

Data Partitioning Strategies for Simulating non-IID Data Distributions in the DDM-PS-Eval Evaluation Platform

Mikołaj Markiewicz^a and Jakub Koperwas^b

Institute of Computer Science, Warsaw University of Technology, Warsaw, Poland

Keywords: Distributed Data Mining, Federated Learning, non-IID Data, Data Partitioning Strategies, Algorithm Evaluation, Clustering, Classification.


Abstract: Nowadays, the size of the various datasets collected worldwide is growing rapidly. These data are stored in different data centres or directly on IoT devices, and are thus located in different places. Data stored in different locations may be uniformly distributed and convergent in terms of the information carried, and are then known as independent and identically distributed (IID) data. In the real world, data collected in different geographic regions tend to differ slightly or have completely different characteristics, and are then known as non-IID data. Increasing numbers of new algorithms have been implemented to work with such distributed data without the need to download all the data to one place. However, there is no standardised way of validating these, and such algorithms are typically tested on IID data, which are uniformly distributed. The issue of non-IID data is still an open problem for many algorithms, although the main categories of "non-IID-ness" have been defined. The purpose of this paper is to introduce new data partitioning strategies and to demonstrate the impact of non-IID data on the quality results of distributed processing. We propose multiple strategies for dividing a single dataset into multiple partitions to simulate each of the major non-IID data category problems faced by distributed algorithms. The proposed methods of data partitioning integrated with the DDM-PS-Eval platform will enable the validation of future algorithms on datasets with different data distributions. A brief evaluation of the proposed methods is presented using several distributed clustering and classification algorithms.


1 INTRODUCTION

Various methods of data mining have been developed, and in the last few years, there has been growing interest in the study of federated learning (FL) techniques. In this approach, a large amount of data is distributed over a large number of devices that cooperate in the learning process. It is worth noting that the concept of FL represents a special case of general distributed data mining (DDM) methods, which also deal with distributed data but at different scales. For example, the processing of megabytes of data stored in millions of devices requires a different approach from the processing of petabytes of data stored in a few data centres. A lack of knowledge about the characteristics of the data distribution and the inability to download all of the data locally are challenges for modern algorithms operating in distributed environments. This type of data, which are typically collected and stored in different geographic regions, does not guarantee an equal distribution of data samples for each location. The data are then not independent and identically dis-

tributed (non-IID).

Research into non-IID data partitioning has become very popular in the last few years, and researchers have begun to pay more attention to the impact of uneven data distributions on the results of distributed algorithms, as discussed in (Markiewicz and Koperwas, 2019), (Sattler et al., 2019) or (Hsieh et al., 2020). These studies have mainly analysed the skew in data label distributions and training communication costs to improve efficiency, particularly for non-IID data distribution. However, recent research has focused on rigorous FL, which has been extensively studied in terms of the algorithms used, FL systems as a whole, and benchmarking tools for these systems. The literature contains various approaches to validation (mainly tools or frameworks for benchmarking FL algorithms), as summarised in a recent study of such systems (Li et al., 2021b). Several publications have documented new benchmark systems (Caldas et al., 2018), (Luo et al., 2019) containing existing non-IID partitioned datasets. Most of the latest tools have focused on complete FL systems and the testing of their components (Hu et al., 2020), (Bouraqqadi et al., 2021), but have not placed particular emphasis

^a  <https://orcid.org/0000-0003-4229-8098>

^b  <https://orcid.org/0000-0003-0693-5109>

on a wide range of data partitioning strategies. Recent work (Liu et al., 2020) and (Li et al., 2021a) have demonstrated several approaches to generating non-IID datasets, nevertheless for (Li et al., 2021a) data generation only applies to provided datasets placed in their benchmarking system. In (Liu et al., 2020), the authors, among other things, simulate the problem of skew in the feature distribution and distort the image data by adding random noise, which is an interesting but unrealistic approach, since it distorts the real data attributes themselves. Many other tools are described in (Li et al., 2021b), although there is as yet no general consensus on how to properly evaluate distributed algorithms and how to define metrics for the non-IID-ness of data. Non-FL distributed processing tools and algorithms have also been intensively developed across the world. Like the FL benchmarking tools, these tools were created for the purpose of validating other DDM methods, and can be found in (Limón et al., 2019) and (Markiewicz and Koperwas, 2022). The authors of the latter started preparations to enable the evaluation of the algorithms in the case of non-uniform data distributions by allowing the use of various user-defined data partitioning strategies.

This paper represents a modest contribution to the ongoing discussions of data partitioning methods for simulating non-IID data distributions, with the aim of facilitating better algorithm evaluations. Our current research on data partitioning strategies focuses on the horizontal partitioning of data. We consider the impact and propose methods for simulating such data distributions in order to face the open issues around the use of distributed processing algorithms, as summarised recently in (Kairouz et al., 2019). We define various approaches to data partitioning that may be appropriate for different research questions, as mentioned in a recent paper (Li et al., 2021b). Although we cannot know the actual distribution for each distributed dataset in the real world, we can prepare the implementation of the algorithm to be more resistant to data distributions.

This paper is structured as follows. Section 2 refers to non-IID data partitioning and consists of two substantial and related subsections. Subsection 2.1 provides an overview of an extended taxonomy of non-IID data. The proposed data partitioning methods are described in Subsection 2.2. Section 3 explains the contribution and connection to the DDM-PS-Eval platform. Section 4 reports the results of experiments performed on the proposed data partitioning strategies. Finally, Section 5 concludes with a summary and plans for future work.

2 non-IID DATA PARTITIONING

In order to start designing partitioning strategies for each non-IID data distribution, it is required to know the origin of such distribution. The origin is directly related to the data itself, and it would be beneficial to know where it formally belongs in the hierarchy of data partitioning methods.

2.1 Extended Data Partitioning Taxonomy

Data may be split in multiple ways in independent data centres, or may simply be in separate physical locations. A taxonomy of types of partitioning was presented in (Kairouz et al., 2019) with regard to the problem of FL. In this paper, we introduce new levels to the 'Non-identical client distributions' part of the existing taxonomy. These new levels are related to partitioning strategies and data properties. Extending the existing taxonomy is required to create suitable and non-overlapping partitioning strategies for data distribution characteristics. This extension improves the categorisation of the new and existing methods within the taxonomy structure. It also helps to avoid strategy duplication for already handled non-IID distributions. The focus of our attention is on numerical or nominal data rather than on text, as partitioning such data is a different issue that needs to be investigated separately. Another brief categorisation of non-IID data related to images, which is partially beyond the scope of this work, and time-series data, is presented in (Zhu et al., 2021).

