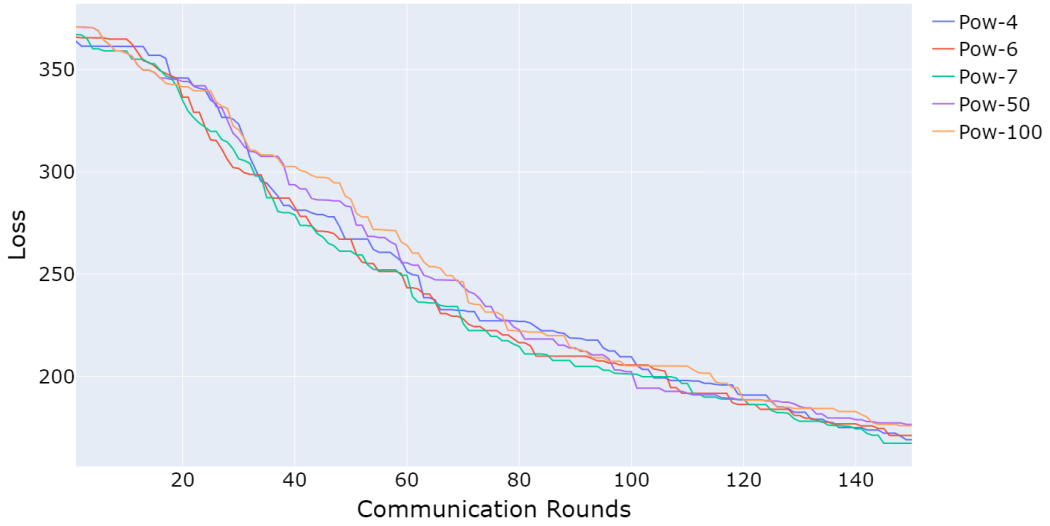


Figure 1.2. Power-Of-Choice - Loss - $m = 3$ and various levels of d

Following the research conducted in [Cho et al., 2020], we also analyzed the performance of the Power-Of-Choice (Pow-d) against the Weighted Random Selection (Rand) and the Uniformly-at-Random Federated Averaging (FedAvg). We did so by training our model on $m = 3$ clients, for $n = 150$ rounds and $j = 2$ epochs per round. The dataset used was again Fashion MNIST, federated with a Hellinger distance of 90%, and with $K = 100$. In Figure 1.3 and 1.4 we present the results achieved by each strategy in terms of Accuracy and Loss.

As we can see, when the ratio between d and m is well balanced, as in the case $d = 6$ and $m = 3$, the Power-Of-Choice can achieve both faster convergence and better results in terms of accuracy and loss with respect to the traditional client selection strategies, despite being trained with the same amount of clients. At this point, it should be evident that balancing efficiency and fairness must be of primary consideration when implementing the Power-Of-Choice. Doing so in practice may involve fine-tuning the parameters of the strategy, particularly d, to find an optimal point where the benefits of faster convergence are realized without compromising the model's ability to generalize.

1.5 Convergence Analysis

In their convergence analysis, [Jee Cho et al., 2022] demonstrate the advantages of adopting a biased client selection strategy. They show that, for any client selection strategy π with partial device participation, a biased client selection can give faster

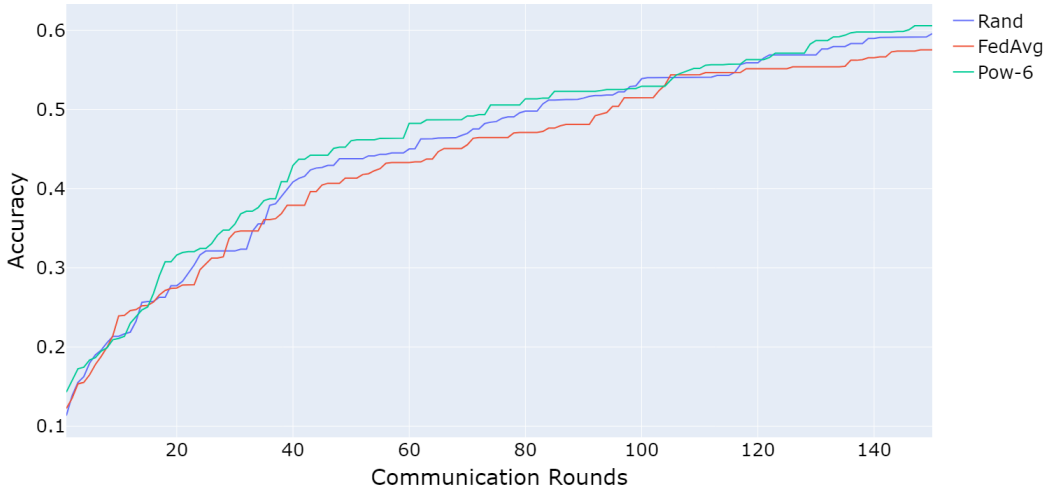


Figure 1.3. Pow-6 vs Rand vs FedAvg - Accuracy - $m = 3$

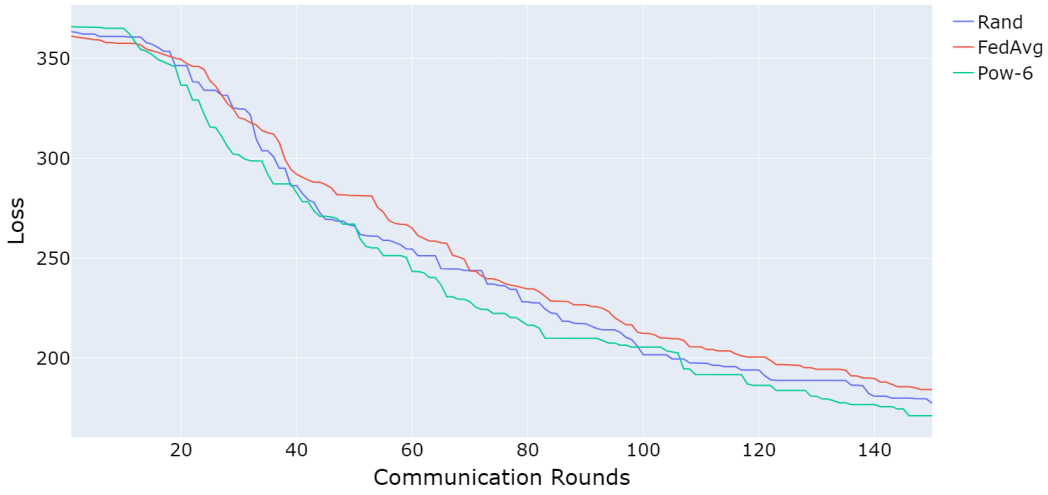


Figure 1.4. Pow-6 vs Rand vs FedAvg - Loss - $m = 3$

convergence at the cost of a non-vanishing error term. This non-vanishing error term can be quantified as the gap between the true optimum set of parameters $\mathbf{w}^* = argminF(\mathbf{w})$ and the set of weights returned by a training process conducted over the highest possible number of rounds n, that is $\lim_{n\to\infty} \bar{\mathbf{w}}^{(n)}$.

1.5.1 Assumptions and Definitions

This convergence analysis is grounded in several key assumptions and definitions:

Assumption 1. *The local objective functions*[7] $F_1, \ldots, F_k$ *are assumed to be* L*-smooth. This smoothness condition implies a certain regularity and boundedness in the gradient of the functions.*

$$F_k(\mathbf{v}) \leq F_k(\mathbf{w}) + (\mathbf{v} - \mathbf{w})^T \nabla F_k(\mathbf{w}) + \frac{L}{2}\|\mathbf{v} - \mathbf{w}\|_2^2 \qquad \forall \mathbf{v}, \mathbf{w} \tag{1.1}$$

Assumption 2. *Each* F_k *is also assumed to be* μ*-strongly convex. Strong convexity is a stronger form of convexity that guarantees a unique minimum and a certain curvature of the function.*

$$F_k(\mathbf{v}) \geq F_k(\mathbf{w}) + (\mathbf{v} - \mathbf{w})^T \nabla F_k(\mathbf{w}) + \frac{\mu}{2}\|\mathbf{v} - \mathbf{w}\|_2^2 \qquad \forall \mathbf{v}, \mathbf{w} \tag{1.2}$$

Assumption 3. *The stochastic gradients, derived from mini-batches* ξ_k *uniformly sampled from the dataset* B_k *belonging to client* $k \in K$*, are unbiased with a bounded variance. These two assumptions are crucial for ensuring that the stochastic gradient descent method converges to the true gradient.*

$$\mathbb{E}[g_k(\mathbf{w}_k, \xi_k)] = \nabla F_k(\mathbf{w}_k) \qquad \forall k \in K \tag{1.3}$$

$$\mathbb{E}\|g_k(\mathbf{w}_k, \xi_k) - \nabla F_k(\mathbf{w}_k)\|^2 \leq \sigma^2 \qquad \forall k \in K \tag{1.4}$$

Assumption 4. *The expected squared norm of the stochastic gradients is uniformly bounded, a condition that helps in controlling the variance of the gradient estimates.*

$$\mathbb{E}\|g_k(\mathbf{w}_k, \xi_k)\|^2 \leq G^2 \qquad \forall k \in K \tag{1.5}$$

We now define two metrics of extreme relevance for the purpose of this convergence analysis: the *local-global objective gap*, Γ, and the *selection skew*, ρ. The first quantifies the gap between the global optimum $\mathbf{w}^*$ and the local optimum $\mathbf{w}_k^*$, the second measures the amount of skew produced by a given client selection strategy π.

Definition 1. *The local-global objective gap* Γ *between the global optimum* $\mathbf{w}^* = argmin_{\mathbf{w}} F(\mathbf{w})$ *and the local optimum* $\mathbf{w^*}_k = argmin_{\mathbf{w}} F_k(\mathbf{w})$ *is defined as:*

$$\Gamma \triangleq F^* - \sum_{k=1}^{K} p_k F_k^* = \sum_{k=1}^{K} p_k (F_k(\mathbf{w}^*) - F_k(\mathbf{w}_k^*)) \geq 0 \tag{1.6}$$

Obviously, if $\Gamma = 0$, the local and global optimum values are consistent, and we deduce that we have no bias in our solution. On the other hand, a large value of Γ

[7] In our case, the local objective functions are the local loss functions.

implies high data heterogeneity in our pool of clients. This first metric is an inherent property of the local and global objective functions, and does not depend on the chosen client selection strategy.

Definition 2. *The selection skew ρ of a partial client selection strategy π is defined as:*

$$\rho(S(\pi, \mathbf{w}), \mathbf{w}') = \frac{\mathbb{E}_{S(\pi,\mathbf{w})}\left[\frac{1}{m}\sum_{k\in S(\pi,\mathbf{w})}(F_k(\mathbf{w}') - F_k^*)\right]}{F(\mathbf{w}') - \sum_{k=1}^{K} p_k F_k^*} \geq 0 \qquad \forall k \in S(\pi, \mathbf{w}) \tag{1.7}$$

The selection skew captures the effect of π on the local-global objective gap Γ. More specifically, it measures the amount of skew the client selection strategy is responsible for in our system. We can read $S(\pi, \mathbf{w})$ as the set of clients returned when applying a given client selection strategy π with parameters $\mathbf{w}$ to our solution. The selection skew $\rho(S(\pi, \mathbf{w}), \mathbf{w}')$ differs depending on π, $\mathbf{w}$ and $\mathbf{w}'$, the latter being the point at which the local and global objective functions are being currently evaluated. At any given point $\mathbf{w}'$, we average the numerator over the randomness introduced by the chosen client selection strategy π, that is $\mathbb{E}_{S(\pi,\mathbf{w})}$.

1.5.2 Convergence Results

From the selection skew introduced in Definition 2 we can now compute a conservative error bound for this convergence analysis:

$$\bar{\rho} \triangleq \min_{\mathbf{w},\mathbf{w}'} \rho(S(\pi, \mathbf{w}), \mathbf{w}'), \quad \tilde{\rho} \triangleq \max_{\mathbf{w}} \rho(S(\pi, \mathbf{w}), \mathbf{w}^*) \tag{1.8}$$

It is evident that, no matter the adopted client selection strategy π, it will always be the case $\bar{\rho} \leq \tilde{\rho}$. From Assumption 1 and Assumption 2 we can also compute the following constant:

$$\gamma \triangleq \frac{4L}{\mu} \tag{1.9}$$

We assume furthermore to be in a Federated Learning scenario where the model selects for training $m = C|K|$ clients per round, and implements a decaying learning rate, re-computed at each round t based on the values of t, γ and μ:

$$\eta_t = \frac{1}{\mu(t + \gamma)} \tag{1.10}$$

We can use all the previous assumptions and definitions to state the following Theorem, from [Jee Cho et al., 2022]:

Theorem 1. *Under Assumptions 1 to 4, Definitions 1 to 2 and Equations 1.8 to 1.10, for any given client selection strategy π, the error after T iterations of federated averaging with partial device participation satisfies the following inequation:*

$$\mathbb{E}[F(\bar{\mathbf{w}}^{(T)}] - F(\mathbf{w}^*) \leq \underbrace{\frac{1}{(T+\gamma)}[\frac{4L(32\tau^2G^2 + \frac{\sigma^2}{m})}{3\mu^2\bar{\rho}} + \frac{8L^2\Gamma}{\mu^2} + \frac{L\gamma\left\|\bar{\mathbf{w}}^{(0)} - \mathbf{w}^*\right\|^2}{2}]}_{\textit{Vanishing Error Term}} + \underbrace{\frac{8L\Gamma}{3\mu}(\frac{\tilde{\rho}}{\bar{\rho}} - 1)}_{\textit{Non-Vanishing Bias}}$$

Now that we have stated Theorem 1, let us consider an unbiased client selection strategy π, such as the Uniformly-at-Random Federated Averaging (FedAvg), where:

$$\rho(S(\pi_{FedAvg}, \mathbf{w}), \mathbf{w}') = 1 \qquad \forall \mathbf{w}, \mathbf{w}' \tag{1.11}$$

The last statement should not come as a surprise if we take into account the definition of Selection Skew.[8] In such case, we have that $\tilde{\rho} = \bar{\rho} = 1$. From Theorem 1 we notice that when $\tilde{\rho} = \bar{\rho} = 1$ the error after T iterations is only represented by the Vanishing Error Term. The Non-Vanishing Bias only comes into play when we introduce a biased Federated Averaging client selection strategy such as the Power-Of-Choice (Pow-d), where $\tilde{\rho} > \bar{\rho} \geq 1$. Such non vanishing bias is proportional to the quantity $(\frac{\tilde{\rho}}{\bar{\rho}} - 1)$. The more π favors clients with a higher local loss $F_k(\mathbf{w}^*)$, the higher the gap between $\tilde{\rho}$ and $\bar{\rho}$ and, consequently, its related non vanishing bias term. However, the higher the bias term, the faster the convergence, which increases at a rate equal to $\mathcal{O}(\frac{1}{T\bar{\rho}})$. We have thus proved what was asserted in section 1.4: the more we bias our selection towards specific clients, as in the Power-Of-Choice, the more we trade off accuracy for speed. A proof of Theorem 1 can be found in [Cho et al., 2020], pp. 14-17.

[8] Note, with an unbiased choice for π, the numerator and denominator in the formula of the Selection Skew in Definition 2 are equal.

Chapter 2

A Clustered Approach to Federated Learning

2.1 Working with Clusters in Unsupervised Learning

Clustering is a fundamental technique in unsupervised learning that aims to identify groups or patterns within unlabeled data. Unlike supervised learning, which requires labeled data with predefined classes, clustering algorithms operate on unlabeled data, seeking to uncover underlying structures and groups based on intrinsic properties or relationships within a given dataset. The core principle of clustering revolves around the notion of similarity. By defining a measure of similarity, clustering algorithms group data points based on their closeness or proximity to each other. This similarity measure could be based on distance metrics, such as the Euclidean distance or the cosine similarity, or on any other measure that could better capture the relationships between data.

By combining related data points into groups, clustering can substantially reduce the overall complexity of a dataset and make it more manageable, especially in the context of big data applications. This methodology not only facilitates trend identification and decision making, but it also helps reduce features in high-dimensional datasets. Clustering can also be used as a preprocessing step to enhance the performance of the entire model. In fact, after clustering, diverse machine learning algorithms can be applied within each cluster, potentially improving the accuracy and efficiency of the model, as each cluster may represent a more homogeneous subset of the data.

Clustering algorithms can also be particularly good at identifying outliers or anomalies in the dataset. By grouping similar data points together, those that do not fit into any cluster stand out as anomalies. In scenarios where labeled data is scarce or expensive to obtain, clustering can provide an initial structure to the data,

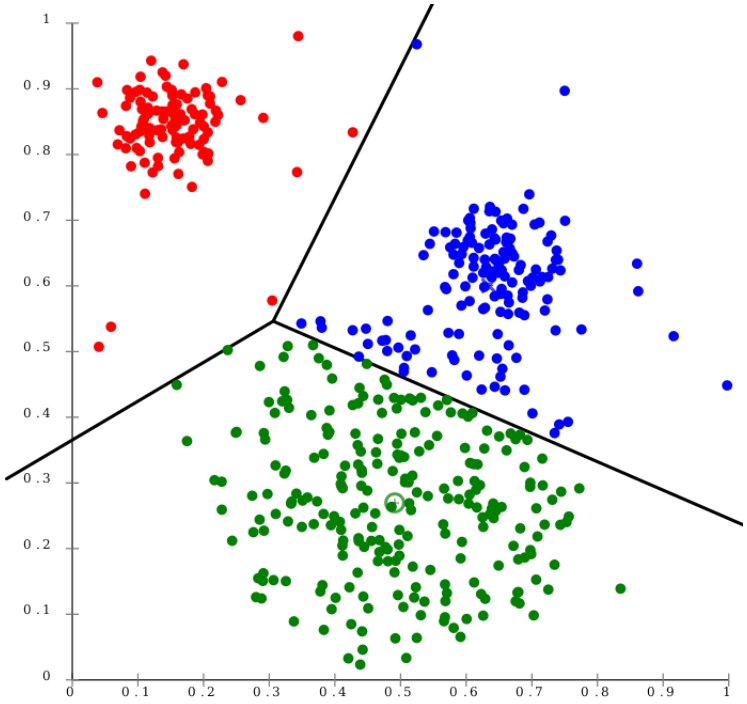


Figure 2.1. Cluster analysis with k-Means on a Gaussian based data set, from [Chire, 2011c]

which can then be used to guide the labeling process. This approach is especially useful in semi-supervised learning, where clustering can help in identifying potential labels or in dividing the data into manageable subsets for further analysis.

