



Research paper

Node and relevant data selection in distributed predictive analytics: A query-centric approach

Tahani Aladwani^a, Christos Anagnostopoulos^{a,*}, Kostas Kolomvatsos^b^a School of Computing Science, University of Glasgow, UK^b Department of Informatics and Telecommunications, University of Thessaly, Greece

ARTICLE INFO

Keywords:

Distributed predictive analytics
Distributed machine learning
Query-centric node selection
Data relevance

ABSTRACT

Distributed Predictive Analytics (DPA) refers to constructing predictive models based on data distributed across nodes. DPA reduces the need for data centralization, thus, alleviating concerns about data privacy, decreasing the load on central servers, and minimizing communication overhead. However, data collected by nodes are inherently different; each node can have different distributions, volumes, access patterns, and features space. This heterogeneity hinders the development of accurate models in a distributed fashion. Many state-of-the-art methods adopt random node selection as a straightforward approach. Such method is particularly ineffective when dealing with data and access pattern heterogeneity, as it increases the likelihood of selecting nodes with low-quality or irrelevant data for DPA. Consequently, it is only after training models over randomly selected nodes that the most suitable ones can be identified based on the predictive performance. This results in more time and resource consumption, and increased network load. In this work, holistic knowledge of nodes' data characteristics and access patterns is crucial. Such knowledge enables the successful selection of a subset of suitable nodes for each DPA task (query) before model training. Our method engages the most suitable nodes by predicting their relevant distributed data and learning predictive models *per* query. We introduce a novel DPA query-centric mechanism for node and relevant data selection. We contribute with (i) predictive selection mechanisms based on the availability and relevance of data per DPA query and (ii) various distributed machine learning mechanisms that engage the most suitable nodes for model training. We evaluate the efficiency of our mechanism and provide a comparative assessment with other methods found in the literature. Our experiments showcase that our mechanism significantly outperforms other approaches being applicable in DPA.

1. Introduction

1.1. Background

Distributed Machine Learning (DML) pushes model training, inference and prediction tasks to the network edge addressing concerns related to data privacy, centralized data transfer, and high communication costs (Ye et al., 2020). Traditionally, *predictive analytics* relies on collecting vast amounts of data in Cloud infrastructure before training ML models for tasks like classification and regression (Abbott, 2014). A contrasting approach, *distributed predictive analytics* (DPA) (Anagnostopoulos, 2020), where code and models for training and inference are distributed to places where data are collected, has been emerged due to (i) increased processing power and memory capacity available in e.g., distributed micro-/edge-servers and road-side units at the network edge, and (ii) increased demands for improved data privacy.

DPA rely on a federated learning based infrastructure that enables many disparate sources of data owned by many organizations to contribute to training and using large-scale DML models for inference by exchanging only models among distributed *nodes*. Models are trained using only data that are requested for specific DPA tasks, referred to as *queries*. These queries include tasks such as regression, classification, outlier and novelty detection, and missing value substitution.

DPA amalgamates distributed knowledge *without* requiring data sharing, thereby eliminating the dependency on centralized training, which necessitates data sharing (Feng and Yu, 2020). However, DPA is still in the early stages of developing models with quality that can be compared to centrally trained models for analytics queries. DPA copes with heterogeneous nodes in terms of data distribution, data size, and access patterns. Such heterogeneity hinders the development of accurate models. Understanding nodes' data characteristics

* Corresponding author.

E-mail addresses: Tahani.Aladwani@glasgow.ac.uk (T. Aladwani), christos.anagnostopoulos@glasgow.ac.uk (C. Anagnostopoulos), kostask@uth.gr (K. Kolomvatsos).<https://doi.org/10.1016/j.jnca.2024.104029>

Received 7 June 2024; Received in revised form 19 August 2024; Accepted 10 September 2024

Available online 19 September 2024

1084-8045/© 2024 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

like distribution, outliers, and quality, along with query access patterns across nodes, is essential. Such holistic knowledge enables the tailored selection of a subset of nodes for *each* DPA query. Such nodes should be involved in (i) predicting the relevant distributed data for each query and (ii) learning ML models for each query in a distributed fashion. However, this approach is neither directly applicable nor trivial in DML environments due to concerns about violating nodes' data privacy as elaborated in [Zhu and Sun \(2021\)](#).