We started our work by investigating the main categories of non-IID distributions in order to determine which data properties are related to each category. There are three properties that can be used to describe a data distribution: the distribution of data attributes, the distribution of labels (classes)¹ of the samples, or the amount of data itself. These distributions can be observed together, which naturally defines it as a taxonomy rather than a tree-structured hierarchy. The listed properties form the parent level of the category level. The second additional level introduced here defines methods of realising the expected data partitioning for the purposes of quality evaluations of algorithms. Having top levels in place, we can correctly link partitioning strategies to the categories. The proposed extended taxonomy, illustrated in Fig. 1, shows data partitioning strategies associated with main categories and their relationship to data properties.

¹The terms "class" and "label" are used interchangeably in the text because their meaning is the same, but the nomenclature depends on the context.

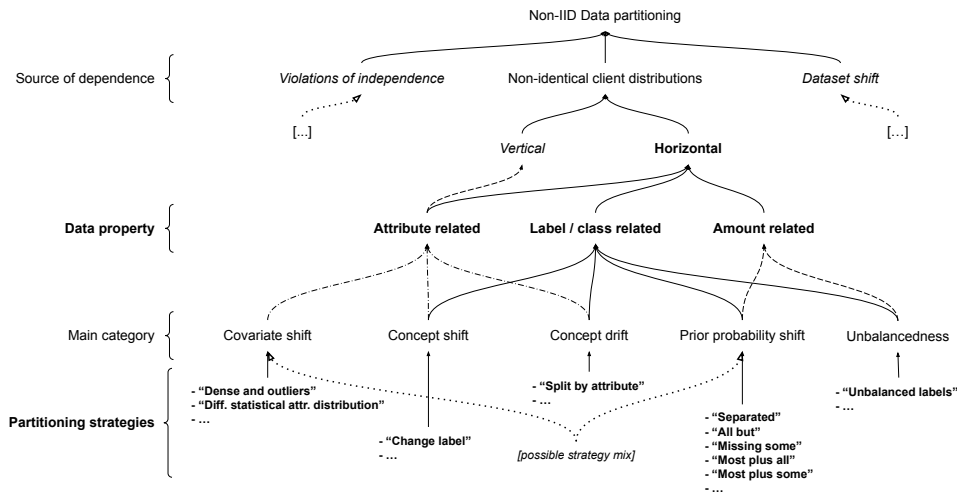


Figure 1: Extended non-IID data partitioning taxonomy, including new levels in bold (the line style of the arrows is not important, and is only used to indicate the groups).

The taxonomy presented in Fig. 1 gives us a more detailed view of data distribution possibilities and their origins. It also allows us to prepare a comprehensive test suite covering multiple non-IID data distributions to evaluate algorithms. Thanks to the taxonomy graph, we can quickly notice that both 'covariate shift', 'concept shift', and 'concept drift' categories relate to data attributes. At the same time, they do not consider the quantity amount of the data. On the other hand, 'prior probability shift' and 'unbalancedness' are closely related to the data quantity amount unconcerned about data attributes. Based on this observation, we can state that a minimal test suite for algorithm evaluation utilizing non-IID data requires examining at least two of five partitioning categories. Therefore, we cannot fulfil the correct but minimal algorithm evaluation without using at least one partitioning strategy that belongs to each data property. An example set of strategies that fulfil this condition would be the 'concept drift' and 'unbalancedness' partitioning that covers each data property introduced at the parent taxonomy level. A possible strategy mix used as a single test suite covering multiple data properties might be a different approximation for algorithm evaluation; however, mixing strategies is out of the scope for this work.

2.2 Data Partitioning Strategies

In the following sections, we provide details of the proposed partitioning strategies with examples for each main category. For each strategy, we assume that the dataset and number of target partitions are provided as input along with the strategy parameters.

2.2.1 Covariate Shift (Feature Distribution Skew)

One of the first problems that arise is that data may be placed in partitions with different distributions of attributes. In other words, the combined histogram for all 'split-separated' data collected from each partition would show relatively shifted attribute densities between the data in each partition. This is illustrated in Fig. 2 and described in more detail below. Although these partitions contain the same classes of data sample representatives, some of the attribute distributions are slightly different. Examples include typical differences in handwriting characteristics, speed limit signs that differ slightly between European countries (Grigorescu, 2018), or attributes describing the same plant species based on their size, colour, etc., which vary depending on the weather conditions and sunlight exposure.

In this method, we want to perform partitioning that will simulate this problem for a given dataset. It is easy to get nearly separate and evenly spaced data partitions based on a single attribute, by sorting the data according to the value of the attribute and dividing it into D equal parts. However, such parts have a data *shift* equal to $\frac{1}{D}$, which is the relative distance between the mean densities for these parts, as discussed below. Hence, to perform data splitting for different data shifts, we need to define an objective function for this problem. In our approach, we propose a function that consists of two user-defined parameters and unknown variables defined as mean data values $M_n; n \in [1; splits] \wedge n \in N$ for each data split. The first parameter (*splits*) is the number of partitions, and the

second parameter (*shift*) is the distance between the data distribution means of the splits, which is the target skew of the data. The objective function is determined by mathematical induction. We define the requirement for maintaining a certain ratio of the distances between the distribution centres of the splits in Equation 1:

$$\begin{aligned}
 \frac{M_2 - \min}{\max - \min} - \frac{M_1 - \min}{\max - \min} &= \text{shift} \\
 \Rightarrow \frac{M_2 - M_1}{\max - \min} &= \text{shift} \\
 \wedge \frac{M_3 - M_2}{\max - \min} &= \text{shift} \\
 \wedge \dots \Rightarrow \frac{M_n - M_{n-1}}{\max - \min} &= \text{shift}
 \end{aligned} \quad (1)$$