2.1.1 Main Clustering Categories

Clustering algorithms can be broadly categorized into several types, each with its unique approach and application domain. The most prominent categories include:

- **Partitioning Methods:** These methods, such as K-Means and K-Medoids, partition the dataset into a predefined number of clusters. They are typically used when the number of clusters is known a priori.
- **Hierarchical Methods:** These methods create a tree of clusters. Agglomerative (bottom-up) and divisive (top-down) are two approaches to hierarchical clustering. They are particularly useful for hierarchical data and for obtaining a dendrogram representing data similarity.
- **Density-Based Methods:** Algorithms like DBSCAN and OPTICS fall into this category. They define clusters as dense regions in the data space, separated by regions of lower density. They are adept at identifying clusters of arbitrary shapes and handling noise.

- **Model-Based Methods:** These methods assume a model for each cluster and try to optimize the fit between the data and the model. Gaussian Mixture Models (GMM) are a typical example.

2.1.2 Challenges in Implementing Clustering

Despite their advantages, clustering techniques can be tricky to implement, as they involve several challenges that can complicate the analysis. One major challenge is determining the "right" number of clusters to produce, which can be ambiguous without prior knowledge of the dataset's underlying structure. An additional challenge is choosing a distance metric. This crucial decision significantly impacts the clustering outcome. In fact, different distance metrics can yield vastly different results, making it important to select the one that best suits the nature of the data. A further challenge is working with large datasets, as clustering algorithms must be able to handle vast amounts of data efficiently without compromising on performance. Scalability is particularly important in high-dimensional data, where the *curse of dimensionality* can hinder traditional distance metrics, making it difficult to accurately capture the true relationships between data points.

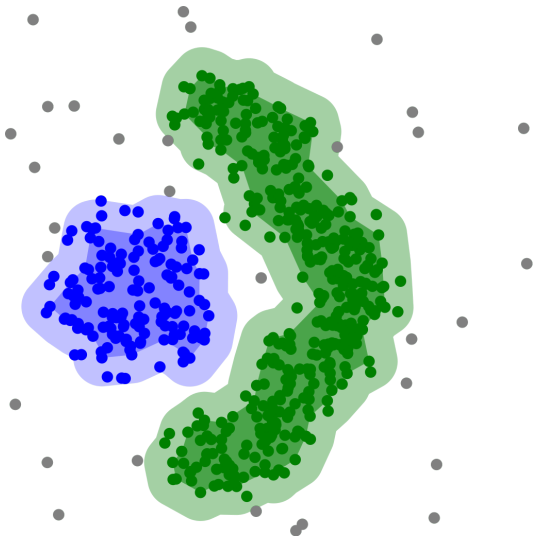


Figure 2.2. Cluster analysis with DBSCAN on a density-based data set, from [Chire, 2011a]

Handling noise and outliers is also critical, as their presence can bias the clustering results, leading analysts to deceiving conclusions. The subjective nature of clustering makes the outcome of the chosen algorithm open to different interpretations, often requiring domain expertise to draw meaningful insights. Additionally, preprocessing steps like feature selection and data transformation play a crucial role in shaping the clustering outcome. These steps need to be carefully considered and executed to ensure that the final results highlight important patterns without obscuring critical

data characteristics. Choosing the right clustering algorithm can also be challenging, as there are many to choose from, each with its own different assumptions and strengths. Selecting the right algorithm and its parameters is essential, and it often depends on the specific data and the current application domain.

2.2 The HACCS Algorithm: A Novel Approach

The Heterogeneity-Aware Clustered Client Selection (HACCS) introduced by [Wolfrath et al., 2022] exploits some of the clustering techniques seen in section 2.1. This client selection strategy represents a significant improvement in terms of resource utilization of Federated Learning devices, as it selects only those clients that have the most diverse data distributions, thus abandoning the traditional paradigm of giving each client the opportunity of being chosen during training. In other words, it skews the selection process to learn more efficiently in a Federated Learning context, while at the same time achieving a better use of resources.

The idea is simple: the HACCS algorithm starts by asking each client to locally compute a summary that identifies its own data distribution. Subsequently, the server receives these summaries and uses them to group the pool of clients into clusters. Eventually, the server can perform its client selection directly on the set of clusters.

Moreover, this approach of identifying similar data distributions preserves privacy. This is because the summaries being computed are designed to be informative of each client's data distribution without exposing sensitive or proprietary data.

2.2.1 HACCS at a Glance

At the core of HACCS are two main components: the Summary Function and the Distance Function. The Summary Function, $S(Z_i)$, executed on each client device, generates a concise representation of the local dataset Z_i. This function strikes a compromise between the need for the server to obtain a synthetic representation of each client's local data, and the general requirements for efficient data transmission and adherence to privacy laws. The Distance Function $d(S(Za), S(Z_b)))$, instead, quantifies the dissimilarity between the collected Summary Functions, eventually enabling the central server to cluster clients based on their data distributions.

Empirical evaluations of HACCS have been conducted in [Wolfrath et al., 2022], including several simulations involving real world datasets[1] and a convolutional neural network. More specifically, the Authors evaluated the performance of their model at first using an implementation of HACCS that determines its cluster structure

[1] In particular, CIFAR-10 and the Fashion Extended MNIST.

based on each client's *distribution of response labels* $P(y)$. Next, they conducted further testing with a different implementation of HACCS that employs each client's *distribution of features conditioned on the response labels* $P(X|y)$. They chose these two specific distribution summaries based on the assumption that traditional distributed machine learning models draw IID samples of response variables y_i and feature variables X_i from the shared joint distribution $P(X, y)$. This joint distribution can be refactored as:

$$P(X, y) = P(X|y)P(y) \tag{2.1}$$

From equation 2.1 we derive that if $P(y)$ or $P(X|y)$ differs at any device, we cannot assume anymore IID-ness, hence one of these distributions must differ across one or more devices. In other words, in non-IID environments, we can rely on either $P(y)$ or $P(X|y)$ to represent how data is distributed in each client. More studies on how to deal with non-IID data in a Federated Learning environment can be found in [Kairouz et al., 2021].

2.2.2 Analysis of HACCS Implementation

Experimenting with HACCS showed how working with clusters in a Federated Learning context can improve the model's performance, particularly in terms of Time To Accuracy.[2] Results also revealed how crucial it is to select clients with distinct data distributions during the training process as opposed to choosing clients at random. For our purposes, it is interesting to note how effective it was to use the distribution summary of response labels $P(y)$ for client clustering. Of course, the degree to which the data in our working environment is distributed and varied determines whether or not strategies like the HACCS can be implemented. In fact, when there is a high degree of non IID-ness, adopting a type of clustered client selection can be particularly fruitful.

2.3 Effective Clustering Methods in Federated Learning

The studies on the HACCS introduced in section 2.2 serve as a basis for the experimental models presented in chapter 3, where the Power-Of-Choice client selection strategy will be performed on top of client clustering. As with the HACCS, we will take into account a distance function to measure how similar (or dissimilar) clients can be in terms of their data distributions. In the following subsections, we will describe the three techniques used in this thesis for client clustering.

[2] For a definition of Time To Accuracy, see section 4.6.

2.3.1 The K-Medoids Technique

In the context of unsupervised learning, K-Medoids is a powerful clustering technique widely recognized for its resilience and effectiveness across diverse applications. Unlike K-Means, which minimizes the sum of squared distances to a central point, K-Medoids focuses on minimizing the sum of dissimilarities between data points within a cluster and a designated representative object called *medoid*. This fundamental distinction makes K-Medoids particularly efficient at handling outliers and noisy data, making it a preferred choice when dealing with non-uniformly distributed data or when anomalies in the distribution are prevalent.

At the heart of K-Medoids lies the identification of k representative objects within a dataset, around which clusters are formed. These representative objects, or medoids, are actual data points, as opposed to the centroids in K-Means, which are often abstract and may not correspond to any real data point. Implementing K-Medoids typically involves the Partitioning Around Medoids (PAM) method, which begins by randomly selecting k objects from the dataset as the initial medoids. The process then iteratively refines the medoids: each object is assigned to the nearest medoid, and the overall cost of the configuration is evaluated. If swapping a medoid with a non-medoid object reduces the total cost, then the swap is made. This iterative process continues until no further improvements are possible, resulting in a locally optimal set of medoids [Kaufman and Rousseeuw, 2009].

The practical applications of K-Medoids are far-reaching and diverse. Its resilience to outliers makes it particularly valuable in domains where data integrity is crucial and anomalies are common.[3] Moreover, since medoids are actual data points, the results of K-Medoids clustering are often more interpretable, a highly sought-after quality in fields like medical diagnosis or customer behavior analysis. Yet, K-Medoids' strength in handling outliers comes with its own set of challenges, primarily computational. The algorithm's complexity, shown in algorithm 2, can be significantly high, especially for large datasets, as it involves calculating and comparing the costs of numerous potential swaps. This computational demand limits its scalability and can hinder its use in big data applications. While various optimizations and adaptations of the PAM algorithm have been proposed to address this issue, the trade-off between computational efficiency and clustering robustness remains a key consideration in employing K-Medoids. Nonetheless, K-Medoids stands out as a robust and reliable clustering technique, especially suited for datasets where outliers cannot be ignored, and where the interpretability of results is crucial. Despite being less advanced than other clustering techniques such as DBSCAN and OPTICS, K-Medoids is still of interest to our research as it has been specifically

[3]For instance, in finance, where fraudulent transactions can distort data, or in marketing segmentation, where extreme consumer behaviors might skew the analysis, K-Medoids can provide a more reliable clustering solution.

designed to work with pre-computed distance matrices[4] like the one we will be using in the next chapter.

Algorithm 2 K-Medoids Clustering Algorithm

```
Input: Dataset D, Number of clusters k
Output: A set of k medoids and clusters
procedure K-Medoids(D, k)
    Initialize: Select k medoids randomly from D
    currentCost ← ∞
    repeat
        Assign each point in D to the closest medoid
        newCost ← Compute total cost of the configuration
        if newCost < currentCost then
            currentCost ← newCost
            for each medoid m do
                for each non-medoid point o do
                    Swap m and o, recompute cost
                    if new cost is lower then
                        Accept the swap
    until no change in medoids
    return Clusters and medoids
```

2.3.2 DBSCAN: A Density-Based Approach

Unlike traditional methods that focus on centroid-based clustering, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) introduces a new approach based on the concept of density. This technique has revolutionized the way experts approach clustering, particularly when dealing with complex datasets that exhibit varied densities and contain significant noise or outliers. DBSCAN operates on the principle of identifying *dense* regions in the data space, and classifies points into three categories: core points, border points, and noise points. A core point has a predefined number of points (MinPts) within a specified radius ε (Eps). Border points are those within the Eps radius of a core point but with fewer neighbors than MinPts. Noise points on the other hand do not meet the criteria to be either core or border points.

The workflow of DBSCAN begins with the arbitrary selection of a point in the dataset. If this point is a core point, the algorithm proceeds to identify all points connected to this core point, with the purpose of forming a cluster. This process involves recursively finding all points that are density-reachable from the core point. If the initial point is not a core point, it's marked as noise, though it may later be found to be a border point of a different cluster. The algorithm iterates through the

[4]More detailed information can be found in [Mark Van der Laan and Bryan, 2003].

Algorithm 3 DBSCAN Clustering Algorithm

Input: Dataset D, Epsilon ε, Minimum Points $MinPts$
Output: Clusters of D
procedure DBSCAN($D, \varepsilon, MinPts$)
    Initialize all points in D as unvisited
    $ClusterId \leftarrow 0$
    for each point P in D **do**
        if P is not visited **then**
            Mark P as visited
            $NeighborPts \leftarrow$ RegionQuery(P, ε)
            if size of $NeighborPts < MinPts$ **then**
                Mark P as noise
            else
                $ClusterId \leftarrow ClusterId + 1$
                ExpandCluster($P, NeighborPts, ClusterId, \varepsilon, MinPts$)
procedure ExpandCluster($P, NeighborPts, ClusterId, \varepsilon, MinPts$)
    Add P to cluster $ClusterId$
    for each point P' in $NeighborPts$ **do**
        if P' is not visited **then**
            Mark P' as visited
            $NeighborPts' \leftarrow$ RegionQuery(P', ε)
            if size of $NeighborPts' \geq MinPts$ **then**
                $NeighborPts \leftarrow NeighborPts$ joined with $NeighborPts'$
        if P' is not yet member of any cluster **then**
            Add P' to cluster $ClusterId$
function RegionQuery(P, ε)
    return all points within ε distance of P

entire dataset, eventually categorizing each point and generating clusters.

DBSCAN's ability to identify clusters of arbitrary shapes makes it exceptionally powerful when dealing with complex data, as in the presence of non-linearly separable data, or when clusters are not globular. Another significant advantage of DBSCAN is its inherent ability to handle noise and outliers.[5] In many real-world datasets, the presence of outliers can significantly skew the results of centroid-based clustering algorithms. DBSCAN, by designating certain points as noise, effectively isolates these outliers, ensuring that they do not influence the formation of clusters.[6]

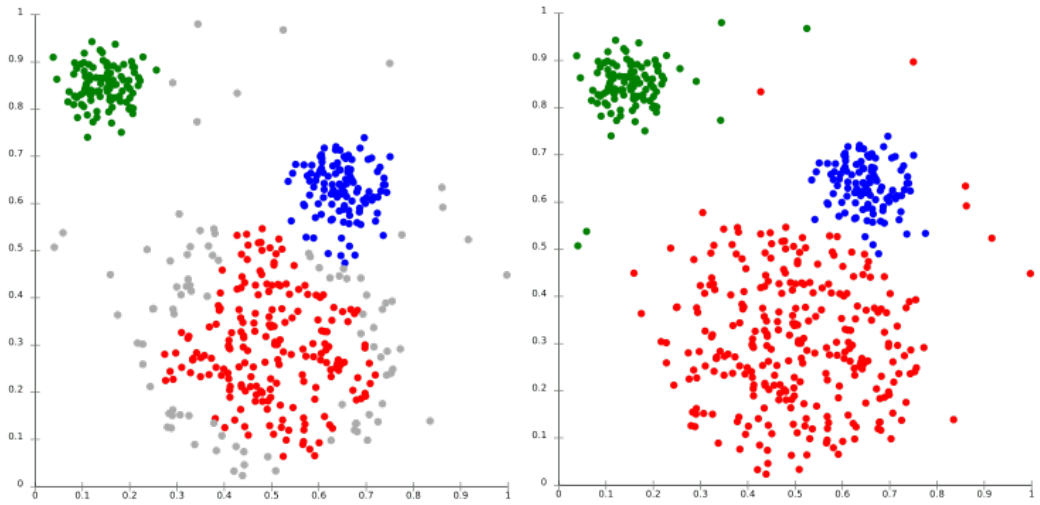
However, DBSCAN can encounter difficulties when dealing with datasets with varying densities, where a single threshold for density may not be appropriate across the entire dataset. Moreover, the algorithm of DBSCAN, shown in algorithm 3, has an effectiveness that is heavily dependent on the chosen values for Eps and MinPts.

[5] See [Ester et al., 1996] for more details on handling noise and outliers with DBSCAN.
[6] This feature is of invaluable importance in domains such as fraud detection, where anomalous points are not just noise but critical signals.

An inappropriate choice of these parameters can lead to over (or under) clustering. Determining the optimal values for Eps and MinPts often requires domain knowledge or experimental tuning, which can be challenging in practice. For this reason, as we will see in section 3.2, a grid search function has been implemented to find the optimal values for the parameters Eps and MinPts.

2.3.3 Clustering using OPTICS

Ordering Points To Identify the Clustering Structure (OPTICS) is a sophisticated clustering algorithm that has the ability to uncover particularly complicated clustering structures from datasets without being heavily dependent on input parameters. OPTICS thus overcomes a limitation often encountered in traditional density-based methods like DBSCAN, which requires a global density threshold to identify clusters. Unlike DBSCAN, OPTICS varies this threshold, allowing it to discover clusters of varying densities.[7] This flexibility makes OPTICS especially adept at handling complex datasets where clusters may not be uniform in terms of their density.



(a) DBSCAN assumes clusters of similar density, and may have problems separating nearby clusters, from [Chire, 2011b]

(b) OPTICS is a DBSCAN variant capable of improving handling of different densities clusters, from [Chire, 2011d]

Figure 2.3. Comparison in terms of clustering efficiency between DBSCAN and OPTICS

The core mechanism of OPTICS revolves around the concept of reachability distance, which is used to identify how dense a particular area of the dataset is. The OPTICS algorithm traverses through each point in the dataset, determining the reachability distance of neighboring points and building an ordered list that captures the density-based clustering structure. This reachability plot serves as a

[7] This characteristic makes OPTICS particularly useful in fields like geographic information systems (GIS), where spatial data often exhibits clusters of different densities, or in bioinformatics, where data can be highly heterogeneous.

visual representation of the clustering structure. It presents the ordering of data points generated by OPTICS along the x-axis, and their reachability distances along the y-axis. Since points within a cluster have relatively short reachability distances to their nearest neighbors, clusters appear as "valleys" in the plot. The deeper the valley, the denser the cluster. The parameter ξ plays a fundamental role in extracting clusters from the reachability plot. It acts as a threshold by defining the level of steepness (or depth of valleys) in the plot that are considered significant for cluster formation. By setting a ξ value, the algorithm distinguishes between significant valleys (representing clusters) and minor fluctuations in reachability distances (which might not correspond to actual clusters). A higher ξ value results in fewer, larger clusters, while a lower ξ value leads to more, smaller clusters [Ankerst et al., 1999].