The straightforward approach of selecting random nodes to engage in model training per query is not sufficient and effective especially when dealing with heterogeneity in query access patterns. Fundamentally, random node selection neither considers the aforementioned data characteristics nor the query requirements. A federated infrastructure for DPA based on random selection of nodes' data has a significant impact on massively distributed ML models ([Yang et al., 2021](#)). With the growth in real-time analytics tasks and the variability of distributed data, a comprehensive DPA task scheduling policy is required to ensure timely and efficient model training.

Ignoring this knowledge increases the likelihood of selecting arbitrary nodes with irrelevant data compared to *what* the model actually needs for training, as evidenced by [Anagnostopoulos \(2020\)](#). This inevitably leads to situations where nodes' local models negatively impact the final predictive performance due to potentially irrelevant training samples, as opposed to the data required by the queries. Consequently, aggregating irrelevant local models deteriorates the overall performance and quality of the model for each analytics query, as discussed by [Deng et al. \(2021\)](#), [Casado et al. \(2022\)](#), and [Hong et al. \(2022\)](#).

1.2. Motivation & challenges

In DML environments, nodes (e.g., edge servers and road-side units in smart cities) collect large-scale contextual data as shown in [Fig. 1](#). In practice, not *all* of a node's data are needed for a given query; however, some parts may be relevant and useful for improving model performance ([Kolomvatsos and Anagnostopoulos, 2022](#)). Our challenge is that these relevant data parts are usually distributed across nodes. Therefore, it is essential to leverage DPA to train a holistic model using *only the relevant data for each query* aiming to (i) improve prediction performance and (ii) avoid accessing irrelevant training data. Consequently, we must ensure that the model is trained exclusively with the required and relevant data *for each query*.

The objective of DPA on *query-centric node selection*, which significantly impacts the global model's quality, has not been thoroughly investigated. Most existing approaches use the random selection paradigm to engage nodes in distributed training, irrespective of the analytics queries, as in [Rai et al. \(2022\)](#), [Tran and Zheleva \(2022\)](#). Such approaches do not consider the specific requirements of queries that demand engaging only the most relevant and appropriate nodes. Our research question is: *How can we select only the relevant data from each chosen node per query given limited access to the data?*

Two fundamental strategies are to be established in node and data relevant selection challenge: (S1) Engage only those nodes in DML model training hosting the most appropriate data for each query *without* transferring data; (S2) Migrate data (or part of them) to nodes or the Cloud so DPA queries can be centrally processed ([Bellavista et al., 2019](#); [Boulougari and Kolomvatsos, 2022](#)). However, S2 strategy is not applicable in edge computing environments due to violation of nodes' data privacy and data transfer. Whereas, S1 strategy seems promising in DML environments, nonetheless, the key challenge still lies in selecting *which* data to be selected thus avoiding accessing irrelevant data for DPA query processing. This *data selectivity* challenge requires a DPA paradigm to *learn and predict* the relevant data that are expected to be accessed and processed in each query.

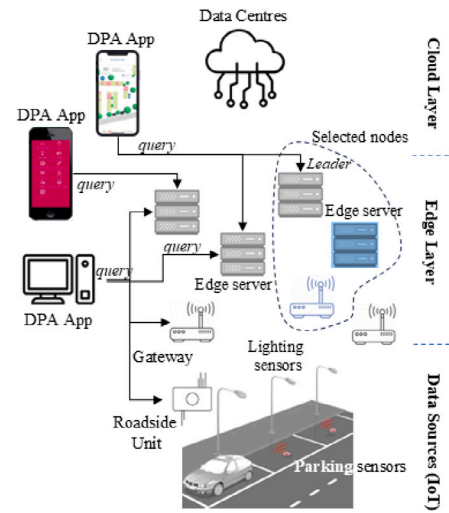


Fig. 1. Node & relevant data selection based on a query-centric paradigm. Nodes, e.g., edge servers, roadside units, receive DPA queries from DPA Apps to train ML models for predictive analytics. Generated data by Internet of Things (IoT) devices, e.g., sensors, are *selectively* accessed from only selected nodes (encircled in a dashed line) avoiding accessing irrelevant data for each DPA query.