It follows from Equation 1 that an overall shift between first and last is a multiple of the *shift* parameter:

$$\begin{aligned}
 \frac{M_3 - M_2}{\max - \min} + \frac{M_2 - M_1}{\max - \min} &= 2 \cdot \text{shift} \\
 \Rightarrow \frac{M_3 - M_1}{\max - \min} &= 2 \cdot \text{shift} \\
 \Rightarrow \frac{M_n - M_1}{\max - \min} &= (n - 1) \cdot \text{shift}
 \end{aligned} \quad (2)$$

We can write the above expression as in Equation 3:

$$\begin{aligned}
 \frac{M_n - M_1}{\max - \min} &= \sum_{n=2}^{\text{splits}} \frac{M_n - M_{n-1}}{\max - \min} \Rightarrow \\
 \sum_{n=2}^{\text{splits}} \frac{M_n - M_{n-1}}{\max - \min} &= (n - 1) \cdot \text{shift}
 \end{aligned} \quad (3)$$

Since $n = \text{splits}$, we obtain the final restrictions for the objective function as in Equation 4:

$$\begin{cases} \sum_{n=2}^{\text{splits}} \frac{M_n - M_{n-1}}{\max - \min} = (\text{splits} - 1) \cdot \text{shift} \\ \forall n \frac{M_n - M_{n-1}}{\max - \min} = \text{shift} \end{cases} \quad (4)$$

$$\wedge \text{splits} > 1; \quad \text{shift} \in \left(0, \frac{1}{\text{splits}}\right]$$

where *min* and *max* are the minimum and maximum attribute values.

Solving this problem with these constraints is not an easy task. Our goal is to minimise the difference between the defined target function and the selected ratio. To achieve this, we need to search the space of possible solutions, which grows with the amount of data. We therefore approach this as a typical search task that involves finding a solution that meets certain conditions. This can be done in many ways, including brute force, random methods, or evolutionary algorithms. In our implementation, we choose the

simplest solution, which has two phases. We start by sorting the data based on the selected attribute value, to create an initial even split. We then iteratively move a random sample from one split to another, rolling back the operation if the objective function deteriorates. The stop condition is defined in the standard way, as a low epsilon value or a maximum number of iterations. Several strategies for moving the data samples were tested, including random splits and moving borderline data samples between partitions. However, the most straightforward solution turned out to produce the best results. Fig. 2 gives an example of the split obtained after applying the proposed method to a single attribute. We can see how the distribution of attribute values varies depending on the split numbers. Using a higher value for the *shift* param, we obtain far apart separated distributions with low overlapping for each split. Each split represents a data distribution dedicated to an independent target partition. For the low value of the *shift* param, attribute distribution across splits is more mixed; however, the mean values are notably disjoint.

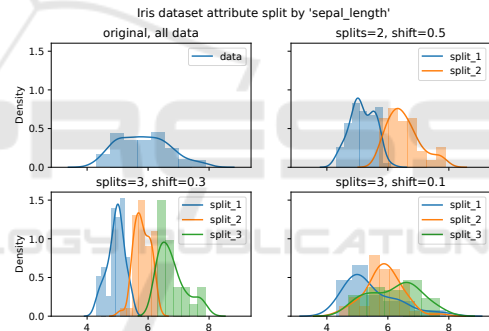


Figure 2: Example of attribute densities for the 'sepal_length' attribute of the Iris dataset, in the original case and after applying the proposed splitting method.

Splitting a dataset using a covariate shift for multiple attributes at the same time is possible, and requires that the function in Equation 4 is fulfilled for each attribute independently. However, the task then becomes more complex, and it may be impossible to achieve an acceptable function result for some datasets. Nevertheless, experiments described in the next section show that "shifting" data based on even one attribute affects the quality of the distributed algorithms.

Another variation of the covariate shift approach that is worth mentioning is the use of the 'dense-and-outliers' partitioning strategy mentioned in (Markiewicz and Koperwas, 2022). The authors suggested that separating anomalies and boundary data from dense data clusters defined by a data-

specific measure of similarity affects the quality of distributed algorithms. Since the proposed method operates on attribute values and their distribution, it can be classified as a covariate shift partitioning strategy.

2.2.2 Prior Probability Shift (Label Distribution Skew)

Due to geographical or cultural differences, data samples from the same class may be stored on independent nodes in different amounts. A typical example of this would be the presence of kangaroos in various regions around the world: this animal lives in the wild only in Australia, although single individuals can be found in a few zoos around the world. Another situation is represented by people from different countries who emigrate and settle in groups that affect the distributions of the local communities. A third situation is the use of English in books or conversations, which is of course mainly found in the US or UK; however, this particular language is also used all over the world. On the basis of these examples, we can distinguish many possibilities of separating data distributions into independent data partitions. We therefore distinguish the following partitioning strategies:

- 'separated [data]', where it is assumed that data samples with the same class are present only on one partition.
- 'all [of the data] but [some]', where specific labelled data samples are missing from some partitions;
- 'most [of the data] plus some', where the data are partitioned according to labels and small subsets of them are then scattered among several independent partitions;
- 'most [of the data] plus all', which is the same as the previous strategy except that each subset is also spread over each independent partition. This scenario simulates the case where each partition contains data for each label but with a different distribution;
- 'with anomalies', where most data are partitioned evenly but contain anomalies - certain labels in single quantities;

All of them are based on scattering data samples across different partitions but with different and specific distributions for their cases. The first three strategies assume main concentrations of data samples with certain classes and several classes missing within partitions. The last two strategies require representatives of all data classes to be kept together on every partition but in different amounts. Based on analyses and

numerous experiments performed to receive distinguished distributions, we have concluded that a single but parameterised method is sufficient to address each of the abovementioned strategies, rather than preparing separate, dedicated partitioning methods for each of them.

The proposed method is divided into two stages, related to labels and data quantities. First, we separate each class or group of classes depending on the number of partitions. We then determine additional classes for partitions and classes for empty partitions². When the number of partitions is greater than the number of unique labels in the dataset, we need to choose how to fill the empty partitions, as there are not enough unique labels available for each partition. In our method, we have two options: to fill empty partitions evenly with samples of all labels, or select the number of labels to be filled with data samples. The most treacherous part is to avoid adding the same label on every partition. We therefore use a cyclic iterator concept by adding classes in order which aligns the distribution of label presence across partitions. As a result of the first stage, we create tuples with three items: main labels of the partition, additional labels, and labels to fill empty partitions. An example of the abovementioned partitioning methods is given in Fig. 3. The figure shows the final algorithm assignment of different labels numbered from 0 to L to each group of labels within D partitions.