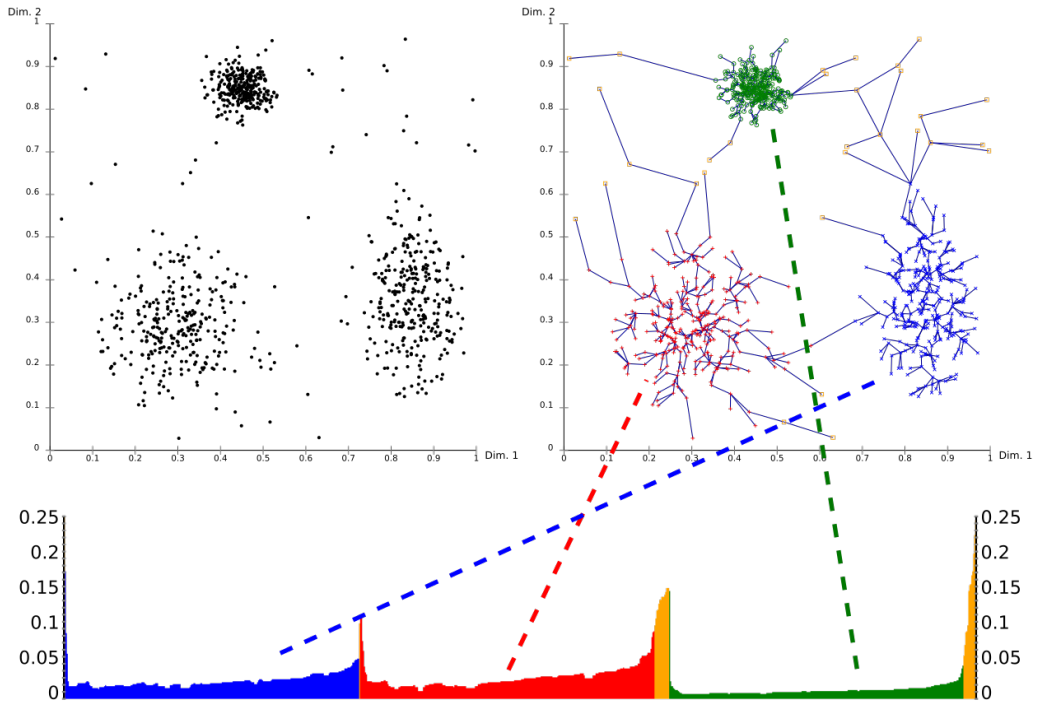


Figure 2.4. OPTICS Reachability Plot, from [Chire, 2010]

Figure 2.4 demonstrates this concept. The upper left portion of the plot shows a synthetic example dataset, while the upper right section visualizes the spanning tree formed by OPTICS. The lower part, instead, displays the reachability plot generated by OPTICS. The colors in the plot are labels assigned to the data points, and are not calculated by the algorithm. It's evident how the valleys in the plot align with the clusters in the dataset. The yellow points in the image represent noise, which don't exhibit distinct valleys in their reachability plots. These points

are typically not assigned to clusters,[8] ensuring that they do not skew the clustering results.[9] OPTICS' ability to create a reachability plot allows for a more nuanced understanding of the dataset structure, providing insights that go beyond mere clustering. The pseudo code to implement OPTICS is shown in algorithm 4.

Algorithm 4 OPTICS Clustering Algorithm

```
procedure OPTICS(DB, ε, MinPts)
    for each point p in DB do
        p.reachability-distance = UNDEFINED
    for each unprocessed point p in DB do
        N = getNeighbors(p, ε)
        mark p as processed
        output p to the ordered list
        if core-distance(p, ε, MinPts) ≠ UNDEFINED then
            Seeds = empty priority queue
            update(N, p, Seeds, ε, MinPts)
            for each next q in Seeds do
                N′ = getNeighbors(q, ε)
                mark q as processed
                output q to the ordered list
                if core-distance(q, ε, MinPts) ≠ UNDEFINED then
                    update(N′, q, Seeds, ε, MinPts)
function UPDATE(N, p, Seeds, ε, MinPts)
    coredist = core-distance(p, ε, MinPts)
    for each o in N do
        if o is not processed then
            new-reach-dist = max(coredist, dist(p,o))
            if o.reachability-distance == UNDEFINED then ▷ o is not in Seeds
                o.reachability-distance = new-reach-dist
                Seeds.insert(o, new-reach-dist)
            else                                  ▷ o in Seeds, check for improvement
                if new-reach-dist < o.reachability-distance then
                    o.reachability-distance = new-reach-dist
                    Seeds.move-up(o, new-reach-dist)
```

However, such algorithm is computationally intensive, especially for large datasets, due to the need to calculate reachability distances for every point. This computational demand can be a limiting factor in its applicability to very large datasets. Additionally, interpreting the reachability plot and converting it into a meaningful clustering result can be non-trivial and may require domain expertise.

[8] Except for the omnipresent "all data" cluster in hierarchical results.

[9] This feature is crucial in datasets where outliers are not just anomalies but may represent critical, rare events.

Chapter 3

A New Strategy for Cluster and Client Selection

3.1 Biased Client Selection Based on Power-Of-Choice and Clustering

In this chapter we are going to merge the Power-Of-Choice selection strategy introduced in section 1.2 with the clustering techniques presented in section 2.2 and 2.3. Grouping our pool of clients into clusters and implementing the Power-Of-Choice strategy allows us to take advantage of the best aspects of both approaches. Indeed, we expect our final model to achieve:

1. *A more balanced data representation.* Thanks to the implementation of a clustering algorithm, we can select only those clients that contain the most diverse and heterogeneous data. In other words, we want to avoid spending rounds and rounds of training on similar data.

2. *A faster and more efficient learning.* We gain this trait by implementing the Power-Of-Choice strategy. Note that faster convergence can be achieved by prioritizing clients according to a chosen criteria. For example, if we choose the local loss as our prioritizing criteria, we can guarantee faster convergence in our training process. This faster convergence can also yield more accurate and more efficient results if a good balance is found between the key parameters d and m of the Power-Of-Choice.[1] The best values for such parameters can be obtained by means of a well performed fine tuning process. Note that in addition to using the local loss, we will also experiment with static types of selection strategies.

[1]For a discussion of this topic see section 1.4; for an explanation involving a convergence analysis see section 1.5.

Each variant of the final model proposed in section 3.3 shares the same clustering algorithm, namely OPTICS,[2] but uses a unique implementation of the Power-of-Choice on top of it. Specifically, we decided to use OPTICS after our experiments determined it was the most effective.[3] Nevertheless, all of these unique Power-Of-Choice implementations follow a common procedure: they all sample a predetermined number of clusters, and then perform client selection within those sampled clusters. However, each variant uses a different algorithm (in terms of complexity or chosen criteria) to sample clusters and clients. We can group all these different Power-Of-Choice implementations into two main categories: *static selection strategies* and *adaptive selection strategies*. In short:

- **Static selection strategies** rely on static types of criteria to perform their selection, such as the amount of data in each client (in the case of client selection), or the average amount of data held by each cluster and/or the number of clients in each cluster (in the case of cluster selection). Such criteria are static in the sense that they do not depend on any metric that evolves during training.
- **Adaptive selection strategies** dynamically sample clusters and clients based on the ongoing value of a chosen metric that evolves during the training process, round after round. All the adaptive algorithms developed in this research use as their dynamic type of metric either the local loss of each client (in the case of client selection), or the average local loss of the entire cluster (in the case of cluster selection), or a mix of the two.

Our goal is to develop a new and more advanced client selection strategy that can effectively handle data in a non-IID scenario, while also achieving faster convergence. Ideally, our results might open the door to more powerful and scalable future federated learning models.

3.2 A Clustering Algorithm for Highly Non-IID Environments

We first developed and tested several distinct clustering algorithms. Based on our results, we determined OPTICS was the best version, as can be seen in the experimental results presented in section 4.2. Our chosen clustering algorithm starts with the central server requesting the distribution of response labels $P(y)$ to each available device $k \in K$. Once all the distributions are collected, the server starts

[2]For detailed information on the OPTICS clustering method, see subsection 2.3.3.

[3]We experimented with all the clustering methods mentioned in section 2.3, namely: K-Medoids, DBSCAN and OPTICS.

computing the set of *Pairwise Hellinger Distance Matrices* between each device and all the other clients $j \in K$, for each $j \neq k$. The result is a set of $|K|^4$ pairwise distance matrices all containing $|K|$ elements.[5]

An example of a *Pairwise Hellinger Distance Matrix* is shown in matrix 3.1. It contains the pairwise Hellinger distances between client "1" and every other available device $j \in K$, where $|K| = 30$. Obviously, the first element of the matrix is equal to 0, as it corresponds to the Hellinger distance between the given client "1" and itself. It is also worth noting that the average value of matrix 3.1 is approximately equal to 0.9. This is not a coincidence: these 30 clients, generated by splitting a centralized Fashion MNIST dataset, contain data that was federated with the FedArtML library by distributing the response labels according to a Dirichlet distribution with an alpha level of 0.05, which in fact corresponds to a Hellinger distance between distributions of 90%.

$$\begin{pmatrix} 0 & 0.4959 & 1 & 1 & 1 \\ 1 & 1 & 0.9353 & 0.9523 & 1 \\ 1 & 0.9674 & 0.9179 & 0.9291 & 1 \\ 0.8592 & 0.369 & 0.943 & 0.8703 & 0.9558 \\ 0.9888 & 1 & 1 & 0.9983 & 0.9267 \\ 0.7701 & 1 & 0.9744 & 0.9926 & 0.9652 \end{pmatrix} \tag{3.1}$$

Apart from the response labels distributions $P(y)$, there are other types of data distributions that can be leveraged to compute the Hellinger distance between clients. In fact, another option is to use the *distribution of features conditioned on the response labels* $P(X|y)$.[6] However, as we will see in section 4.2, this particular data distribution was outperformed by $P(y)$ in terms of results, especially when comparing their *silhouette score*. Therefore, it was not considered for further analysis. Algorithm 5 provides the pseudo code to compute the entire set of pairwise Hellinger distance matrices given a list of data distributions of choice, either $P(y)$ or $P(X|y)$. Every data distribution has to be sent from each client to the central server as a histogram, indicated as the input "hist" in the pseudo code.

Having calculated the set of pairwise Hellinger distance matrices, we can now visualize how the data of each client differs from the data of other clients. This information will be used to identify and group devices with similar data. However, before proceeding, we need to determine the optimal parameter configurations for

[4]Remember that $|K|$ is the cardinality, or size, of the set K.

[5]If we have K available clients, then each $k \in K$ has its own Hellinger Distance matrix containing $|K|$ elements, with every element corresponding to a pairwise Hellinger Distance between the current client $k \in K$ and every other client $j \in K$. Note that if $k = j$, then the corresponding value is 0.

[6]This particular data distribution has been used for clustering in [Wolfrath et al., 2022]. For more information, see section 2.2.

Algorithm 5 Compute Pairwise Hellinger Distance Matrix

procedure ComputePairwiseHellingerDistanceMatrix($hist$)
  $numClients \leftarrow$ length of $hist$
  $distanceMatrix \leftarrow$ zero matrix of size $numClients \times numClients$
  for $i \leftarrow 0$ **to** $numClients - 1$ **do**
    for $j \leftarrow 0$ **to** $numClients - 1$ **do**
      if $i \neq j$ **then**
        $distanceMatrix[i][j] \leftarrow$ HellingerDistance($hist[i], hist[j]$)
  return $distanceMatrix$
function HellingerDistance(p, q)
  $distance \leftarrow \frac{1}{\sqrt{2}} \times \sqrt{\sum (\sqrt{p} - \sqrt{q})^2}$
  return $distance$

every employed clustering algorithm. To achieve this, we developed a few distinct grid search functions, each tailored to the unique characteristics of a specific clustering method. These functions aim to estimate the best parameter values (within a specified portion of the parameter space) for their related clustering algorithm. They do so by exhaustively exploring as many combinations of input parameters as possible, among those needed by that particular clustering technique. The goal is to find that specific combination of parameter values that yields the highest *silhouette score*,[7] which is iteratively calculated based on the clustering results. Additionally, we impose a constraint that at least 70% of clients must be assigned to a cluster, to ensure that the clustering process is meaningful and does not leave an excessive number of clients unlabelled. Specifically:

- The DBSCAN grid search function explores a range of minimum sample sizes (MinPts) and values for epsilon (ε).
- The OPTICS grid search function evaluates alternative combinations of maximum values for epsilon (ε), levels for ξ, and minimum sample sizes (MinPts).
- The K-Medoids grid search function systematically tests various numbers of clusters, different types of distance metrics (pre-computed, euclidean), and various initialization methods (random, k-medoids++, heuristic, build).

Once we had determined the optimal parameters for the proposed clustering algorithms, we proceeded with the actual clustering process. Initially, we tried the K-Medoids technique. However, regardless of the Hellinger distance level imposed during data federation, K-Medoids did not produce satisfactory results, particularly

[7] The silhouette score measures the cohesion of objects within their own cluster and the separation between clusters. It provides a concise measure of how effectively data points have been clustered, making it a valuable tool for evaluating and comparing different clustering models. The score ranges from -1 to 1, with a high value indicating that the object is well matched to its own cluster and poorly matched to neighboring clusters.

with respect to its *silhouette score*. For this reason, it was discarded. We then tested OPTICS and DBSCAN, which produced superior findings, although similar to each other. This similarity is likely due to the fact that both of these clustering techniques are density-based. Only after observing the results in chapter 4 it became evident that OPTICS was a better option than DBSCAN, as it produced better results in terms of accuracy and loss. For more details on the empirical results achieved on clustering, see section 4.2.

3.3 Static and Adaptive Cluster and Client Selection Strategies

This section synthesizes the proposed selection algorithms for the final model, each of which is based on the Power-Of-Choice strategy. They all need as input two key parameters:

- The first parameter is w, which is the number of clusters to be sampled at random at every round t. By construction, every strategy presented in this thesis eventually selects only one client from each sampled cluster, therefore, w can also be considered as the number of clients to be trained at every round.[8]
- The second parameter is d, which corresponds to the total number of candidates to be drawn from all the w sampled clusters. It is equivalent to the d parameter of the standard Power-Of-Choice strategy. Given that d represents the total number of candidates regardless of the size of w, a value $z = \left\lceil \frac{d}{w} \right\rceil$ is computed when the algorithm is initialized to determine how many candidates to select from each sampled cluster. The objective is to distribute the d candidates as evenly as possible among all the sampled clusters. Note that for the experiments shown in chapter 4 we chose values of d that are multiples of w. This is not a strict requirement, as our final model is capable of working effectively even when the distribution of z candidates across clusters is not uniform. Nevertheless, we made this choice to compare our model's results with those returned by traditional versions of the Power-Of-Choice strategy. Given that $w = m$, by using the same value of d in both implementations we know that the end results will be comparable.

In 3.3.1 we present our first algorithm proposals. The goal was to understand the impact of employing static selection strategies when training our Multilayer Perceptron on clusters of clients. After that, we will study more sophisticated techniques, and introduce a few additional client selection variants based on adaptive types of selection strategies. These adaptive variants are shown in 3.3.2.

[8] Using the Power-Of-Choice notation, we have that w is equivalent to m.

3.3.1 Static Selection Strategies

As mentioned, we did not have high expectations for these first attempts. The primary objective of these experiments was to determine whether the training of our Multilayer Perceptron could be enhanced by simply exploiting the data heterogeneity and the static features of individual clients. We evaluated three distinct implementations of static selection strategies, which can be distinguished by the following unique cluster sampling methods:

1. Cluster sampling with clusters drawn uniformly at random. In this case, they all have the same probability of being selected (Uniform-Oriented).
2. Cluster sampling with probability proportional to the average data size of their clients (Data-Oriented).
3. Cluster sampling with probability proportional to their number of clients (Client-Oriented).

Once the algorithm selects w clusters, it randomly draws z clients from each cluster, ensuring that each client is chosen with a probability proportional to its share of data p_k. To achieve this, the algorithm normalizes the size of local data in every client within the cluster. By this point, the server has collected a total amount of $d = z \cdot w$ clients. Eventually, these devices are sorted by their size of p_k, in descending order. The server then selects the top w clients that possess the highest size of local data: these are going to be the devices involved in the current round of training. It is worth noting how, regardless of the chosen cluster sampling strategy, the client selection process prioritizes clients based on their local data size.

However, the results collected from the static selection strategies were not encouraging. The best results were produced by training with the Data-Oriented variant, listed above as number 2. Still, it could not match the performances returned by a standard Power-Of-Choice with equivalent values of d and m. Therefore, we abandoned our studies on static selection strategies, and focused on adaptive kinds of selections based on local loss. For the sake of completeness, we nonetheless include the plots representing the Accuracy (Figure 3.1) and Loss (Figure 3.2) returned by the Data-Oriented static strategy and the standard Power-Of-Choice, both initialized with the same configuration of d and m. The training was conducted on a federated Fashion MNIST dataset with a Hellinger Distance of 90% and $K = 100$, for $n = 150$ rounds, with $j = 2$ epochs per round.