We tackle this challenge by learning the relevant data across the data features exploiting the access patterns of queries issued by DPA applications (see [Fig. 1](#)). Learning of relevant data is achieved by extracting dependencies between queries and local datasets, while rejecting data that are irrelevant to model learning.

Example: Consider several smart city DPA applications for real-time monitoring of on-street parking, surface parking lots in shopping malls, train stations, and corporate campuses in a city as shown in [Fig. 1](#). Such applications initiate DPA queries across nodes installed in e.g., parking areas and road-side units for e.g., (i) ranking the places with the highest parking occupancy in the last 30 min, (ii) local (re)training of time-series forecasting models to predict the expected parking capacity (*per* parking area) in the next hour, (iii) updating city council recommender ML applications that inform drivers about available parking spaces in real-time (way-finding & interactive parking maps), (iv) updating local predictive maintenance ML models with the status of the surface sensors installed in each parking area, and (v) train traffic ML models for updating parking duration pricing based on city drivers' parking habits and daily schedules. The nodes deployed across diverse areas in the smart city should be able to predict which data they need to access for DML model training for each query improving e.g., urban decision making and planning. Awareness of *relevant* data emerged as a necessary feature of nodes ([Kolomvatsos et al., 2022](#)). Nodes benefit if they are aware of the trends and query access patterns of incoming DPA queries over their data ([Karanika et al., 2020](#); [Soula et al., 2022](#)).

Our query-centric paradigm is proposed to enhance data selection by balancing nodes' privacy concerns with the need for high-accuracy analytics and efficient data processing. In real-world analytics applications as above-mentioned, model performance depends not only on the data volume but also on its relevance to the training process. Often, only a small subset (such as 20 percent or less) of data in each node is relevant to a specific analytic task ([Kolomvatsos et al., 2022](#)). In our example in [Fig. 1](#), for instance, at time 11:00am, a DPA application generates a query requesting parking sensors' data in the last 30 min for those sensors which have been occupied from 30 to 60 min (vehicles detected parked from 30 min to 1 h) and stage-of-charge ranges between 60% and 80%. These data of interest need e.g., to train predictive models to forecast parking slot occupancy in a smart city or for incrementally updating pre-trained forecasting ML models. In e.g., banking sector, as another example, an analytic task

normally involves nested queries, such as determining the number of customers who have both a credit score between 700 and 800 and an annual income between \$50,000 and \$100,000, and who have made at least three mortgage payments in the last six months. Additionally, the task might require understanding the impact of these customers' loan repayment behaviors on the bank's overall risk profile. By measuring the overlap between the data each node hosts (e.g., parking sensor occupancy rate within a time window and sensor battery level, or credit scores and income levels) and the specific query requirements, one can efficiently retrieve the relevant data needed for training models that predict pollution levels in a smart city or default risk in light of optimizing urban planning or loan offerings, respectively. Such DPA queries are common across various sectors, including finance (Bagdasaryan et al., 2020), weather forecasting (Chen et al., 2023), healthcare (Shokri and Shmatikov, 2015), sports analytics (Boobalan et al., 2022), manufacturing (Wang et al., 2020), transportation and logistics (Jiang et al., 2024), energy (Cheng et al., 2022), and marketing (Nguyen et al., 2024), among many other fields that rely heavily on quantitative and predictive analysis. Compared to other methods (Zhu and Sun, 2021; Lee, 2022; Saha et al., 2022; Lee, 2022; Saha et al., 2022; Hammoud et al., 2022; Goetz et al., 2019; Ye et al., 2020; Zhu and Sun, 2021), our mechanism reduces the need to access 100% of the data in each node for relevant data selection and training purposes.