L - labels $> D$ - partitions	Legend:	L - labels $< D$ - partitions
EmptyAdd=0	1st [] - majority labels 2nd [] - additional labels 3rd [] - empty partition labels	
Separated:	d1=[0, 3] + [1, 2] + [] D=3 d2=[1, 4] + [3, 0] + [] L=5 d3=[2] + [1, 3] + [] Add=0	Separated:
All but:	d1=[0, 3] + [1, 2] + [] D=3 d2=[1, 4] + [3, 0] + [] L=5 d3=[2] + [1, 3, 4] + [] Add=-1	Separated:
Most plus some:	d1=[0, 3] + [1, 2] + [] D=3 d2=[1, 4] + [3, 0] + [] L=5 d3=[2] + [1, 3] + [] Add=2	Separated:
Most plus all:	d1=[0, 3] + [1, 2, 4] + [] D=3 d2=[1, 4] + [0, 2, 3] + [] L=5 d3=[2] + [4, 1, 3, 0] + [] Add=all	Separated:
		EmptyAdd=all
		Separated:
		All but:
		Most plus some:
		Most plus all:
		EmptyAdd=1
		EmptyAdd=2

Figure 3: Examples of different label partitioning strategies depending on the parameters L (number of unique labels), D (number of partitions), Add (additional number of labels), and $EmptyAdd$ (additional number of labels for empty partitions).

The second step uses simple mathematical proportions based on the provided parameters and the amounts of data to be transferred from the main subsets. At the end of the process, the data samples are randomised with a uniform distribution for each label according to the defined dependencies, and then split

²An empty partition is one without a dedicated majority of data samples for a specific label, which is possible when the number of unique labels is lower than the number of target partitions.

between partitions. Both steps require several parameters related to the label separation itself and the quantity of data samples to be distributed. The first pair is *additionalClasses* and *emptyPartitionClasses*, which determine how many additional classes should exist in the partition despite the initial number of labels present in the partition, and how many labels add to empty partitions. The second pair is *emptyPercent* and *additionalPercent*, which describe the maximum percentage of the data that should be used to fill empty partitions, and how much data should be used as additional padding for other partitions. To keep the majority in the initial subsets, the first should not be less than 50%, and the second should be at most half the value of the first parameter. The same concept is described in a recent paper by (Markiewicz and Koperwas, 2019), with reference to different strategies for preparing uneven data distributions.

There are many parameterisation possibilities, which depend on the dataset used, the number of labels, and the amount of data. By looking at the final result of this partitioning process, we can clearly see that it is closely related to both the label and the amount in terms of the data property. In itself, it does not take attribute values into account; although this is possible, it would involve mixing different non-IID partitioning main categories rather than a single aspect of the data distribution.

2.2.3 Concept Drift (Same Label, Different Features)

The term 'concept drift' refers to a problem of distributed learning training phase in which a certain set of data objects from the same class on each independent node is represented by samples with disjoint attribute values. It can refer to a fully disjoint set of attribute values between nodes or simply a different value distribution for some attributes in different nodes. The most common example where this issue occurs is a classification task for pictures with houses, where photos of houses are taken at different times of the year or under different weather conditions. Another example is the features of people from different regions of the world who differ in terms of their height, weight, hair colour, etc., which has an impact when exploring data on medical conditions. To simulate this type of data distribution, the dataset with the selected label should be separated to obtain the most separable data sample groups possible based on the attribute values. A simple technique would be to apply a partitioning algorithm to a subset to find a specific number of groups; however, this may be a non-deterministic method. We took a different approach, and our algorithm is defined in Algorithm 1.

Algorithm 1: 'concept drift' partitioning strategy.

Input Dataset *Data*, target label *L*, and partition "drifts" *D*

Output *Data* partitioned into *D* partitions

- 1: Extract a data subset *Data* with label *L*;
 - 2: [optional] Discretise the numerical attributes into *R* ranges for further processing;
 - 3: Group data into *buckets* consisting of samples with the same attribute values, in order to separate the *individuals* and groups of disjoint sets of attribute values, where *individuals* are samples or buckets of samples with the unique values of all attributes. A disjoint set refers to a group of samples whose values for each attribute in every sample are disjoint to values for each attribute of every sample from other sets. Fig. 4 gives an example of this process. This example shows the result and the second pass after processing step 5.2.;
 - 4: [optional] For a dataset that contains only nominal attributes, find *individuals* - they are candidates for scattering anywhere because of completely different feature values than other samples; they can be treated as data anomalies;
 - 5: Check whether it is possible to perform the selected "drift" from the groups found by checking the conditions: $|buckets \setminus individuals| \geq D$ and $|bucket| > \frac{|Data|}{D} \cdot r; \forall bucket$; where *r* - ratio with default, but possibly parameterisable value 0.2, to avoid highly unbalanced partitioning:
 - 5.1: If the conditions are met, or all the attributes have been checked, partitioning begins;
 - 5.2: Otherwise, remove *individuals* from the *Data* subset, exclude the single *Data* attribute with the lowest entropy value, and repeat the steps, starting with step 3;
 - 6: Partitioning is done by dividing the found *buckets* sorted by the value from the last examined attribute into *D* partitions, where the number of samples is divided between them as evenly as possible;
 - 7: If any *individuals* are found, place them into partitions to fill or equalise sample quantities between partitions;
 - 7.1: [optional] For numerical data, the target partitions for *individuals* are those closest to the centre in terms of the Euclidean distance (note that without the discretisation performed in Step 2, there is usually a large number of *individuals*);
-

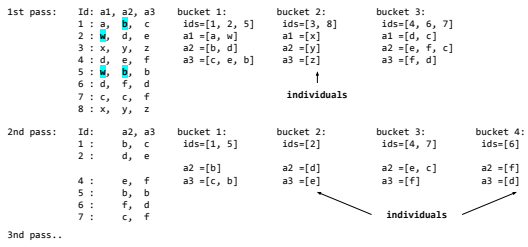


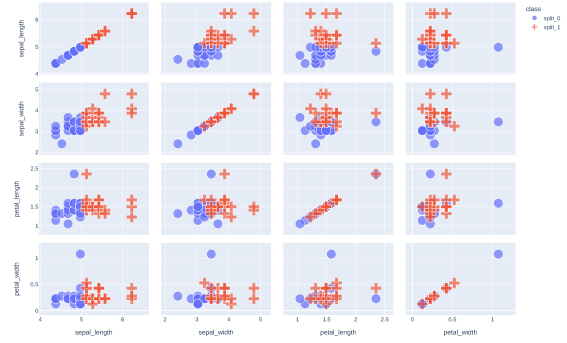
Figure 4: An example of found *buckets* with *individuals* for the first two passes of the 'concept drift' strategy. Bolded value with cyan background marks the attribute values responsible for grouping samples into the first bucket. Sample with id 5 affects the unrelated to each other samples with ids 1 and 2, as they have a transitive similarity: $1 \sim 5 \wedge 2 \sim 5 \implies 1 \sim 2$.