3.3.2 Adaptive Selection Strategies

Each of the five strategies presented in this subsection involve, to some degree, a biased selection towards the highest local losses returned by either the set of available

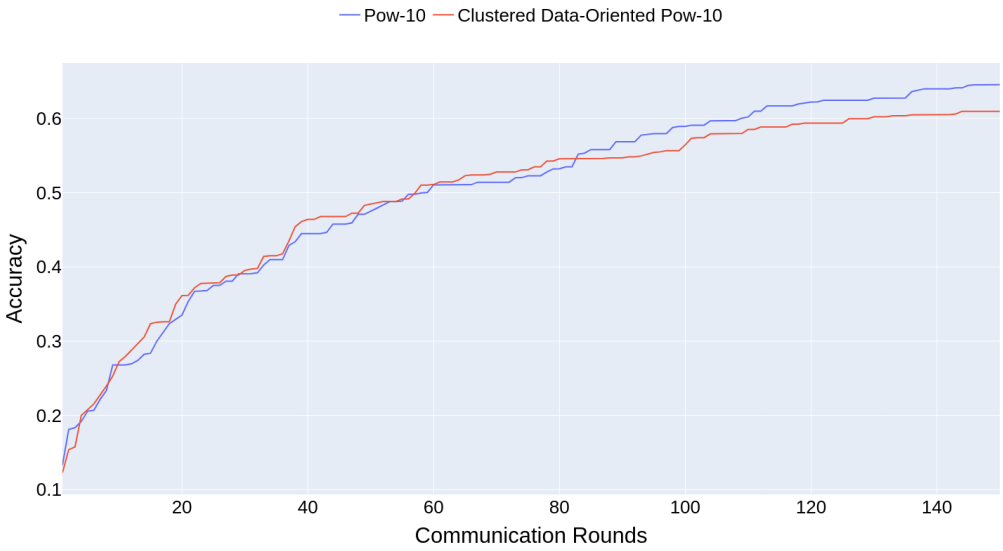


Figure 3.1. Clustered Data-Oriented Pow-d vs Pow-d - Accuracy - $d = 10$, $m = 5$

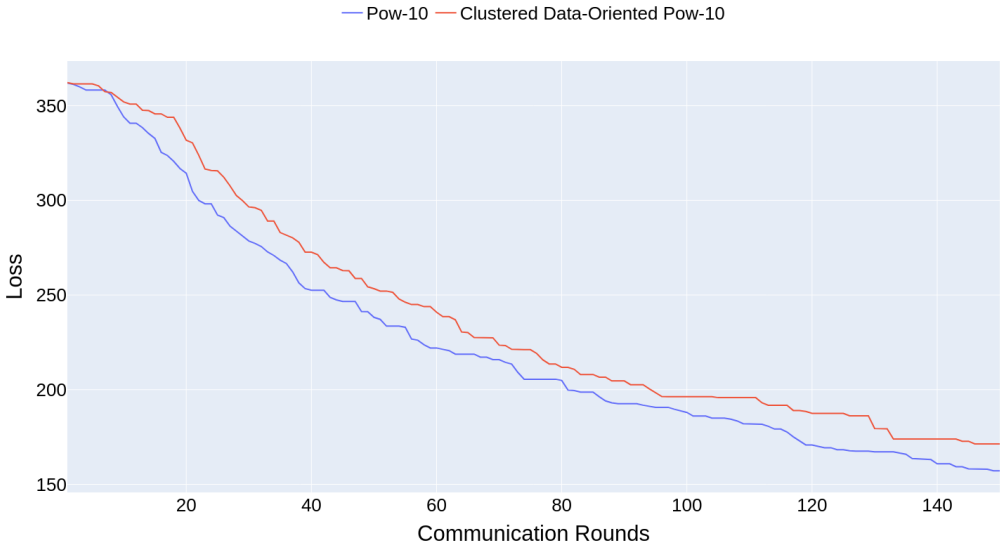


Figure 3.2. Clustered Data-Oriented Pow-d vs Pow-d - Loss - $d = 10$, $m = 5$

clients K, or by a subset of K. The bias in each implementation varies based on the degree to which the selection towards highest local losses is mitigated by additional computations. In contrast to the static selection strategies, which employed distinct cluster sampling algorithms and a common client selection based on local data sizes, the adaptive selection implementations introduced below present unique strategies for both cluster and client sampling.

1. **Clustered Data-Loss Pow-d**: This strategy begins by randomly selecting a number w of clusters, with each cluster's probability of being chosen proportional to the average data size of its clients. From each of the w clusters, z candidate clients are selected. If a cluster has fewer than z clients, additional clients are chosen at random from other clusters (not in w), weighted by their data size. The server requests local loss computations from these clients, and eventually selects the one with the highest loss from each cluster. As a result, we end up with w clients for training at each round.

2. **Clustered Average-Loss Pow-d**: Initially, every client computes its local loss, and shares the result with the central server. The server then selects w clusters based on the average local loss of the clients they contain. Within each cluster, z clients are chosen based on their local loss. If a cluster has fewer than z clients, the remaining slots are filled with clients from other clusters (not in w), selected based on their current local loss. The highest-loss client from each cluster is then selected for training, for a total of w clients.

3. **Clustered Best-Loss Pow-d**: After each client has computed and sent its local loss to the central server, the server calculates the average local loss for each cluster. Clusters are then sorted by this average loss in descending order, and the top w clusters are selected. From each of these clusters, the top z clients with the highest local loss are chosen. If any cluster has fewer than z clients, additional clients are selected from other clusters (not in w) based on their local loss. The highest-loss client from each cluster is selected for training, for a total of w clients.

4. **Clustered Average-Loss-X Pow-d**: This strategy starts by following the Average-Loss approach for the initial x rounds. After these rounds, it transitions to the Data-Loss approach. In other words, the initial phase of x rounds focuses on selecting clusters with the highest average local loss, while the subsequent phase reduces the model's computational demands by randomly selecting clusters based on their data size, hence computing the local loss only for $d = w \cdot z$ clients per round.

5. **Clustered Best-Loss-X Pow-d**: This version operates like the Best-Loss for the first x rounds, focusing on clusters and clients with the highest local loss. Starting from round $x + 1$, it switches to the Data-Loss approach. This hybrid method aims to boost training efficiency for the first x rounds, after which it transitions to a more sustainable and less communication-intensive approach.

The Data-Loss strategy is the less focused on the local loss, as it starts by simply considering clusters and clients containing the highest amount of data. Only then, the server requests a local loss computation from the limited amount of $d = w \cdot z$

Algorithm 6 Clustered Data-Loss Pow-d

```
1: procedure DATA-LOSS(w, d)
2:     Compute z from d and w
3:     Randomly select w clusters, probability ∝ average data size
4:     Initialize an empty list for selected clients
5:     for each selected cluster do
6:         Select z candidate clients from the cluster
7:         if number of clients in cluster < z then
8:             Fill remaining slots by selecting clients from clusters not in w
9:             Selection based on clients' data size
10:        Request local loss computation from each candidate client
11:        Receive local losses and sort candidates by loss in descending order
12:        Select the client with the highest local loss from the cluster
13:        Add selected client to the list of selected clients
14:    return list of selected clients for training
```

Algorithm 7 Clustered Average-Loss Pow-d

```
1: procedure AVERAGE-LOSS(w, d)
2:     Compute z from d and w
3:     Request local loss computation from each client in all clusters
4:     Receive local losses for each client
5:     Compute the average local loss for each cluster
6:     Randomly select w clusters, probability ∝ average local loss
7:     Initialize an empty list for selected clients
8:     for each selected cluster do
9:         Select z candidate clients from the cluster, based on local loss
10:        if number of clients in cluster < z then
11:            Fill remaining slots by selecting clients from clusters not in w
12:            Selection based on clients' local loss
13:        Sort candidates by local loss in descending order
14:        Select the client with the highest local loss from the cluster
15:        Add selected client to the list of selected clients
16:    return list of selected clients for training
```

candidates. Instead, the Average-Loss and the Best-Loss variants need to ask each available client in K for such computation at the beginning of every round. To address this challenge, we developed the Average-Loss-X and Best-Loss-X variants. Both these variants adopt a hybrid approach: they start by following their original implementation (either Average-Loss or Best-Loss) and then, after a chosen number of rounds x, they fall into the Data-Loss approach. The goal is to start with an intensive learning phase, aimed at "boosting" the first x rounds of training, in order to improve quickly the overall performance of the model. Then, starting from $x + 1$, there is a transition to the less demanding Data-Loss method. We tested this approach also because the Data-Loss selection is less skewed, and not

as loss-driven, as the Average-Loss and the Best-Loss. It is important to highlight that the Best-Loss implementation is even more loss oriented than the Average-Loss, as it directly selects the clients with the highest local loss, rather than sampling them at random. By transitioning to a Data-Loss approach, we can switch to a more balanced method of client selection, refocusing on less significant data. This shift allows us to leverage the diversity inherent in such data, which may have been previously overlooked due to the emphasis on loss.

Algorithm 8 Clustered Best-Loss Pow-d

```
1:  procedure BEST-LOSS(w, d)
2:      Compute z from d and w
3:      Request local loss to each client in all clusters
4:      Receive local loss from each client in all clusters
5:      Calculate average local loss for each cluster
6:      Sort clusters by average local loss in descending order
7:      Select top w clusters with the highest average local loss
8:      Initialize an empty list for selected clients
9:      for each cluster in top w clusters do
10:         Select top z clients with highest local loss in the cluster
11:         if number of clients in cluster < z then
12:             Fill remaining slots by selecting clients from clusters not in w
13:             Selection based on clients' local loss
14:         Sort candidate clients by local loss in descending order
15:         Choose the client with the highest local loss
16:         Add selected client to the list of selected clients
17:     return list of selected clients for training
```

Notice how all these adaptive selection strategies share a common trait: for all of them, we split the amount of d candidates into several z, one for each of the w clusters. By doing so, we enforce equal lists of candidates for each cluster, a constraint that cannot always be satisfied, due to the unpredictable size of each sampled cluster. For this reason, we fill potential gaps by sampling at random from clusters that do not belong to w. Given that there will be only one chosen client out of each list of z candidates (that is the one with the highest local loss), this strategy *willingly discriminates against under-represented clusters*. Unless, of course, the chosen small clusters yield a relatively high level of local loss, in order to "beat" the competition coming from other clusters. This is an important feature, shared by each of the five implementations, and will be key for enhancing their performance.

As we will see in chapter 4, despite providing interesting results, the hybrid approaches could not match the performance of their original implementations. Instead, the Best-Loss variant proved to be the best selection strategy for our final model, with results capable of challenging, and in most cases even outperforming,[9] a state of

[9]Before convergence, the Best-Loss Pow-d selection strategy outperforms Pow-d at any level of d.

Algorithm 9 Clustered Average-Loss-X

```
1: procedure AVERAGE-LOSS-X(w, d, x, t)
2:     Compute z from d and w
3:     if t ≤ x then
4:         // Use Average-Loss Based Cluster Sampling for initial rounds
5:         Request local loss computation from each client in all clusters
6:         Receive local losses for each client
7:         Compute the average local loss for each cluster
8:         Randomly select w clusters, probability ∝ average local loss
9:         for each selected cluster do
10:            Select z candidate clients based on local loss
11:            if number of clients in cluster < z then
12:                Fill remaining slots from other clusters based on local loss
13:        Choose the highest-loss client from each cluster
14:    else
15:        // Switch to Data-Size Based Cluster Sampling
16:        Randomly select w clusters, probability ∝ average data size
17:        for each selected cluster do
18:            Select z candidate clients from the cluster
19:            if number of clients in cluster < z then
20:                Fill remaining slots from other clusters based on data size
21:            Request local loss computation from each candidate client
22:            Receive local losses and select the highest-loss client
23:    return list of selected clients for training
```

the art implementation such as the Power-Of-Choice. Empirical proofs are in chapter 4, including results from the Data-Loss and the Average-Loss implementations, and from the Best-Loss25 and the Average-Loss25 variants (hence, from a Best-Loss-X and an Average-Loss-X with $x = 25$). Note that we chose the name "Best-Loss" as it possible to switch the focus of the selection towards clients returning lowest loss, rather than highest loss. Experiments in such direction are left to future research.

3.4 Towards a Mathematical Validation of Our Strategy

This section is the result of a collaboration with Luca Becchetti, from the DIAG department,[10] and Andrea Vitaletti, co-advisor to this thesis. Here, we try to develop a mathematical foundation for the approaches we have adopted in our research.

In 3.4.1 we prove the following preliminary claim: if we sample from a set X, composed of n integers, a subset Q_1 of k elements, and we compute the expected

At convergence, it proved having at least as good results as Pow-d.

[10] Dipartimento di Ingegneria Informatica, Automatica e Gestionale of La Sapienza University of Rome.

Algorithm 10 Clustered Best-Loss-X

```
1: procedure BEST-LOSS-X(w, d, x, t)
2:     Compute z from d and w
3:     if t ≤ x then
4:         // Use Best-Loss Clustered Power of Choice for initial rounds
5:         Randomly select w clusters, probability ∝ average data size
6:         Initialize an empty list for selected clients
7:         for each selected cluster do
8:             Select z candidate clients from the cluster
9:             if number of clients in cluster < z then
10:                Fill remaining slots by selecting clients from clusters not in w
11:                Selection based on clients' data size
12:            Request local loss computation from each candidate client
13:            Receive local losses and sort candidates by loss in descending order
14:            Select the client with the highest local loss from the cluster
15:            Add selected client to the list of selected clients
16:    else
17:        Randomly select w clusters, probability ∝ average data size
18:        Initialize an empty list for selected clients
19:        for each selected cluster do
20:            Select z candidate clients from the cluster
21:            if number of clients in cluster < z then
22:                Fill remaining slots by selecting clients from clusters not in w
23:                Selection based on clients' data size
24:            Request local loss computation from each candidate client
25:            Receive local losses and sort candidates by loss in descending order
26:            Select the client with the highest local loss from the cluster
27:            Add selected client to the list of selected clients
28:    return list of selected clients for training
```

value of the sum of the k integers in Q_1, then the result will be greater than, or at most equal to, the expected value of the sum of the k smallest integers taken from another subset Q_2 composed of r integers sampled from X, with $k \leq r$. In short, we demonstrate that if we apply the Power-Of-Choice by choosing the k smallest elements from a sample of r elements, we will have a smaller or at most equal expected value than by simply choosing at random k elements. If instead of the lowest we assume to sum the k biggest integers in Q_2, the opposite is also going to be true.

In 3.4.2 we use the above claim to prove that the same principle holds true when applied to any chosen couple of clusters within a system of disjoint clusters. Such system of disjoint clusters obviously represents the one that the central server generates when clustering our pool of clients. Despite an encouraging start, this research is currently only applicable to systems of clusters all of fixed size n. Further studies are needed to generalize these claims to clusters of possibly different sizes.

3.4.1 Preliminaries

Consider a set $X \subset \mathbb{N}$, such that $|X| = n$. Consider the following two ways of computing a random integer value:

1. We sample a subset $Q_1 \subset X$, $|Q_1| = k$, *without replacement*. We return $S_1 = \sum_{x \in Q_1} x$.

2. We sample a subset $Q_2 \subset X$, $|Q_2| = r$, *without replacement*, with $k \leq r < n$. We return $S_2 = \sum_{x \in Q_2(k)} x$, where $Q_2(k)$ denotes the subset of the k smallest values in Q_2.

We are interested in the following question. Is it true that:

$$\mathbb{E}[S_2] \leq \mathbb{E}[S_1]$$

Although intuition suggests that the answer should be affirmative, proving this claim might seem impervious at first. We first note that sampling a subset of X of size $m \leq n$ without replacement is accomplished by the sequential process described in Algorithm 11.

Algorithm 11 Iteratively Sampling Without Replacement

Input: The original set X, The size of the subset to sample m
Output: Q (The sampled subset of X)
$Q \leftarrow X$
$S \leftarrow \emptyset$
for $t = 0$ to m **do**
    Sample a value v uniformly at random from U
    $Q \leftarrow Q \cup \{v\}$
    $U \leftarrow U - \{v\}$
return Q

Indeed, Algorithm 11 samples a subset of X of size m uniformly at random, as stated by the following well-known fact, whose proof is only given for the sake of completeness.

Fact 1. *Any subset $S \subset X$ of size m is sampled with probability $1/\binom{n}{m}$ by Algorithm 11.*

Proof. Consider any $S \subset X$, $|S| = m$. The probability of sampling the elements of

S in any particular order is exactly

$$\frac{1}{n} \cdot \frac{1}{n-1} \cdots \frac{1}{n-m+1} = \frac{1}{\prod_{i=0}^{m-1} (n-i)}$$

Moreover, there are exactly $m!$ different orderings in which elements of S can be sampled. As a consequence, the probability of sampling S (with its elements sampled in any of the possible alternative $m!$ orderings) is

$$\frac{m!}{\prod_{i=0}^{m-1} (n-i)} = \frac{m!(n-m)!}{n!} = 1/\binom{n}{m}$$

□

Now, consider the following two ways to compute the sum of k elements sampled from X:

Process $\mathcal{P}_1$:

1. Sample a subset $Q_1 \subset X$ of size k, uniformly at random, using algorithm 11.
2. Compute the sum $\hat{S}_1 = \sum_{x \in Q_1} x$, which is the sum of all elements in the subset Q_1.

Process $\mathcal{P}_2$:

1. Start with the subset Q_1 obtained from Process $\mathcal{P}_1$.
2. Extend this subset to form Q_2 by sampling additional $r-k$ elements uniformly at random from the set $X - Q_1$ (i.e., from X excluding elements already in Q_1), ensuring no duplication.
3. From the resulting set Q_2, identify $Q_2(k)$, which is the subset of the k smallest elements within Q_2.
4. Compute the sum $\hat{S}_2 = \sum_{x \in Q_2(k)} x$, which is the sum of all elements in the subset $Q_2(k)$.