1.3. Contributions

In this work, we introduce DPA query-centric mechanisms for predicting the most appropriate subset of nodes involved in DPA *per* query. By leveraging on nodes' statistical signatures over their local data helps predict and rank superior nodes that accelerate the overall model accuracy *per* query while avoiding redundant access to irrelevant training data. The main technical contributions of our paper are:

- We introduce a novel proactive, query-centric node and relevant data selection mechanism for DPA. This approach focuses on identifying relevant data to be accessed in advance, as indicated by each DPA query. This, in turn, determines the suitability of a node to be engaged in the DML process.
- We propose query-centric DML algorithms that utilize relevant data and node ranking for each query. We further elaborate on the complexity and effectiveness of these algorithms.
- We provide a comprehensive performance evaluation and comparative assessment using real data and query workloads. Our evaluation showcases the efficiency of our mechanism across various DML mechanisms, compared against baseline approaches and the relevant methods Anagnostopoulos (2020), Ye et al. (2020), Hammoud et al. (2022), Puthiya Parambath et al. (2021), and Puthiya Parambath et al. (2024) found in the literature.

The rest of the paper is organized as follows. In Section 2, we shed light on the related work elaborating on node and node and relevant data selection. Section 3 provides extensive discussion of the theoretical aspect of our problem and elaborates on the problem formulation along with preliminaries and definitions. In Section 4, we introduce our node and relevant data selection mechanisms, while Section 5 elaborates on the proposed query-centric DML mechanisms. Section 6 reports on the comprehensive performance evaluation, comparative assessment, discusses on the limitations of our approach and future directions for enhancement. Section 7 concludes the article.

2. Related work

It has been evidenced in Lee et al. (2022), Zhou et al. (2023) and Anagnostopoulos and Kolomvatsos (2019) that the performance of a distributed learned model is greatly influenced by the quality of the distribution of data across the nodes. Therefore, many studies have focused on improving the model's performance through node selection

and/or data selection. In light of this, we provide an overview of the most relevant studies on node selection. Additionally, we show how relevant data have been selected within each chosen node, according to previous approaches.

2.1. Node selection in DPA

Most approaches adopt random node selection. Consequently, averaging across randomly selected nodes often leads to insufficient performance due to the heterogeneity of the nodes (Zhu and Sun, 2021). The heterogeneity of the nodes fundamentally determines how each individual node's data contributes to distributed model training. Ye et al. (2020) considered various factors for node selection, such as edge device computational capability, network bandwidth, and energy consumption. The decision mechanism in Goetz et al. (2019) is based on the importance of user data by inspecting the current loss of the model at each training round. Both aforementioned mechanisms perform node selection decisions *after* model training. This means that resources are consumed in advance, and then it is decided whether engaging a node is worth it. This is not applicable in our case. For a given query, we need to predict in advance the most suitable nodes to be engaged in DML training, thus avoiding unnecessary resource consumption.

Hammoud et al. (2022) introduced a mechanism where the selection criteria are based on choosing nodes with data different from what the models have previously learned. However, this approach also requires first training the model and then choosing the nodes. Additionally, it runs the risk of training a model on irrelevant data, which can be detrimental to the final model's performance. Saha et al. (2022) proposed using a data quality score, computation score, and communication score as criteria to quantify the capabilities of the selected nodes. Meanwhile, Lee (2022) introduced a reward selection function based on the remaining energy budget (for mobile edge nodes), expected computation load, and communication status. In both mechanisms, a model must be trained in advance to assess data quality and the computational load used.

Zhu and Sun (2021) proposed the Federated Trace mechanism to track model training and data distribution across a random set of nodes. Similarly, Lin et al. (2022) introduced a mechanism for nodes selection mechanism based on measuring each node's contribution in the previous round of the model training, which is mainly quantified as the prediction accuracy of the globally (and currently) trained model before and after aggregating it with each node's local model. In these approaches, it is mandatory a (global) model to have been trained in advance helping to choose node selection for future engagement. Finally, Huang et al. (2022) proposed a node selection approach based on a fairness mechanism. With such mechanism, each candidate node obtains the same chance to get involved during the training process. The fairness is quantified based on a pre-trained model (before node selection).