Although we assume here that drifting is performed for one selected label, the same algorithm can be applied to others. Examples of the partitioning results for the Iris dataset and the two-dimensional numerical single ellipse Gaussian dataset are shown in Fig. 5. With the use of colours and shapes, it is easy to notice how data samples are divided separately from each other. The dataset firmly separated represents the same object class, e.g. the same genre of flower or cluster of points; however, divided as much as possible after attribute distribution examination. As a result, we obtain well-separated subclasses.

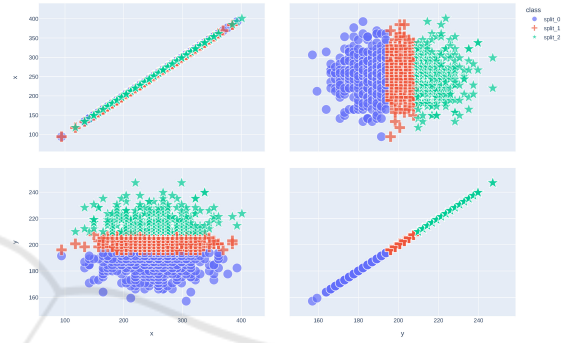
However, this approach may fail if the dataset cannot be split in this way, in which case the method processing comes down to splitting the data sorted by the most diverse value of a single attribute. For images or textual datasets, using the proposed approach should additionally consider either a prior aggregation of attributes or a reduction in the number of dimensions. Without this preprocessing step, the algorithm would try to carry out an analysis at too low a level of detail, such as a single pixel value, which is irrelevant in the overall scope.

2.2.4 Concept Shift (Same Features, Different Label)

The concept of this issue relates to a divergent class identification of data objects described by the same features in different regions, or simply a different understanding of what is represented by the characteristics when marking the data used for training. This is more related to the NLP process for sentiment analysis problems, where words can be tagged in different ways in the training dataset depending on the understanding. An example of nominal data can be a different understanding of a high or wealthy person depending on the country. When classifying non-text data, it is difficult to avoid errors in processing. It is



(a) Iris dataset - 'Iris-setosa' attribute.



(b) 2-dimensional single cluster dataset.

Figure 5: Scatter matrix plots for single label data partitioned using the 'concept drift' strategy.

almost impossible to evaluate the correct solution, as training is based on previously known labels. When the features of data objects are similar but labelled differently, the only way to get a correct result is to use fuzzy classification, where the classified sample can have multiple classes assigned to it. However, the clustering task is not affected by this problem at all, as it is an unsupervised learning method that does not involve an analysis of the training data classes. Due to the conditions mentioned above, the proposed method of data partitioning consists of changing the label to a new, unique one for a certain number of data samples. Our implementation of this partitioning strategy allows us to change the label of training samples for a subset of data with a selected class. This training subset of size $s \cdot \frac{1}{N}$ is then separated and the labels are changed, where s is the number of parts of the data subset to perform an independent label change, and N is the number of partitions. Fig. 6 gives an example of this type of partitioning. The exemplary dataset consists of data samples of two different labels required to be divided into three partitions and create two more classes using 'concept shift'. As a result, we obtain four classes distributed on three partitions where two classes are artificially created from the chosen one.

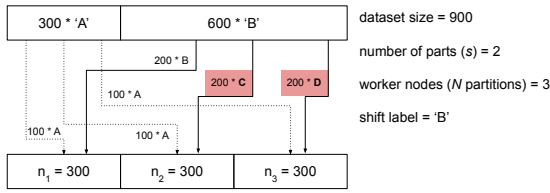


Figure 6: 'concept shift' partitioning strategy applied to data samples labelled 'B'.

The overall purpose of using this method for validation is to see how much such a divergent understanding of object class may affect the final results. This can be useful when the evaluated algorithm produces a fuzzy result or returns artefacts describing samples rather than a simple classification label.

2.2.5 Unbalancedness (Quantity Skew)

The last method of data partitioning described in this paper is the preparation of unbalanced data in terms of the number of samples. In this case, we assume that some nodes have more data samples, and others have significantly fewer. The partitioning method can be trivial by using random data partitioning along with the probability weights of presence in each partition. However, it would be difficult to maintain the ratio between sizes of those partitions with more data samples and other partitions for various numbers of target partitions. A simplified diagram illustrating parameterisation for this approach to obtain a 1/4 ratio is shown in Fig. 7.

However, our intention was to allow the user to choose this ratio along with the number of nodes that should contain more data samples. In order to make this possible, a simple system of two equations has to be solved, in which we have two unknown variables and four known parameters. Equation 5 describes the conditions to solve in order to find unknown values L and S , where L is the large amount of data and S is the small amount of data; the known parameters from the data description are C and W , where C is the dataset size and W is the number of target partitions; and the user-defined parameters are node threshold T as the number of target partitions that should have more data, and ratio U which is the target unbalancedness.

$$\begin{cases} T \cdot L + (W - T) \cdot S = C \\ \frac{S}{L} = U \end{cases} \implies \begin{cases} L = \frac{C}{U \cdot (W - T) + T} \\ S = \frac{C - T \cdot L}{W - T} \end{cases} \quad (5)$$

Based on the previous example in Fig. 7, we can obtain these numbers by using parameter values of 0.25 for the unbalancedness ratio and two nodes as the threshold. In our implementation of this method,

it is also possible to select a proportional imbalance to keep the same ratio for each class as a result of data partitioning. However, this behaviour is usually undesirable, as it tends towards the IID data distribution. As a result of this partitioning strategy, the data samples are scattered among the partitions in the number of the calculated partition sizes: L and S . Samples are chosen from the original dataset by a uniformly random function.