The random variables $\hat{S}_1$ and $\hat{S}_2$ are correlated due to the shared elements in the initial subset Q_1. This shared subset influences both sums $\hat{S}_1$ and $\hat{S}_2$, creating a statistical dependence between them. In particular, each of the pairs (Q_1, Q_2) and $(\hat{S}_1, \hat{S}_2)$ is a coupling in the sense described in [Wilmer et al., 2009]. In particular:

1. It is clear that, while $\hat{S}_1$ and $\hat{S}_2$ are correlated, their marginal distributions $\mathbb{P}(\hat{S}_1 = y)$ and $\mathbb{P}(\hat{S}_2 = y)$ are the same as $\mathbb{P}(S_1 = y)$ and $\mathbb{P}(S_2 = y)$ respectively.

2. By design, we have $\hat{S}_2 \leq \hat{S}_1$ deterministically.

Together, 1 and 2 imply:

$$\mathbb{E}[S_2] = \mathbb{E}[\hat{S}_2] \leq \mathbb{E}[\hat{S}_1] = \mathbb{E}[S_1]$$

We have thus given a (constructive) proof of the following claim:

Claim 1. *Consider S_1 and S_2 defined above. Then $\mathbb{E}[S_2] \leq \mathbb{E}[S_1]$.*

3.4.2 Clustering

Assume a set X which is the union of m disjoint subsets, i.e., $X = \bigcup_{i=1}^{m} Q_i$. Each subset Q_i is distinct and non-overlapping with Q_j for all $i \neq j$ (denoted as $Q_i \cap Q_j = \emptyset$), and every subset contains exactly n elements, expressed as $|Q_i| = n$. Now, consider the two following sampling processes, where $c = \frac{k}{m}$ is an integer.

Process $\mathcal{P}_1$:

1. Sample k elements from Q_i, uniformly at random and without replacement, for every $i = 1, \ldots, m$. Then, compute the sum of the k sampled items, denoted by $S_1(Q_i)$.
2. Return $S_1 = \sum_{i=1}^{m} S_1(Q_i)$.

Process $\mathcal{P}_2$:

1. For each subset Q_i (where i ranges from 1 to m), sample r items (with $r \geq \frac{k}{m}$) from Q_i without replacement.
2. For each Q_i, calculate $S_2(Q_i)$, which is the sum of the c smallest items in the sampled set, where $c = \frac{k}{m}$.
3. Return $S_2 = \sum_{i=1}^{m} S_2(Q_i)$.

Furthermore, the number of items sampled from each Q_i in Process $\mathcal{P}_1$ is a random variable denoted by Z_i. We begin with the following

Claim 2. *For every $i = 1, \dots, m$ we have:*

$$\mathbb{E}[S_1(Q_i)|Z_i = c] = \mathbb{E}[S_1(Q_i)] = \frac{cS}{n}$$

where $S = \sum_{x \in Q_i} x$

Proof. Consider the generic $x \in Q_i$. Conditioned to $Z_i = \ell$, we have:

$$\mathbb{P}(\mathcal{P}_1 \text{ samples } x|Z_i = \ell) = \frac{1}{n} + \left(1 - \frac{1}{n}\right)\frac{1}{n-1} + \left(1 - \frac{1}{n}\right)\left(1 - \frac{1}{n-1}\right)\frac{1}{n-2}$$
$$+ \cdots + \left(1 - \frac{1}{n}\right)\left(1 - \frac{1}{n-1}\right)\cdots\left(1 - \frac{1}{n-\ell+2}\right)\frac{1}{n-\ell+1} = \frac{\ell}{n}$$

Indeed, it is easy to check that each term of the sum is a telescopic product, equal to $\frac{1}{n}$. As a result:

$$\mathbb{E}[S_1(Q_i)|Z_i = \ell] = \sum_{x \in Q_i} x\mathbb{P}(\mathcal{P}_1 \text{ samples } x|Z_i = \ell) = \frac{\ell S}{n}$$

On the other hand, we have:

$$\mathbb{E}[S_1(Q_i)] = \sum_{\ell=0}^{\min\{k,n\}} \mathbb{E}[S_1(Q_i)|Z_i = \ell]\mathbb{P}(Z_i = \ell) = \frac{S}{n}\sum_{\ell=0}^{\min\{k,n\}} \ell\mathbb{P}(Z_i = \ell)$$
$$= \frac{S}{n}\mathbb{E}[Z_i]$$

Since all clusters have the same size n, each of them will be sampled the same number of times in expectation, i.e., $\mathbb{E}[Z_i] = \frac{k}{m} = c$.[11] This proves the claim. □

Remark. Note that Claim 2 implies that we must have:

$$(1 - \mathbb{P}(Z_i = c))\mathbb{E}[S_1(Q_i)|Z_i = c] = \sum_{\ell \neq c} \mathbb{E}[S_1(Q_i)|Z_i = \ell]\mathbb{P}(Z_i = \ell)$$

Claim 3. *Given a cluster system as above, we have $\mathbb{E}[S_2(Q_i)] \leq \mathbb{E}[S_1(Q_i)|Z_i = c]$, for every $i = 1, \dots, m$.*

Proof. The distributions of $S_2(Q_i)$ and $(S_1(Q_i)|Z_i = c)$ are exactly the same as those of the variables S_2 and S_1 considered in Claim 1. In particular, this is true whenever we restrict to the choices made by $\mathcal{P}_1$ and $\mathcal{P}_2$ regarding the sampling of items from Q_i. As a consequence, the claim follows immediately from Claim 1. □

[11] We may want to try and make this completely formal for clusters of possibly different sizes.

Having shown that $\mathbb{E}[S_1(Q_i)|Z_i = c] = \mathbb{E}[S_1(Q_i)]$ for every $i = 1, \ldots, m$ (Claim 2) and that $\mathbb{E}[S_2(Q_i)] \leq \mathbb{E}[S_1(Q_i)|Z_i = c]$ for every $i = 1, \ldots, m$ (Claim 3), we immediately have the following

Claim 4. *Assume $X = \bigcup_{i=1}^{m} Q_i$, where $Q_i \cap Q_j = \emptyset$ for $i \neq j$ and $|Q_i| = n$ for every $i = 1, \ldots, m$. Consider the random variables S_1 and S_2 defined by the processes $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively. We have*

$$\mathbb{E}[S_2] \leq \mathbb{E}[S_1]$$

Chapter 4

Experimental Results

4.1 Empirical Framework

In every implementation of the final model we can observe m different copies of the Multilayer Perceptron (MLP) performing their training for $j = 2$ epochs and $n = 150$ rounds. The chosen optimization technique for each MLP is a Stochastic Gradient Descent (SGD) with a learning rate of 0.005 and a mini-batch size of 64. All the K clients are orchestrated by a central server developed with FedLab.

A pool of K clients was generated by federating the Fashion MNIST dataset into K local datasets using the FedArtML library. Each local dataset was sampled from the centralized dataset according to a Dirichlet distribution with a chosen level of alpha, based on the original distribution of response labels $P(y)$.[1] The first set of experiments was performed on a federated version of Fashion MNIST with $K = 100$ and an alpha value of 0.05, resulting in an average Hellinger distance of 90% among the data of the 100 clients. Another set of experiments employed a federated dataset generated with alpha equal 0.05 but with a reduced number of clients, $K = 30$. Subsequent experiments involved a new federation, again with $K = 100$ but this time with an alpha value of 0.2, leading to an average Hellinger distance of 80% between the data of the 100 clients. These alterations allow for a more comprehensive analysis of the data under different federated scenarios.

All the methodologies outlined in chapters 1, 2 and 3 were developed on JupyterLab notebooks operating under Python v3.10.12, and were ultimately evaluated by averaging the results coming from 10 distinct simulations, each initialized with a unique random seed[2] to ensure the robustness of the findings. Nevertheless, interpreting the final model's Accuracy and Loss remained challenging due to significant

[1] Note that this process of federating data using the response labels distribution is distinct from the clustering process conducted in section 4.2.

[2] This introduces variability to the simulation outcomes, thereby enhancing the reliability of the results. Specifically, the set of seeds used is: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9].

variance across rounds caused by the high level of non IID-ness in our federated data, produced by the high levels of Hellinger distance we imposed during federation (80% and 90%). To improve readability, all the Accuracy plots show the maximum accuracy value recorded up to each round x. Similarly, the Loss plots present the minimum Loss value recorded up until each round x. This approach smooths out fluctuations, providing a clearer view of the overall trend in model performance throughout the training process.

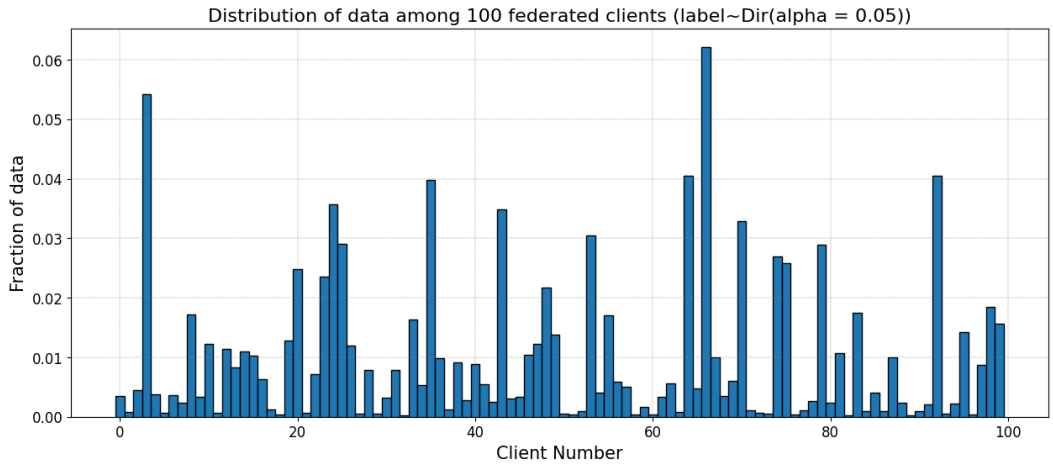


Figure 4.1. Distribution of $K = 100$ Federated Datasets from Fashion MNIST - HD: 0.9

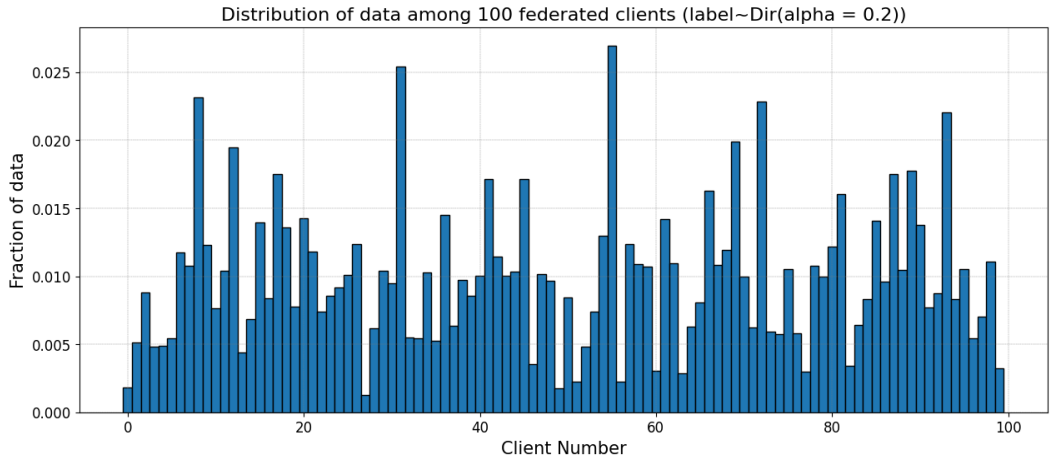


Figure 4.2. Distribution of $K = 100$ Federated Datasets from Fashion MNIST - HD: 0.8

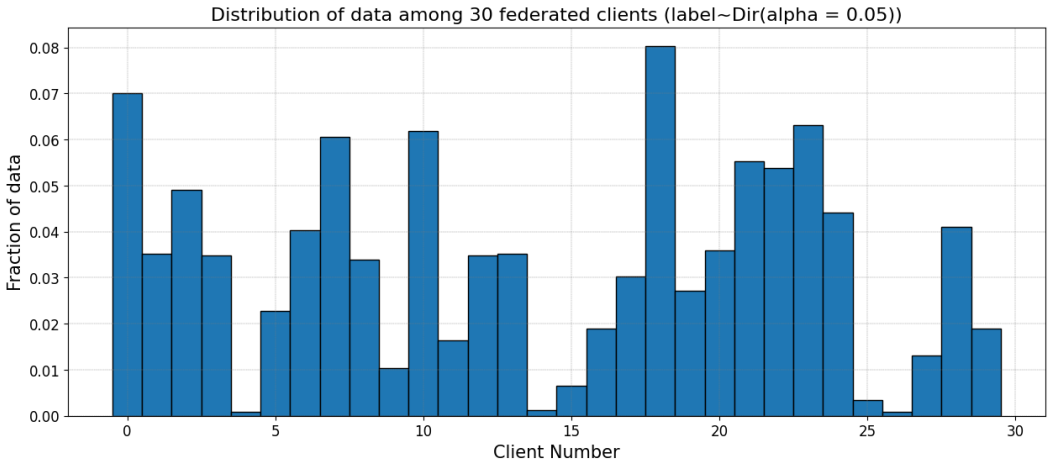


Figure 4.3. Distribution of $K = 30$ Federated Datasets from Fashion MNIST - HD: 0.9

4.2 Clustering Results

Table 4.1 presents the results of applying the grid search functions to each of the clustering algorithms examined in this thesis. The objective of these functions was to explore a given portion of the parameter space, and find the parameter values that could maximize the silhouette score, while at the same time satisfying the constraint of assigning at least 70% of clients to any cluster.

	K-Medoids			**DBSCAN**			**OPTICS**		
	K: 100 HD:0.9	K: 30 HD:0.9	K: 100 HD:0.8	K: 100 HD:0.9	K: 30 HD:0.9	K: 100 HD:0.8	K: 100 HD:0.9	K: 30 HD:0.9	K: 100 HD:0.8
P(y)	0.452	0.438	0.231	0.673	0.584	0.306	0.663	0.584	0.334
P(X|y)	0.294	NA	0.139	0.208	NA	NA	0.267	NA	NA

Table 4.1. Silhouette scores from K-Medoids, DBSCAN and OPTICS

Remember that a silhouette score close to +1 indicates that the sample is well assigned to its cluster, and is relatively far from its neighbors. A value close to 0 suggests that the sample lies on the boundary between two clusters and could potentially belong to either one. Negative values, instead, mean that the sample might have been misclassified and could belong to a different cluster. Analyzing table 4.1 reveals that the clustering results obtained using $P(X|y)$ significantly underperform compared to those achieved with the distribution of response labels $P(y)$. This does not come as a surprise, if we think about the role $P(y)$ played when federating the original Fashion MNIST dataset. When working with $P(X|y)$,

sometimes it was not even possible to cluster the minimum amount of clients required by the constraint (70%). Those cases have been classified as NA. For all these reasons, we decided to adopt $P(y)$ as our preferred data distribution for clustering. Additionally, we can observe how, by reducing the Hellinger distance between clients, we hinder the clustering algorithms' ability to effectively distribute the pool of clients into distinct clusters, as evidenced by the lower silhouette score observed for $K = 100$ and $HD = 0.8$.

Figure 4.4. Clustering Outcome - K = 100, HD: 0.9 - K-Medoids

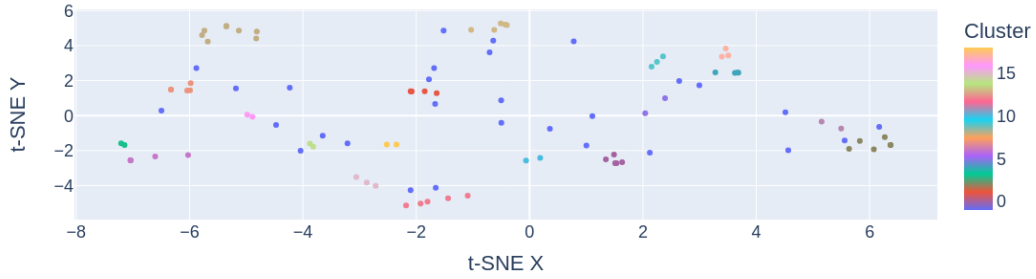


Figure 4.5. Clustering Outcome - K = 100, HD: 0.9 - DBSCAN

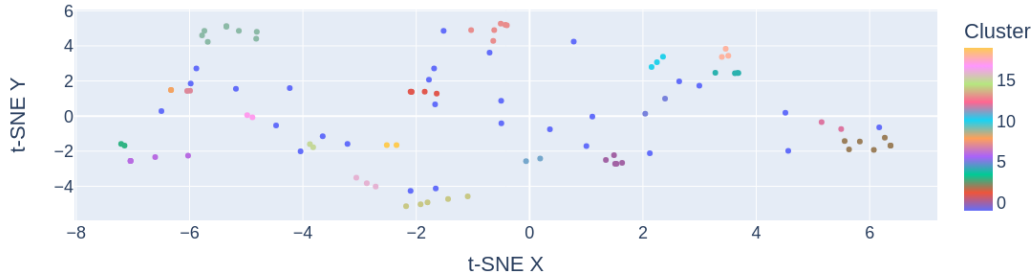


Figure 4.6. Clustering Outcome - K = 100, HD: 0.9 - OPTICS

Figures 4.4, 4.5 and 4.6 visually represent the outcomes returned by each clustering technique. To create these plots, we first reduced the dimensionality of the pairwise Hellinger distance matrices to 2D using T-SNE. The approximately 30% of clients which could not be clustered will be, from now on, considered as unique members of their own clusters. This approach allows their inclusion in the training process, ensuring we don't miss the opportunity to train our final model with the most "diverse" data in the dataset. In figures 4.7, 4.8 and 4.9, we plot the number of clients in each cluster, with each cluster identified on the x-axis by its creation number. Notably, several clusters are represented by a single device. These are the clients that were previously unlabelled.