Node selection has been investigated in time-optimized sequential decision making and on-line learning systems. There is a category of online decision-making deriving from Reinforcement Learning (RL) known as the Multi-Armed Bandits (MAB) (Wang et al., 2022) tackling specifically this problem. There has been a rise in the utilization of MAB algorithms powered by RL examining the crucial balance between exploration and exploitation in sequential decision-making (Lai and Robbins, 1985; Ghoorchian and Maghsudi, 2020). The objective of MAB is to determine the best possible *arm* (node or sub-set of nodes in our case) from a group of possible arms (nodes) containing previously rewards. This is accomplished by picking a single arm sequentially and then tracking the reward that is realized (Auer et al., 2002). The contextual MAB problem expands the core MAB by incorporating reward functions that rely on the context (Yang et al., 2021; Puthiya Parambath et al., 2024) like queries in our case. Such works cast the node selection problem as a MAB problem in light of identifying the most appropriate

node for maximizing expected rewards. An additional expansion of the MAB, the combinatorial MAB problem permits choice of multiples groups of arms (Wu et al., 2020; He et al., 2020). Notably, the work in Puthiya Parambath et al. (2021) studies the impact of utility functions in these types of problems including node selection. In addition, the approach in Ren et al. (2020) adopts deep RL to choose appropriate distributed node locations while utilizing deep Q-learning method to ensure load balancing. The work in Yu et al. (2020) introduced deep RL to enable actual time and low-overhead computation offloading and allocation of resources. However, it is essential to point out that these studies do not take into account the possibility of reducing data access redundancy nor select a sub-set of nodes to be engaged in a specific query, which are not their principled objectives. In addition, regarding the choice of offloading task to edge server(s), our paradigm takes into consideration the decision in *which* node(s) to offload the model training task for each query, which relates to the query access pattern over the node's data. On the contrary, the above-mentioned methods do not take into consideration any query characteristics for learning task offloading decisions while some of them involving data migration and mandatory full data access; the former is not applicable in our case, while the latter is inefficient especially when involving irrelevant training data as elaborated in our comparative assessment in Section 6. Finally, the approaches in Alghamdi et al. (2021a), Alghamdi et al. (2021b) and Panagidi et al. (2020) address the problem of selecting nodes for tasks and/or data offloading based on the Optimal Stopping Theory. However, encrypted information trading, as opposed to open access to shared data, reduces confidentiality and privacy upon data offloading. Our paradigm diverges from simple task offloading by focusing on selecting a sub-set of the most suitable nodes derived from the overlapping between any arbitrary DPA query and their data for training ML models. In our eco-system there is no data offloading while the model training tasks are offered by the selected nodes. All the decisions are tailored to the incoming DPA queries, which yields the novelty of our paradigm in DML environments.

In all the above-mentioned approaches, it is required, at least, one model training round *before* choosing the participant nodes for model learning. This evidently yields unnecessary time, irrelevant data access, and resource consumption, which in principle does not hold in our mechanism. Nonetheless, the associated node selection decisions in such approaches are not tailored to DPA queries, thus, focusing only on building generic models untied from specific tasks. Contrary to our mechanism, these approaches are unaware of the tasks requirements and needs of predictive applications. Furthermore, the node selection decisions do not take into consideration the relevance of the data residing on the nodes with the incoming DPA queries. That is, even if a node is selected, its data should also be 'scrutinised' in advance to reassure that these are relevant for being used in the DML process. On the other hand, our decision mechanism is purely driven by feeding the distributed learning with the most relevant data out of the most suitable nodes as it will be elaborated later.

2.2. Data relevance in DPA

Since multiple nodes might be involved in the DML process, it is important to consider that not all data in the selected nodes may be relevant given a DPA query. It is crucial not only to select the most suitable nodes *but* also to identify the relevant data within these nodes. Including irrelevant data from nodes can negatively impact the performance of the derived predictive model. Excluding nodes whose some data are considered irrelevant might seem like a straightforward solution. However, such approach may not always be beneficial since it might lead to removal of valuable information that could have contributed to the predictive capacity of the models.