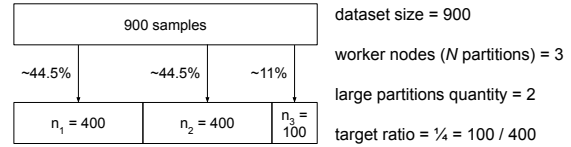


Figure 7: The trivial approach of finding the partition size division.

It should also be noted that when the imbalance factor is high, we do not consider 'unbalancedness' alone, but in conjunction with the 'prior probability' category, which in this case is related to the 'Label / class related' data property defined in the taxonomy. This is another indication that we are considering a taxonomy rather than a hierarchy.

3 DDM-PS-EVAL PLATFORM PARTITIONING COMPONENT

The DDM-PS-Eval platform created and described by (Markiewicz and Koperwas, 2022) is a tool used to perform comprehensive distributed data mining algorithms evaluation. The most exciting element in this platform referring to this work is the possibility of running experiments on the same dataset but with different data distribution. This platform contains built-in partitioning strategies and has been designed to be easily extendable with new custom ones. It is possible to write custom implementations following the given interface rules and use them in the platform.

A crucial part of this work was to prepare working implementations of partitioning strategies that expand algorithms evaluation with different non-IID data distributions. Moreover, using the extended taxonomy presented in section 2.1, already existing strategies have been arranged into the main categories:

- 'uniform', is treated as reference IID partitioning and kept outside of non-IID data partitioning taxonomy;
- 'label-strategy', during this work has become generalised and placed into the 'prior probability shift' category;

- 'dense-and-outliers', aforementioned also in section 2.2.1 and refers to the 'covariate shift' category.

We have prepared implementations of partitioning strategies for the three missing categories in the platform. These were 'concept drift', 'covariate shift', and 'unbalancedness'. Adding them completes the set of non-IID from the 'Non-identical client distributions' taxonomy branch.

The overview of the most interesting and data-related components of the platform is presented in Fig. 8. The platform allows us to perform multiple

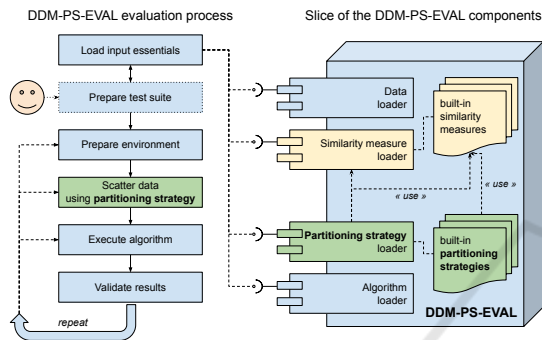


Figure 8: Fragment of the DDM-PS-Eval platform with essential components related to data partitioning; a simplified algorithm evaluation process on the left.

parameterised experiments using different built-in or custom partitioning strategies. Partitioning strategy components marked with a green background are responsible for this process.

3.1 Data Similarity Measures

Commonly, algorithms operating on numerical data use Euclidean distance to compare data samples; however, it is not a rule. Cosine distance works better to compare a vector representation of text documents. On the other hand, specialised algorithms can work with custom similarity metrics. Comparing genomic data described by specialised structures is an excellent example. The DDM-PS-Eval platform is prepared to load and use custom data similarity metrics for partitioning strategy alongside built-in metrics. The already existing 'dense-and-outliers' strategy uses this feature to calculate the distance between data samples depending on data and the provided similarity measure. Nevertheless, the proposed in this work strategies do not require any metric. They work directly on the data attributes, labels or amounts related to labels. On the other hand, to achieve a more sophisticated distribution for 'prior probability shift', moving more similar samples with the same class together between

partitions instead of random choice would be beneficial.

4 EXPERIMENTS

In order to verify the validity of the proposed partitioning strategies, we carried out several experiments. To illustrate the impact on the quality of the results, we ran selected distributed algorithms for datasets that were independently partitioned using multiple strategies.

For the evaluation, we used four different distributed clustering algorithms and two simple distributed classifiers. The classifiers were a distributed implementation of naive Bayes, which builds global a priori probability statistics based on the collected probabilities from local nodes, and the naive distributed SVM approach described in (Navia-Vázquez et al., 2006), tentatively named DN-SVM. In the latter approach, the final SVM model is trained on locally found support vectors sent from local nodes, which are treated as global training data. The clustering algorithms were DK-means (Ji and Ling, 2007), Opt-DKM (Markiewicz and Koperwas, 2019), a lightweight clustering technique (Aouad et al., 2007) tentatively named LCT, and a modified version of the BIRCH algorithm (Zhang et al., 1997), which used a partitioning approach for the final clustering. This modification was made by combining the agglomerative process in BIRCH with a partitioning method. BIRCH clusters the data at the local nodes and sends the cluster centroids to the global node, where the same operation is performed by treating the centroids as data samples to be clustered. Finally, global clustering is done by assigning samples to the closest global centroid.

The data used in the experiments included well-known datasets from the UCI Machine Learning Repository, such as the unbalanced Shuttle dataset and the tiny MNIST dataset. We used the PCA algorithm to preprocess and reduce MNIST data dimensionality from 28x28 into 28 attributes to avoid applying partitioning strategies on raw pixel values. We also used the noiseless "Dataset 1", which was introduced with the CURE algorithm in (Guha et al., 1998), and a dataset generated as two-dimensional points arranged into Gaussian clusters with some overlapping outliers to avoid perfect clustering results. The first two of these are primarily used for the task of classification, whereas the last two are used for clustering purposes. The full parameterisation of the experiments is given in Table 1.

Table 1: Summary of the test suite with experimental configurations.