Figure 4.7. Distribution of Clients Among Clusters - K = 100, HD: 0.9 - K-Medoids

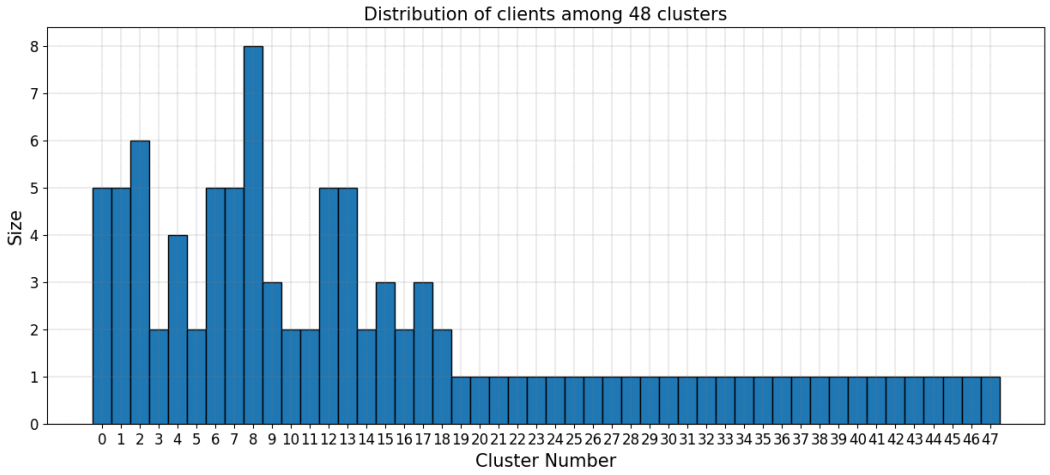


Figure 4.8. Distribution of Clients Among Clusters - K = 100, HD: 0.9 - DBSCAN

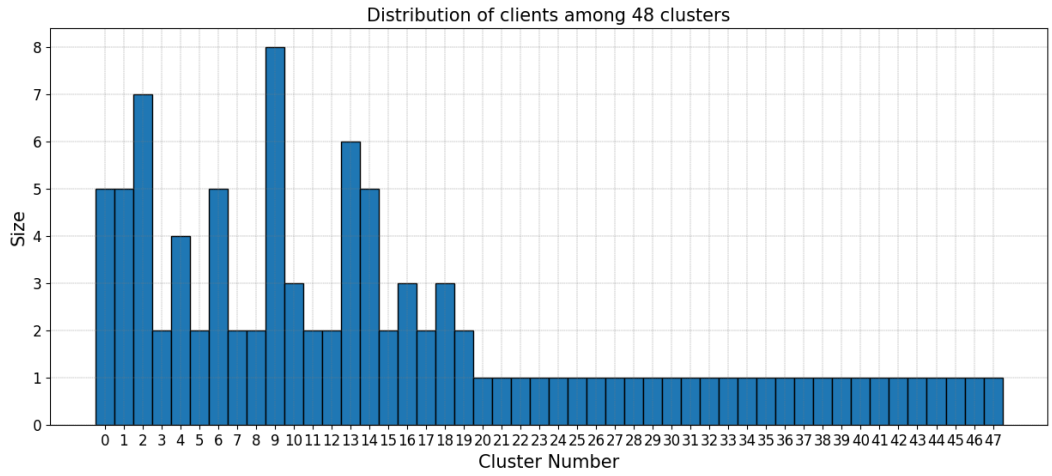


Figure 4.9. Distribution of Clients Among Clusters - K = 100, HD: 0.9 - OPTICS

K-Medoids was at this point abandoned due to its poor performance. Further research has been conducted, instead, on the two density-based clustering algorithms, DBSCAN and OPTICS. One after the other, they were tested in combination with each of the adaptive selection strategies introduced in subsection 3.3.2, to understand which type of clustering method could lead to better results.[3] The chosen scenario was $K = 100$ and $HD = 0.9$. In figures 4.10 and 4.11 it is possible to compare the values of Accuracy and Loss scored by each implementation when choosing $d = 10$ and $w = 5$. These results are comparable, yet they show a slight improvement when using OPTICS. Going forward, we will use OPTICS as single common clustering algorithm in our final model, which will serve as a foundation for all the different implementations of adaptive selection strategies.

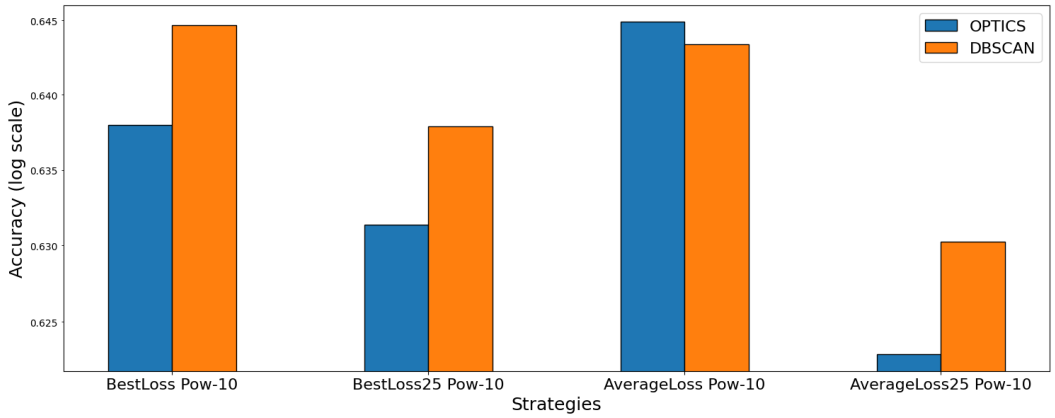


Figure 4.10. OPTICS vs DBSCAN - Accuracy (log scale) - K : 100, HD : 0.9, d : 10, w : 5

[3]Eventually, we had to rely on empirical experiments to understand which clustering method to choose.

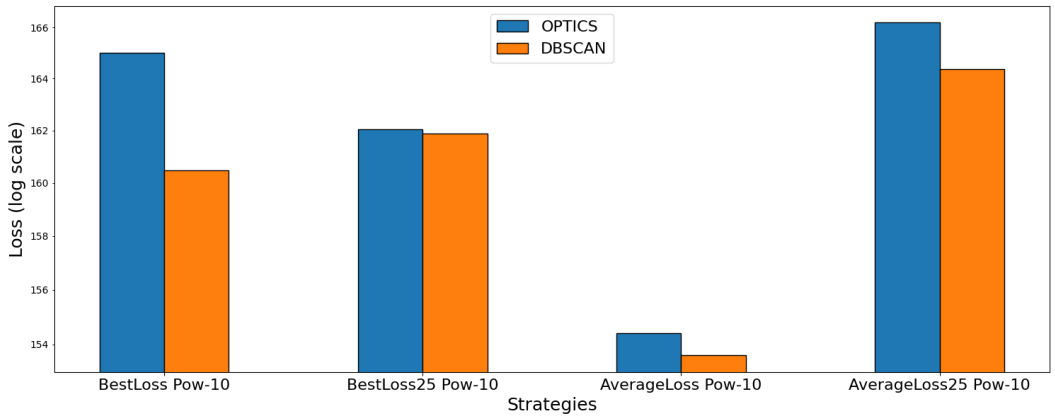


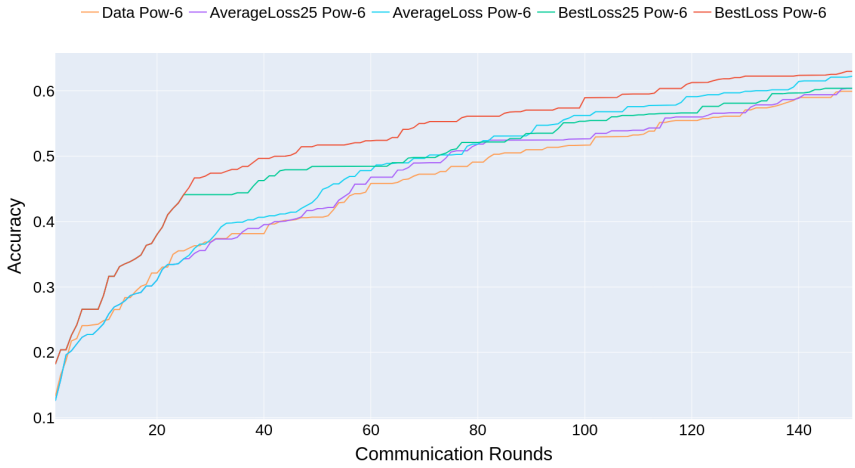
Figure 4.11. OPTICS vs DBSCAN - Loss (log scale) - $K : 100$, $HD : 0.9$, $d : 10$, $w : 5$

4.3 Loss and Accuracy Plots

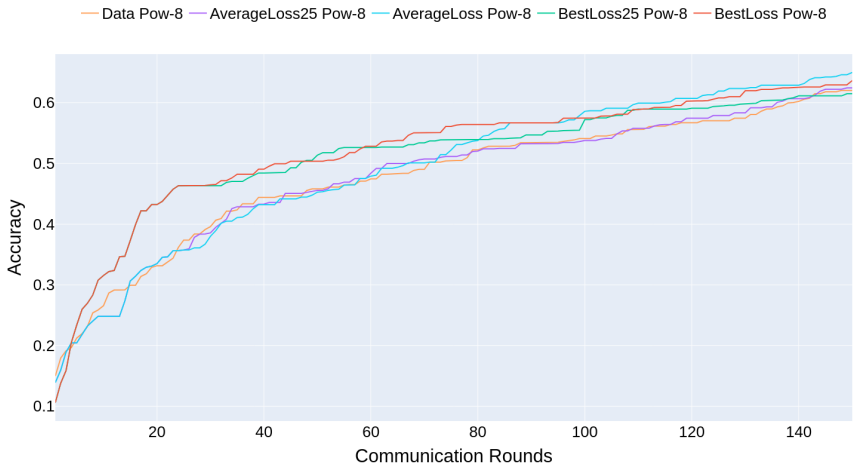
In figures 4.12 and 4.13 we can compare the various adaptive selection strategies discussed in this thesis. We immediately notice how the Clustered Best-Loss Pow-d variant consistently delivered the best results in Accuracy and Loss, outperforming the AverageLoss strategy, which, despite its close performance, never quite matched BestLoss. The Clustered Data Pow-d strategy, instead, yielded unsatisfactory results, suggesting that client selections that favor clients with higher local losses are indeed more effective. The BestLoss25 and AverageLoss25 strategies, while not exactly reaching the levels of their original counterparts, showed nonetheless promising results, offering close metrics to those returned by BestLoss and AverageLoss, with the added benefit of reducing communication overhead. They can be therefore considered as viable alternatives.

We now want to compare the performance of a Clustered BestLoss Pow-d implementation with respect to a standard Power-Of-Choice. Since a Clustered BestLoss Pow-d involves calculating local losses for each client in K at each round, we set out to verify that choosing BestLoss as our preferred strategy leaves no room for the Power-Of-Choice to outperform it, regardless of the chosen d value. Therefore, we conducted several experiments aimed to understand whether a BestLoss with a fixed value of d could perform better than any Power-Of-Choice implementation, irrespective of the choice of d.[4] Of course, the amount of clients w to be trained per round has to be equal. In figures 4.14 and 4.15 we see how the Clustered BestLoss Pow-d indeed performs better than a standard Power-Of-Choice irrespective of the chosen value for d, with a given number of clients to be trained per round w.

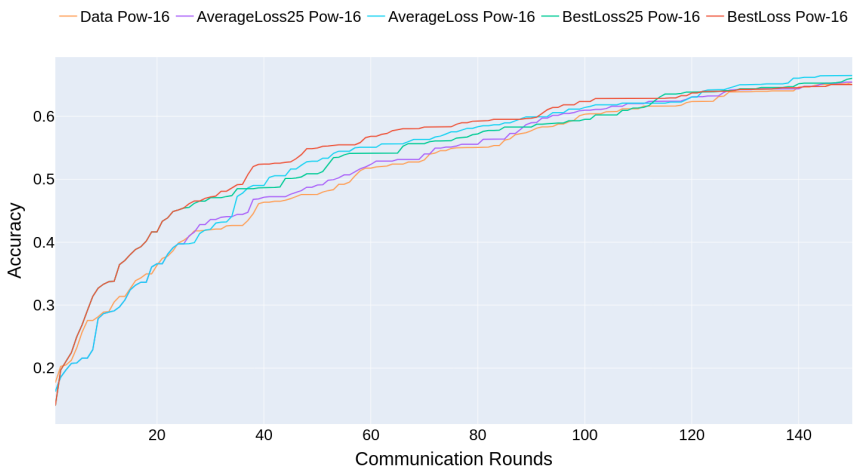
[4]Remember that in the BestLoss d is still operating behind the scenes. The difference between a standard Pow-d, and a Clustered BestLoss Pow-d, is that in the latter d is not responsible anymore for deciding the amount of clients computing their local loss at the beginning of each round.



(a) $d = 6$, $w = 3$

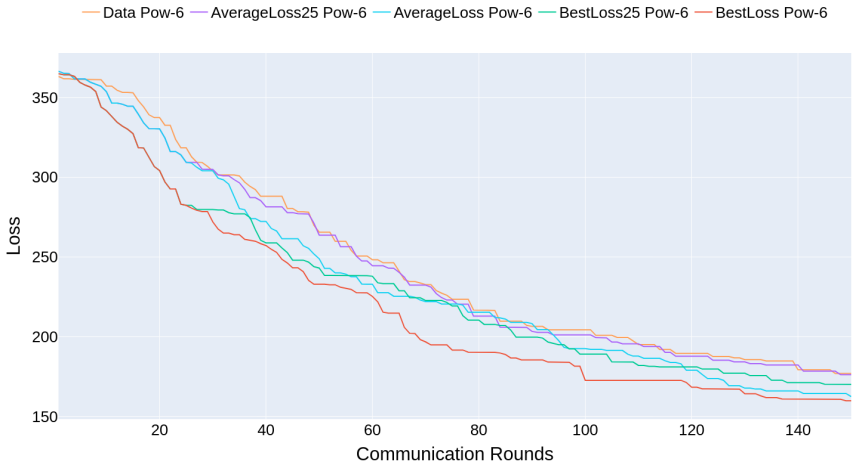


(b) $d = 8$, $w = 4$

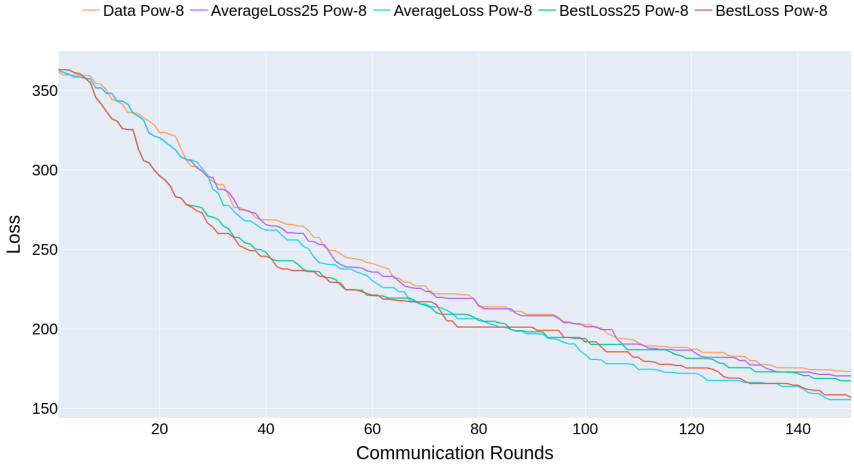


(c) $d = 16$, $w = 8$

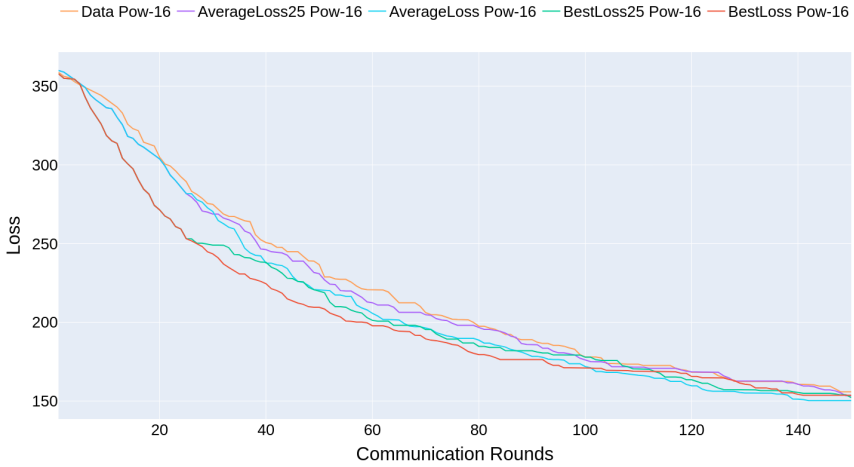
Figure 4.12. Comparison between adaptive selection strategies - Accuracy