Data (classes/groups, number of samples)	Shuttle (7, 58.000), MNIST (10, 70.000), CURE (5, 2.000), Synthetic-Gaussians (7, 100.000)
Algorithms	Naive Bayes, DN-SVM, DK-means, Opt-DKM, LCT, DP-BIRCH
Worker nodes / partitions	Three workers for the first three literature datasets Four workers for the synthetic dataset
Partitioning strategies (parameterised separately for each dataset)	uniform, dense-outliers, covariate-shift, separated, most-plus-some, most-plus-all, all-but, concept-drift, concept-shift, unbalancedness

The results of the experiments on distributed classification are shown in Fig. 9, and those for distributed clustering are shown in Fig. 10. In each chart, the first black bar on the left represents the results for uniform data distribution, and the rest represents the other partitioning strategies. For the non-deterministic algorithms, each bar shows the average result of 30 executions, with error bars representing the minimum and maximum values of all executions. All experiments were carried out on the DDM-PS-Eval platform presented in (Markiewicz and Koperwas, 2022) hence all the methods proposed are ready to use, and have been implemented in the way suggested by the authors of (Markiewicz and Koperwas, 2022). This allowed us to collect multiple forms of information, such as the processing time, transfer load, and final quality; however, for clarity of presentation, only the quality³ is presented in the charts. The quality measure used for clustering is the adjusted Rand index, whereas for classification we use the F-score in response to the unbalanced number of classes in the Shuttle dataset. The number of partitions used is small, due to the small number of classes and data samples, since the goal is to show the impact of non-IID partitioning rather than to add artificial complexity to the problem.

The primary goal of any algorithm design is to obtain high-quality results regardless of data distribution. However, in many implementations of distributed algorithms, the results are affected by different data dispersion. Due to its nature, a distributed implementation of the Naive Bayes algorithm produces exactly the same results, regardless of the partitioning strategy used. The expected exception in experimental results is poor accuracy obtained for the tested classifiers using 'concept shift' partitioning strategy, as they are not fuzzy classifiers. The second classifier encountered problems with highly segregated data, where the number of support vectors was small. The clustering results showed how the mean and maximum possible values of the clustering quality might

³Full results as raw data and an interactive format are available at <https://github.com/Kajo0/icsoft-2022-experiments-results>

vary depending on the data partitioning. It was found that even using the k-means++ initialisation introduced in (Arthur and Vassilvitskii, 2006) did not help to avoid the impact of different data partitioning methods on the results. The effects of the different data partitioning methods on the final clustering are very different, as shown in the results of the DK-Means and Opt-DKM algorithms. It is also interesting that in most cases, uniform data distribution made it difficult to obtain the best results, which was not expected at the outset. However, this can be explained by the density-based nature or non-deterministic initialisation of these algorithms.

An important implication of these findings is that we can roughly summarise how each partitioning strategy affects the algorithms examined here, as presented in Table 2. We cannot state that a particular category of non-IID data affects calculations or results for particular types of algorithms, for example partitioning or density-based methods. This is because modern distributed algorithms are usually based on an ensemble approach, and typically combine different methods for better results. However, we can certainly determine the approximate level of the possible impact of each main non-IID category on the algorithm, as shown in Table 2. This approximation may form a starting point for authors aiming to optimise the implementation of their algorithm.

We also need to remember that the impact of non-IID data partitioning on the quality of the algorithms also depends on the characteristics of the data itself without partitioning. This is illustrated in Figs. 9 and 10, where the distributions of the results for the same algorithm differ between the datasets; this is clearly visible for the DK-means algorithm, for instance. Moreover, it is obvious that there are countless test case parameterisations, including the selection of dataset attributes, the size of the "drift" or "shift", and the number of target partitions. For this article, we have selected more interesting results for the different partitioning strategies to emphasise the impact of the data distribution. Depending on the behaviour of the algorithm, the results may be better or worse,

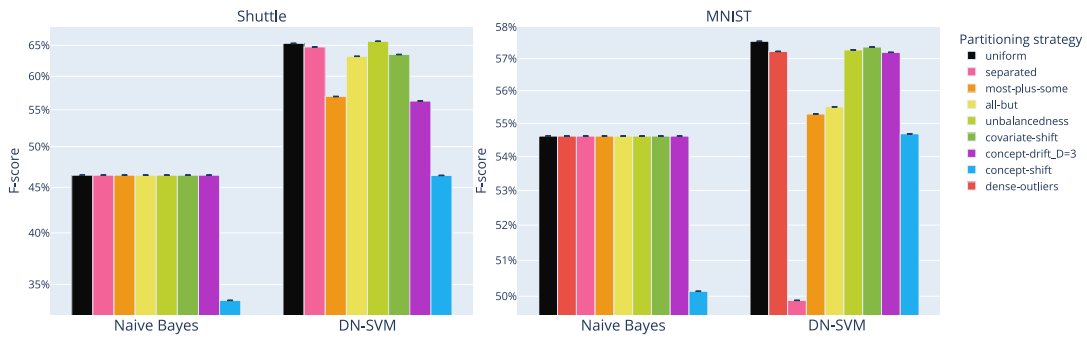


Figure 9: Comparison of results obtained for classification using a logarithmic scale.

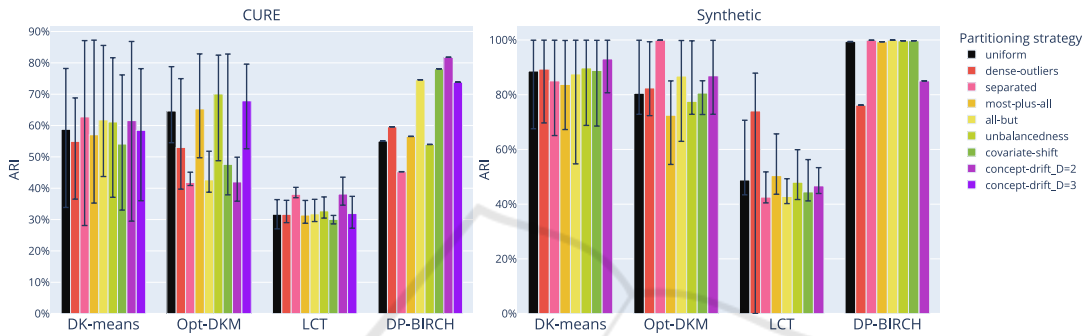


Figure 10: Comparison of average results obtained for clustering.

or may not change at all. For example, in the case of distributed versions of density-based algorithms, data separation is typically an obstacle in subsequent phases of global aggregation. For this reason, it is crucial to validate algorithms using multiple data partitioning strategies instead of a single uniform one.

All the methods proposed represent only a few possibilities for the data distributions, and many more are possible. At this point, it should be noted that in terms of data partitioning, numerical data such as images with large numbers of dimensions should be treated differently from data collected from surveys, which are used for classification purposes. Moreover, in some label-related strategies that perform dataset separation based on the sample class, the number of target partitions plays a role, which opens up another discussion on how to deal with this.

5 CONCLUSIONS AND FUTURE WORK

In this work, we extended the existing non-IID taxonomy by two levels. The first is the data property as the parent level of the main non-IID category. Adding this level then allowed us to determine the relation between data attribute and specific partitioning

strategies defined as the new leaf-level of the taxonomy. We have presented multiple dataset partitioning methods that simulate non-IID data dispersion for numerical and nominal data, with detailed explanations and examples for each category. The proposed methods have been evaluated for distributed clustering and classification algorithms using several datasets and various algorithms. Results for different parameterisations have been presented and discussed. Each described data partitioning strategy has been implemented and is ready to work with the DDM-PS-Eval platform. In conclusion, this study has shown that there is a significant impact of data partitioning on the results provided by distributed algorithms. This work paves the way for better validation of algorithms, in order to allow for the design of algorithms that are data distribution-agnostic.

Further research on different data partitioning schemes, such as for textual datasets, is necessary to extend our evaluation to other types of algorithms. Future work will involve extending the proposed methods to produce more realistic distributions of non-IID data by mixing multiple partitioning strategies based on attributes, labels, and quantity. Several open issues need to be analysed and addressed, such as the aforementioned mixture of strategies and dealing with a large number of partitions for data with few labels.

Table 2: Summarised ranges of the negative impact of each partitioning strategy on the examined algorithms and datasets (L, M, H indicate low (<5%), medium (<10%), high (≥10%) impact, respectively, and 0 indicates no noticeable impact or results that are better than those obtained for uniform data distribution; for non-deterministic algorithms, separate differences were taken into account for both mean and maximum values collected from multiple executions; bold cells indicate potentially high impact).

Main non-IID category of data partitioning strategy	Naive Bayes	DN-SVM	DK-means	Opt-DKM	LCT	DP-BIRCH
Covariate shift	0	0 - L	0 - M	0 - H	0 - H	0 - H
Concept shift	M-H	H	0	0 - L	0 - H	0 - L
Concept drift	0	0 - M	0 - L	0 - H	0 - H	L - H
Prior probability shift	0	0 - H	0 - L	0 - H	0 - H	0 - M
Unbalancedness	0	0	0 - L	0 - L	0 - H	L - H

REFERENCES

Aouad, L. M., Le-Khac, N.-A., and Kechadi, T. M. (2007). Lightweight clustering technique for distributed data mining applications. In *Industrial Conference on Data Mining*, pages 120–134. Springer.

Arthur, D. and Vassilvitskii, S. (2006). k-means++: The advantages of careful seeding. Technical report, Stanford.

Bouraqqaadi, H., Berrag, A., Mhaouach, M., Bouhoute, A., Fardousse, K., and Berrada, I. (2021). Pyfed: extending PySyft with N-IID Federated Learning Benchmark. *Proceedings of the Canadian Conference on Artificial Intelligence*. <https://caiac.pubpub.org/pub/7yr5bkck>.

Caldas, S., Duddu, S. M. K., Wu, P., Li, T., Konečný, J., McMahan, H. B., Smith, V., and Talwalkar, A. (2018). Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*.

Grigorescu, S. M. (2018). Generative one-shot learning (gol): A semi-parametric approach to one-shot learning in autonomous vision. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7127–7134. IEEE.

Guha, S., Rastogi, R., and Shim, K. (1998). Cure: An efficient clustering algorithm for large databases. *ACM Sigmod record*, 27(2):73–84.

Hsieh, K., Phanishayee, A., Mutlu, O., and Gibbons, P. (2020). The non-iid data quagmire of decentralized machine learning. In *International Conference on Machine Learning*, pages 4387–4398. PMLR.

Hu, S., Li, Y., Liu, X., Li, Q., Wu, Z., and He, B. (2020). The oarf benchmark suite: Characterization and implications for federated learning systems. *arXiv preprint arXiv:2006.07856*.

Ji, G. and Ling, X. (2007). Ensemble learning based distributed clustering. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 312–321. Springer.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. (2019). Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*.

Li, Q., Diao, Y., Chen, Q., and He, B. (2021a). Federated learning on non-iid data silos: An experimental study. *arXiv preprint arXiv:2102.02079*.

Li, Q., Wen, Z., Wu, Z., Hu, S., Wang, N., Li, Y., Liu, X., and He, B. (2021b). A survey on federated learning systems: vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering*.

Limón, X., Guerra-Hernández, A., Cruz-Ramírez, N., and Grimaldo, F. (2019). Modeling and implementing distributed data mining strategies in jaca-ddm. *Knowledge and Information Systems*, 60(1):99–143.

Liu, L., Zhang, F., Xiao, J., and Wu, C. (2020). Evaluation framework for large-scale federated learning. *arXiv preprint arXiv:2003.01575*.

Luo, J., Wu, X., Luo, Y., Huang, A., Huang, Y., Liu, Y., and Yang, Q. (2019). Real-world image datasets for federated learning. *arXiv preprint arXiv:1910.11089*.

Markiewicz, M. and Koperwas, J. (2019). Hybrid partitioning-density algorithm for k-means clustering of distributed data utilizing optics. *International Journal of Data Warehousing and Mining (IJDWM)*, 15(4):1–20.

Markiewicz, M. and Koperwas, J. (2022). Evaluation platform for ddm algorithms with the usage of non-uniform data distribution strategies. *International Journal of Information Technologies and Systems Approach (IJITSA)*, 15(1):1–23.

Navia-Vázquez, A., Gutierrez-Gonzalez, D., Parrado-Hernández, E., and Navarro-Abellan, J. (2006). Distributed support vector machines. *IEEE Transactions on Neural Networks*, 17(4):1091.

Sattler, F., Wiedemann, S., Müller, K.-R., and Samek, W. (2019). Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 31(9):3400–3413.

Zhang, T., Ramakrishnan, R., and Livny, M. (1997). Birch: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141–182.

Zhu, H., Xu, J., Liu, S., and Jin, Y. (2021). Federated learning on non-iid data: A survey. *arXiv preprint arXiv:2106.06843*.