(a) $d = 6$, $w = 3$

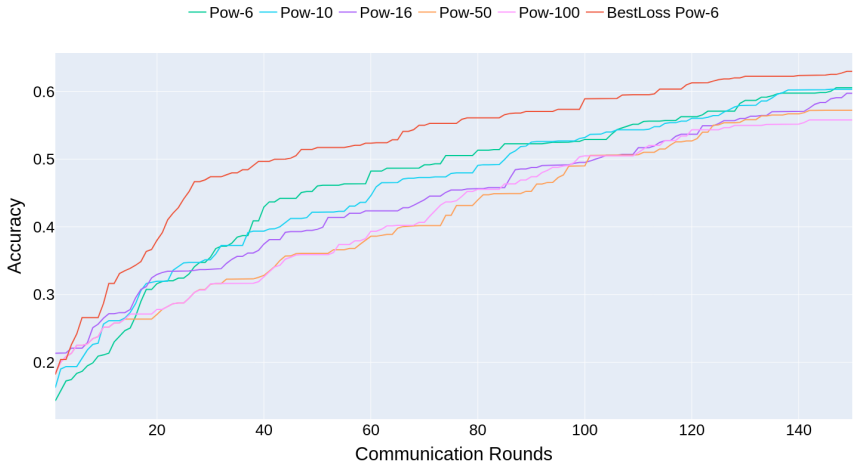


(b) $d = 8$, $w = 4$

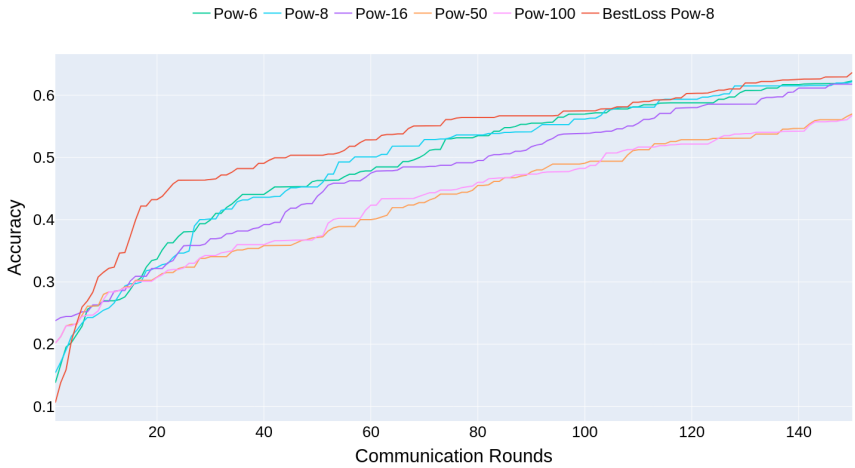


(c) $d = 16$, $w = 8$

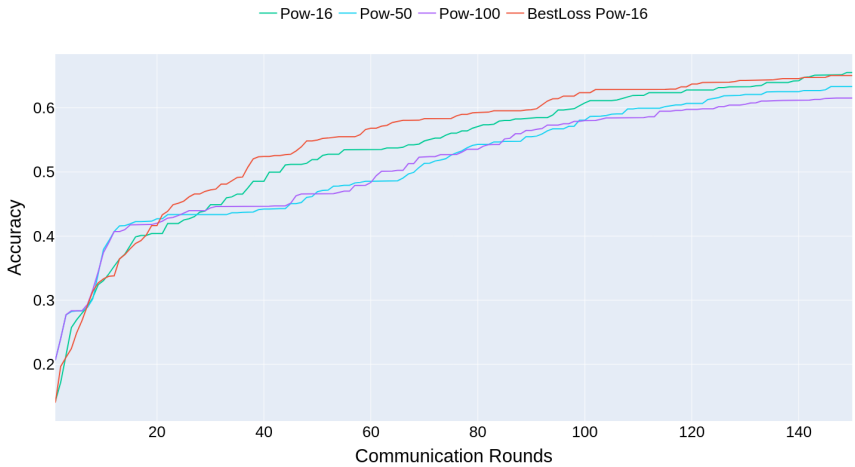
Figure 4.13. Comparison between adaptive selection strategies - Loss



(a) Clustered BestLoss Pow-6 vs Pow-d (different d) - $w = 3$ - Accuracy

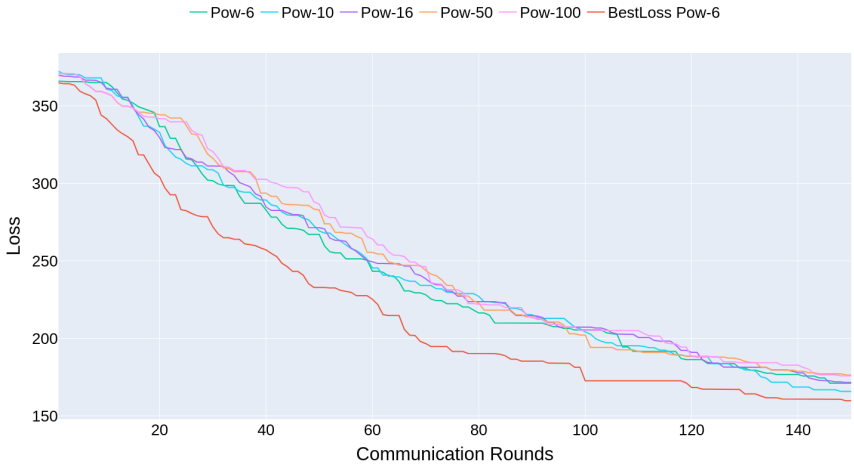


(b) Clustered BestLoss Pow-8 vs Pow-d (different d) - $w = 4$ - Accuracy

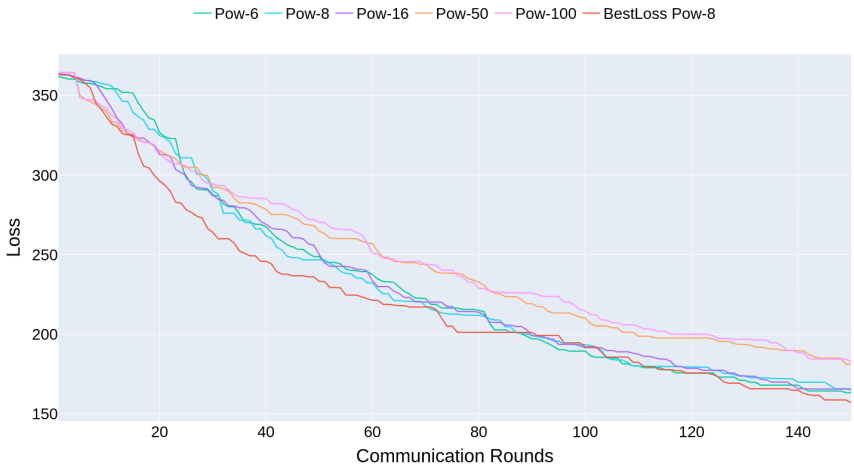


(c) Clustered BestLoss Pow-16 vs Pow-d (different d) - $w = 8$ - Accuracy

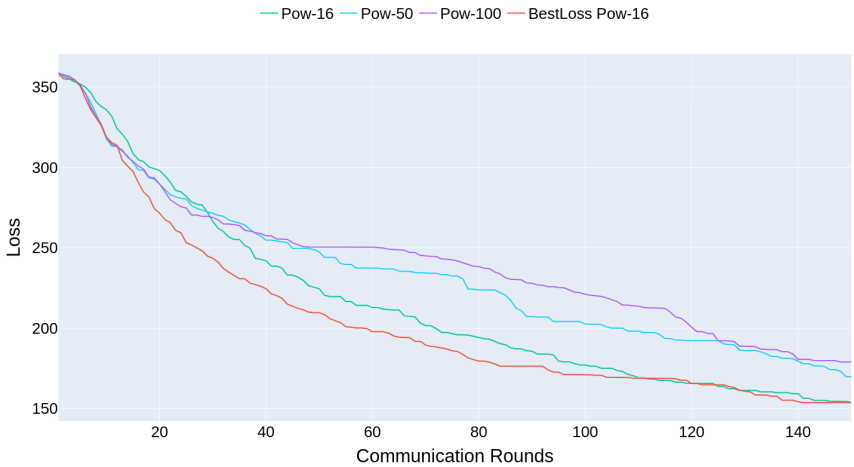
Figure 4.14. Clustered BestLoss Pow-d vs Pow-d - Accuracy



(a) Clustered BestLoss Pow-6 vs Pow-d (different d) - $w = 3$ - Loss

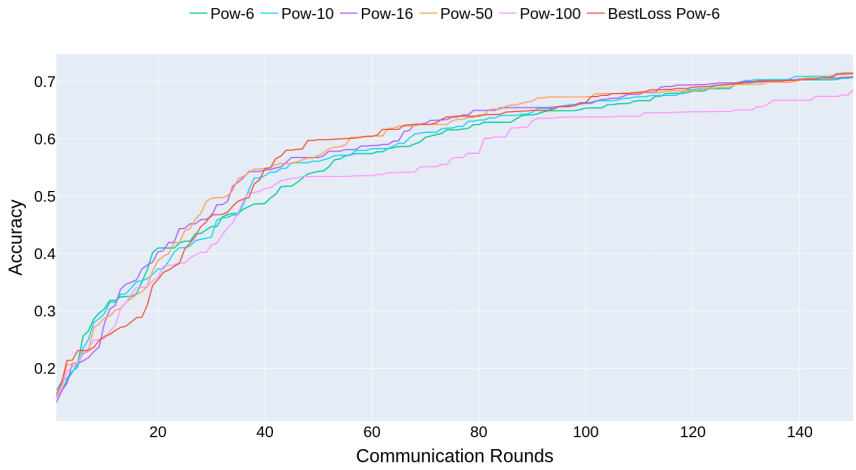


(b) Clustered BestLoss Pow-8 vs Pow-d (different d) - $w = 4$ - Loss

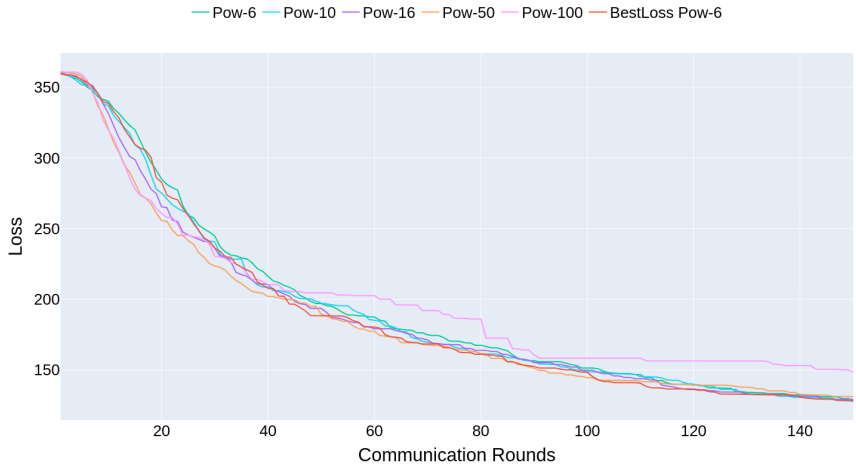


(c) Clustered BestLoss Pow-16 vs Pow-d (different d) - $w = 8$ - Loss

Figure 4.15. Clustered BestLoss Pow-d vs Pow-d - Loss



(a) $d = 6$, $w = 3$ - Accuracy

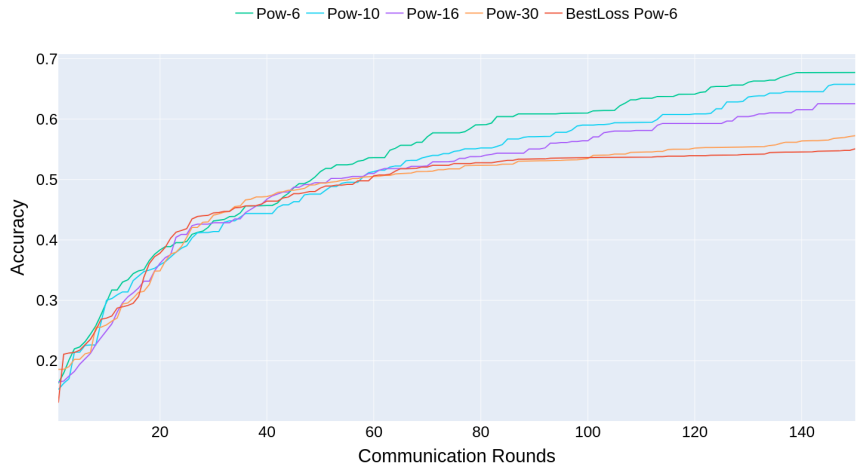


(b) $d = 6$, $w = 3$ - Loss

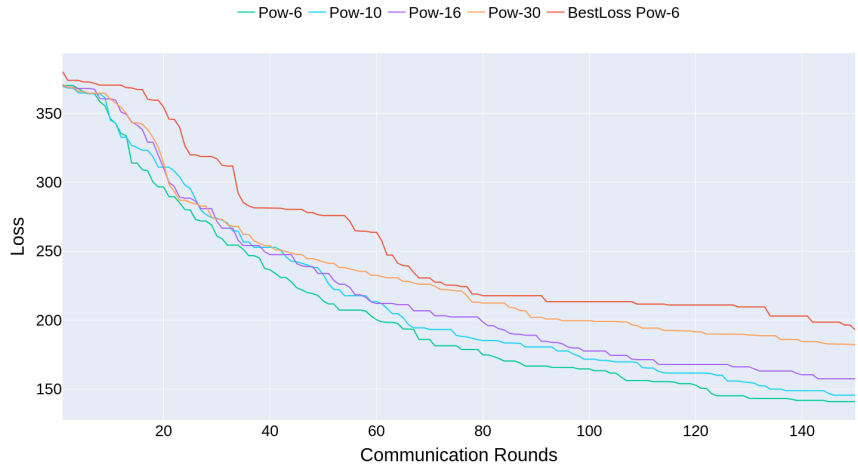
Figure 4.16. Comparison at $K = 100$, $HD = 0.8$

Results were notably different in the federated scenario with a Hellinger distance of 80% and 100 clients. In this setting, the efficiency of the clustering algorithm begins to decline, making it less effective in targeting the selection towards clients with diverse data. As a result, the performance achieved by the BestLoss is not as remarkable as before, as evident in figure 4.16. In any case, the BestLoss strategy can still match the results achieved by the most appropriately fine-tuned versions of the Power-Of-Choice.

In the last scenario, with only 30 clients ($K = 30$), the BestLoss strategy unsurprisingly yielded poor results, as shown in figure 4.17. This is because focusing on the highest local losses is a strategy that thrives primarily when the available data is highly distributed, ideally with values of K far exceeding 100, and with Hellinger distance levels above 85%. In fact, the advantage of implementing the proposed



(a) $d = 6$, $w = 3$ - Accuracy



(b) $d = 6$, $w = 3$ - Loss

Figure 4.17. Comparison at $K = 30$, $HD = 0.9$

adaptive selection strategies lies almost entirely in their efficiency in selecting the most unique (clustering) and diverse (Power-Of-Choice) data. When the data is concentrated among only a few clients, as in the case of $K = 30$, the BestLoss strategy struggles to fully exploit its potential. This is evident in figure 4.18, where we observe that, in 150 rounds, only 15 out of 30 clients have been selected for training. Furthermore, some of these 15 clients have been selected only two or three times. This can be a very useful feature when data is widely spread, but it becomes quite disadvantageous when, on the other hand, data is heavily concentrated within a few devices.

Note that when we talk about achieving "better" performance in terms of Accuracy and Results, we are referring to gaining both higher accuracy and lower

loss (with respect to other strategies) in fewer training rounds, while at the same time investing less computational power on training. Obviously, this statement holds true only if we assume, as we did in chapter 1, that the computational power to compute single local losses at each round is negligible.

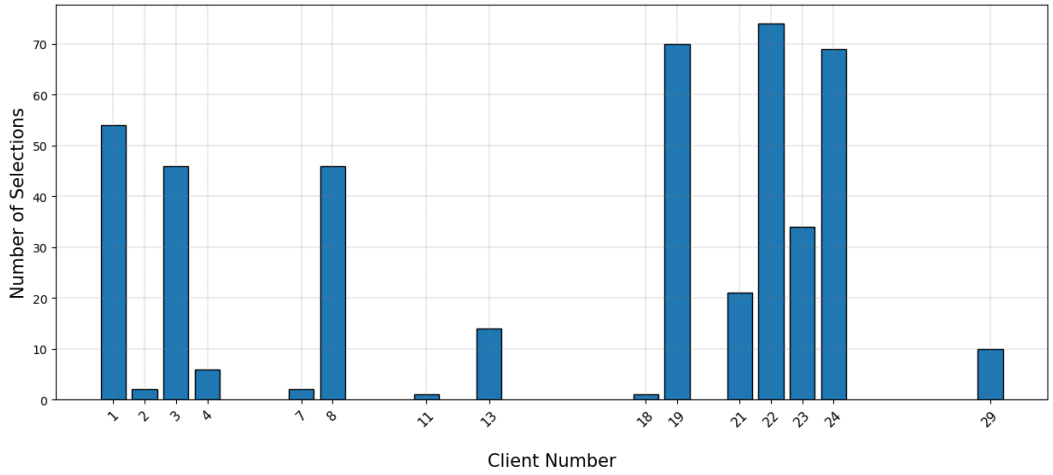
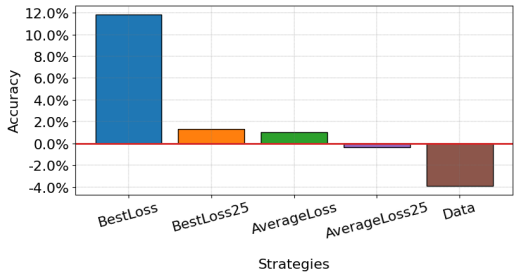


Figure 4.18. Client selection frequency - $K = 30$, $HD = 0.9$ - $d = 6$, $w = 3$ - Loss

4.4 Loss and Accuracy Quantitative Results

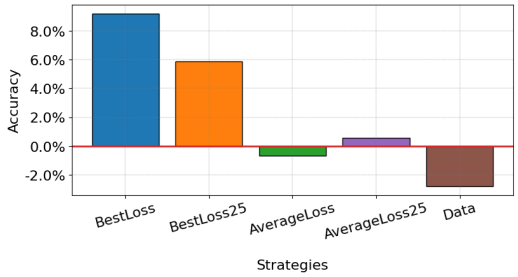
To provide a clearer understanding of how much more efficient the adaptive selection strategies can be when compared to a traditional Power-Of-Choice, we present the plots in figures 4.19 and 4.20. These plots allow us to simultaneously compare the accuracy and loss values of all the adaptive strategies against a Power-Of-Choice implementation with $w = m$, initialized with the optimal value of d for that specific m.[5] This Power-Of-Choice implementation will serve as a baseline for our quantitative studies. Therefore, the accuracy and losses returned by all the adaptive strategies will be normalized by the values returned by their corresponding Power-Of-Choice. The baseline can be seen in the plots as the thick red line on the y axis at $y = 0$. Depending on whether we are consulting the accuracy plots or the loss plots, we can understand how much better (or worse) any experimental strategy has performed by observing whether the values are above or below the baseline. In each figure, there are two columns of plots. The right column displays the performance values measured at the time of convergence ($n = 150$), while the left column presents the performance values captured halfway through the training process ($n = 70$). Comparing these two sets of values side-by-side provides a clearer picture of the improvement we achieved in terms of efficiency by implementing the adaptive selection strategies.

[5]We employed Accuracy as the primary metric to identify the optimal d across all plot types.

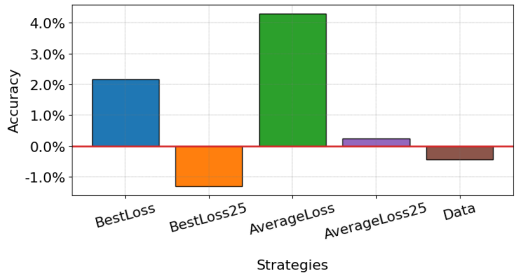


(a) $d = 6$, $w = 3$ vs , $n = 70$

(b) $d = 6$, $w = 3$, $n = 150$

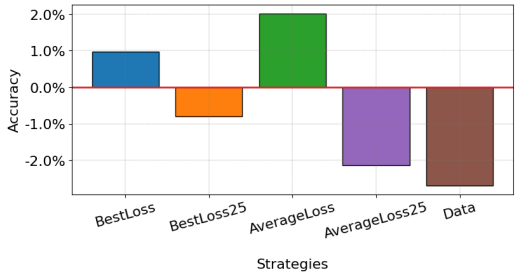


(c) $d = 8$, $w = 4$, $n = 70$



(d) $d = 8$, $w = 4$, $n = 150$

(e) $d = 10$, $w = 5$, $n = 70$

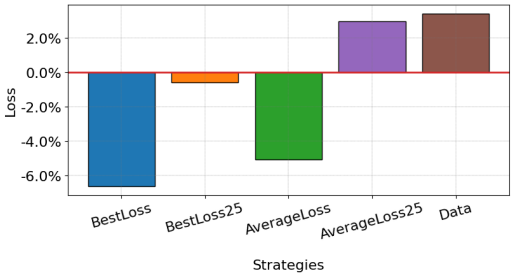


(f) $d = 10$, $w = 5$, $n = 150$

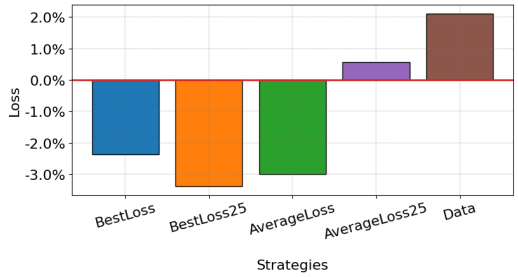
Figure 4.19. Accuracy - $HD = 0.9$, $K = 100$ - $n = 70$ vs $n = 150$

As we can see, the Clustered BestLoss Pow-d strategy consistently outperforms the Power-Of-Choice in the given federated scenario. The worst case occurs for $d = 16$ and $w = 8$, where the Clustered BestLoss Pow-d achieves a performance of approximately $+1\%$ in terms of Loss with respect to the Power-Of-Choice. This result suggests that the worst case for the BestLoss is still comparable to the best case for the Power-Of-Choice.

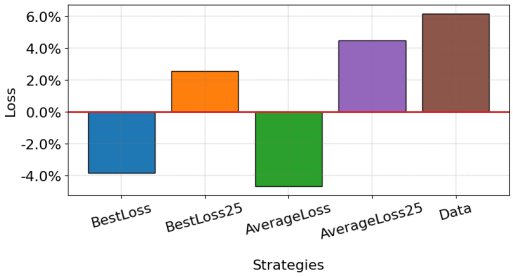
(a) $d = 6$, $w = 3$, $n = 70$



(b) $d = 6$, $w = 3$, $n = 150$

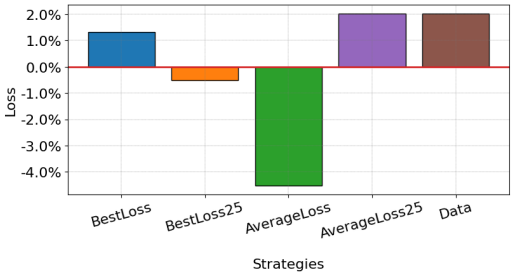


(c) $d = 8$, $w = 4$, $n = 70$



(d) $d = 8$, $w = 4$, $n = 150$

(e) $d = 10$, $w = 5$, $n = 70$



(f) $d = 10$, $w = 5$, $n = 150$

Figure 4.20. Loss - $HD = 0.9$, $K = 100$ - $n = 70$ vs $n = 150$

4.5 Shared Messages and Communication Overhead

The advantages introduced by the Clustered BestLoss Pow-d and the Clustered AverageLoss Pow-d strategies come with a price. The number of messages to be shared between the central server and the available clients is indeed considerable, as shown in figure 4.21. When the number of messages increases, so does the communication overhead. In table 4.2 we include the amount of communication

sizes recorded while training each strategy. The provided examples come from the federated scenario $K = 100$ and $HD = 0.9$, and were returned by different Power-Of-Choice based selection strategies with parameters $d = 6$ and $m = 3$.

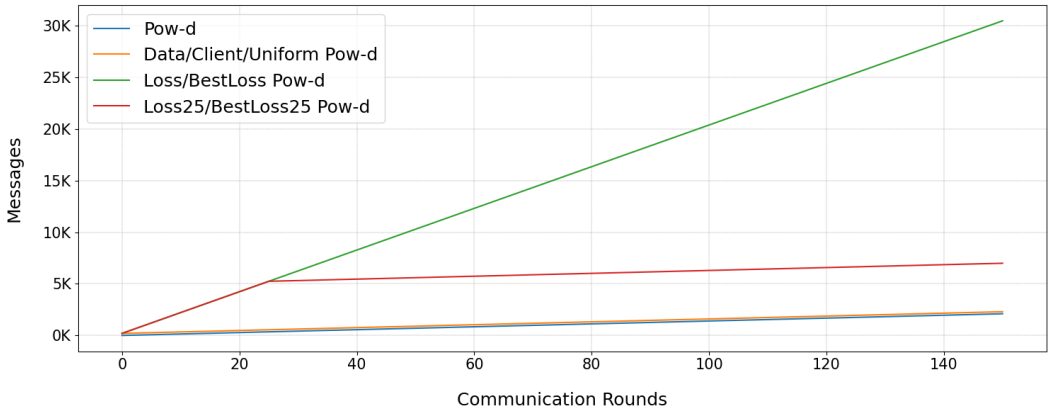


Figure 4.21. Amount of Shared Messages - $K = 100$, $HD = 0.9$ - $d = 6$, $w = 3$

Overall, the size of data to be transferred between the central server and its clients is reduced (at most a few megabytes). However, exchanging such a large number of messages can be challenging.

Source	**Communication Size (KB)**
Pow-6	135.94
Pow-8	161.72
Pow-10	196.88
Pow-16	274.22
Pow-50	754.69
Pow-100	1464.84
BestLoss Pow-6	2016.80
BestLoss25 Pow-6	477.73
AverageLoss Pow-6	2016.80
AverageLoss25 Pow-6	477.73
Data Pow-6	169.92

Table 4.2. Communication Sizes - $K = 100$, $HD = 0.9$ - $d = 6$, $w = 3$

4.6 Time To Accuracy

The *Time to Accuracy*, in the context of machine learning, is a metric that refers to the amount of time it takes for a model to reach a predetermined level of accuracy during its training process. This concept is particularly significant in scenarios where the speed of training is as crucial as the performance of the model itself. In figure 4.22 we present a plot showing the Time to Accuracy required by each strategy to reach different levels of accuracy. As we can see, the time differences in reaching the desired accuracy levels among different strategies are negligible (up to two minutes). These results, however, are purely indicative, as our research is entirely based on simulations conducted on a single machine, using FedLab to *emulate* the communications between the central server and its K clients. While asynchronous in nature, these simulated communications do not fully capture the heterogeneity of real world IoT devices, including their varying computational capabilities, potential latency fluctuations, and network reliability issues. Despite these limitations, we still believe that the provided information can represent a reasonable approximation of what could be observed in a realistic federated learning environment.

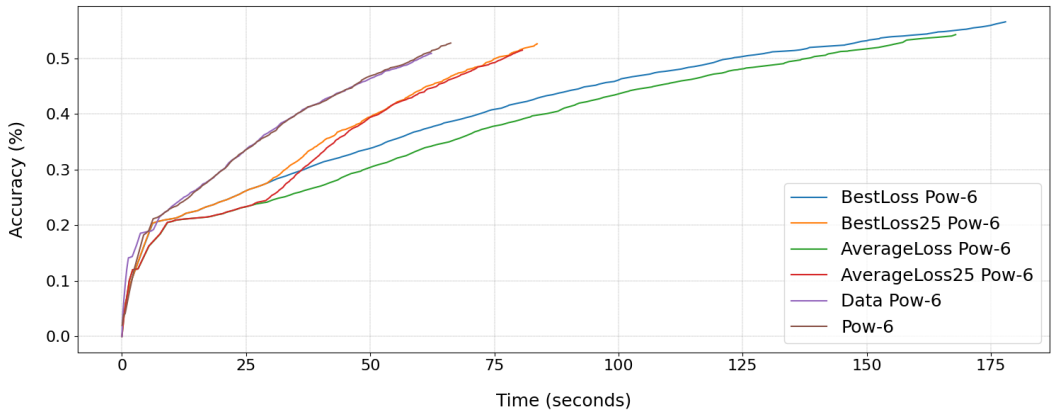


Figure 4.22. Time to Accuracy - $K = 100$, $HD = 0.9$ - $d = 6$, $w = 3$

4.7 F1 Scores

The F1 score is a statistical tool in machine learning that measures the accuracy of a classification model, especially when both precision and recall are essential. Precision assesses how well the model identifies true positives, while recall evaluates if the model captures all positive instances. The F1 score blends these two metrics into a single value, representing a balance between precision and recall. This makes it particularly valuable when dealing with uneven class distributions or when both types of classification errors (false positives and false negatives) are equally important.

Table 4.3. F1 Scores - $K = 100$, $HD = 0.9$, $d = 6$, $w = 3$

Class	Standard Pow-6	AverageLoss Pow-6	BestLoss Pow-6
T-shirt/top	0.00	0.75	0.07
Trouser	0.89	0.93	0.91
Pullover	0.00	0.00	0.00
Dress	0.58	0.73	0.54
Coat	0.00	0.44	0.42
Sandal	0.56	0.59	0.57
Shirt	0.31	0.01	0.35
Sneaker	0.60	0.57	0.67
Bag	0.00	0.90	0.51
Ankle boot	0.74	0.60	0.31

Table 4.4. F1 Scores - $K = 30$, $HD = 0.9$, $d = 6$, $w = 3$

Class	Standard Pow-6	AverageLoss Pow-6	BestLoss Pow-6
T-shirt/top	0.76	0.76	0.03
Trouser	0.96	0.94	0.88
Pullover	0.00	0.53	0.57
Dress	0.71	0.80	0.39
Coat	0.47	0.00	0.00
Sandal	0.58	0.57	0.48
Shirt	0.01	0.03	0.32
Sneaker	0.00	0.00	0.00
Bag	0.02	0.60	0.08
Ankle boot	0.57	0.00	0.00

Table 4.5. F1 Scores - $K = 100$, $HD = 0.8$, $d = 6$, $w = 3$

Class	Standard Pow-6	AverageLoss Pow-6	BestLoss Pow-6
T-shirt/top	0.74	0.56	0.62
Trouser	0.89	0.92	0.87
Pullover	0.50	0.55	0.60
Dress	0.48	0.56	0.56
Coat	0.47	0.62	0.29
Sandal	0.57	0.60	0.57
Shirt	0.00	0.00	0.39
Sneaker	0.73	0.73	0.00
Bag	0.74	0.89	0.76
Ankle boot	0.18	0.83	0.79

In tables 4.3, 4.4 and 4.5 we present the F1 scores recorded at the end of the training phase for each different federated configuration of the Fashion MNIST

dataset. Our objective is to understand whether (and to what extent) each of the ten classes within the Fashion-MNIST dataset has been adequately represented throughout the training process.

The Clustered AverageLoss Pow-d proved to be the most effective model in the first scenario, the one with $HD = 0.9$ and $K = 100$. In fact, it showed consistent performance across a wider range of classes, indicating a good ability in handling the distributed nature of the data. However, when the number of clients was reduced to 30, the standard Power-Of-Choice was the only strategy who showed some resistance to the overall decrease in performance. AverageLoss suffered in a few classes, namely Sneaker, Ankle Boot and Shirt. BestLoss, instead, was penalized almost everywhere. This suggests that the Power-Of-Choice might be better suited to scenarios with fewer data sources, as already mentioned at the end of section 4.3. In the final scenario, with $K = 100$ and $HD = 0.8$, it was the BestLoss to present an improvement in performance. It managed to overcome challenges that had previously been difficult for all models, such as classifying Pullovers, hence demonstrating its ability to generalize across a more distributed dataset. This suggests that it could be more robust to variations in data distribution, an important quality in federated learning environments.

Chapter 5

Conclusions and Future Work

This thesis demonstrates that integrating the Power-Of-Choice client selection strategy with client clustering can significantly outperform the standard Power-Of-Choice approach in federated learning, particularly within highly non-IID data environments. This novel implementation exploits its full potential when data is broadly distributed, possibly over more than 100 devices, and with an average Hellinger distance among each local dataset above 85%. In particular, the Clustered BestLoss Pow-d strategy introduced in this thesis is capable of improving the accuracy of our federated learning model by more than 10% with respect to an equivalent implementation of Pow-d after 70 rounds of training (more or less, half the number of rounds needed to reach convergence). In the same experiment, the proposed strategy also recorded an improvement of almost 14% in reducing the model's loss function. The efficiency of the proposed strategy is dependent on three key factors:

1. Precise fine tuning of the parameters d and w (equivalent, respectively, to the d and m key parameters of the Power-Of-Choice). Note that fine tuning the parameters is also required to efficiently implement a standard Pow-d.

2. A large number of available devices in our federated environment ($K \geq 100$).

3. High non-uniformity in data distribution, ideally with an average Hellinger distance among each local dataset above 85%.

Future research could investigate the Clustered BestLoss-X Pow-d variant, which retains the advantages of its original counterpart while reducing its communication overhead. In particular, we encourage further research to enhance its performance in terms of accuracy and loss. Although the findings presented in this thesis are robust, as they are based on the average of 10 independent simulations, we nevertheless recommend replicating each experiment on a different dataset, such as CIFAR-10, for further validation.

Additionally, the proposed implementation can be easily adapted to different client selection strategies beyond Power-Of-Choice, such as MOON (Model-Agnostic Optimization Over Networks), a strategy introduced by [Li et al., 2021], which we believe could significantly benefit from client clustering. In conclusion, we hope that the studies conducted in this thesis will prove useful for future research, and open new avenues for more efficient and scalable federated learning models in contexts of highly skewed federated data.

Bibliography

[Ankerst et al., 1999] Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. (1999). Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60.

[Chire, 2010] Chire (2010). Reachability plot obtained using the optics implementation in elki.

[Chire, 2011a] Chire (2011a). Cluster analysis with dbscan on a density-based data set.

[Chire, 2011b] Chire (2011b). Cluster analysis with dbscan on a gaussian-distributed-based data set.

[Chire, 2011c] Chire (2011c). Cluster analysis with k-means on a gaussian-distribution-based data set.

[Chire, 2011d] Chire (2011d). Cluster analysis with optics on a density-based data set.

[Cho et al., 2020] Cho, Y. J., Wang, J., and Joshi, G. (2020). Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. *ArXiv*, abs/2010.01243.

[Ester et al., 1996] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231.

[Jee Cho et al., 2022] Jee Cho, Y., Wang, J., and Joshi, G. (2022). Towards understanding biased client selection in federated learning. In Camps-Valls, G., Ruiz, F. J. R., and Valera, I., editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 10351–10375. PMLR.

[Kairouz et al., 2021] Kairouz, P., McMahan, H., Avent, B., Bellet, A., Bennis, M., Bhagoji, A., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R., Eichner, H., El Rouayheb, S., Evans, D., Gardner, J., Garrett, Z., Gascón,

A., Ghazi, B., Gibbons, P., and Zhao, S. (2021). *Advances and Open Problems in Federated Learning*. Now Publishers.

[Kaufman and Rousseeuw, 2009] Kaufman, L. and Rousseeuw, P. J. (2009). *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons.

[Li et al., 2021] Li, Q., He, B., and Song, D. (2021). Model-contrastive federated learning.

[Mark Van der Laan and Bryan, 2003] Mark Van der Laan, K. P. and Bryan, J. (2003). A new partitioning around medoids algorithm. *Journal of Statistical Computation and Simulation*, 73(8):575–584.

[Mitzenmacher, 2001] Mitzenmacher, M. (2001). The power of two choices in randomized load balancing. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1094–1104.

[Wilmer et al., 2009] Wilmer, E., Levin, D. A., and Peres, Y. (2009). Markov chains and mixing times. *American Mathematical Soc., Providence*.

[Wolfrath et al., 2022] Wolfrath, J., Sreekumar, N., Kumar, D., Wang, Y., and Chandra, A. (2022). Haccs: Heterogeneity-aware clustered client selection for accelerated federated learning. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 985–995.