Tuor et al. (2020) proposed a mechanism to identify relevant data for model training based on a relatively small benchmark dataset. The benchmark-trained model evaluates the relevance of each individual

data sample at every node to determine the relevant samples in a centralized location (e.g., a server). This mechanism requires sharing each sample's prediction errors, which can reveal specific information about each sample. Nagalapatti et al. (2022) proposed a data relevance detection method using relevance scores for each sample, which account for inherent noise. In this approach, irrelevant (noisy) data are gradually excluded during the DML process. The data relevance is determined based on the noise of the training samples, regardless the specific DPA query. This differs significantly from our notion of data relevance. Both of the aforementioned mechanisms assess each data sample individually based on the model predictive performance. They do not consider the relevance specific to the DPA query. In contrast, our mechanism identifies relevant data as a fundamental part of the selection decision for each query.

To the best of our knowledge, our approach is the first proactive query-centric node and relevant data selection mechanism yielding in building tailored DML models. Our work drastically departs from our recent preliminary study in Aladwani et al. (2023) about the necessity and impact of identifying the relevant data in building DML models. This guided us to establish in this paper (i) the fundamentals of the node selection problem per DPA query, (ii) identifying the minimum sufficient statistics required for selecting the most suitable nodes in advance to be engaged in the DML training phase, and (iii) introducing query-centric DML mechanisms that leverage the selected nodes for building models tailored to DPA queries.

3. Problem fundamentals

3.1. Preliminaries & definitions

DML aims to train a model over a distributed dataset D estimating the input-output mapping $y = f(x; \theta)$, with d -dim. input $\mathbf{x} = [x_1, \dots, x_d]^T \in \mathcal{X} \subset \mathbb{R}^d$, output $y \in \mathcal{Y} \subset \mathbb{R}$, and θ being the model's parameters from a parameter space. Fundamentally, the difference with the centralized ML is that D is distributed over N nodes, denoted as $\mathcal{N} = \{n_1, n_2, \dots, n_N\}$. Each node in \mathcal{N} utilizes similar data features, such as weather data like humidity, temperature, and pressure, vehicle parking sensors data like occupancy duration, magnetic/inductive sensor type, air quality.

We assume a network of nodes represented via the graph $\mathcal{G}(\mathcal{N}, \mathcal{E})$ with \mathcal{N} nodes (vertices) and edges \mathcal{E} such that nodes n_i and n_j directly communicate (bilaterally) if there exist edges $e_{ij} = e_{ji} \in \mathcal{E}$. Each node $n_i \in \mathcal{N}$ has its own local dataset $D_i = \{(\mathbf{x}, y)_{\ell}^i\}_{\ell=1}^{L_i}$, which consists of L_i samples. Each dataset D_i represents a subset of the entire dataset $D = \cup_{i=1}^N D_i$. When a node n_i receives a DPA task represented as a query \mathbf{q} , this node n_i acts as the 'leader' for that query \mathbf{q} . The leader node is responsible for establishing a mechanism to select the most suitable nodes $\mathcal{N}' \subset \mathcal{N}$ to execute the query \mathbf{q} with the lowest possible global loss.

The loss function of the DML model f , denoted by $\mathcal{L}(f(\mathbf{x}; \theta), y) = \mathcal{L}(\hat{y}, y)$, measures the difference between the predicted output $\hat{y} = f(\mathbf{x}; \theta)$ and the actual output y given an input \mathbf{x} . This loss function is convex. For regression models, it can be $\mathcal{L}(\hat{y}, y) = |y - \hat{y}|$ or $(y - \hat{y})^2$. For binary classification models, it can be $\mathcal{L}(\hat{y}, y) = \max\{0, (1 - y\hat{y})\}$, with $y \in \{-1, +1\}$. Each node $n_i \in \mathcal{N}$ can have different data, which impacts the model f differently (Lin et al., 2022). For example, some nodes $\mathcal{N}' \subset \mathcal{N}$ could improve the model f 's performance, while others might negatively affect it.

We consider a discrete time domain $t \in \mathbb{T}$. At each instance $t = 1, 2, \dots$, we are given a DPA query \mathbf{q}_t . A query \mathbf{q}_t defines the boundaries \mathcal{B} of the input subspace \mathcal{X} , such that the samples contained therein are requested for training a DML model $f(\mathbf{x}; \theta(\mathbf{q}))$ given query \mathbf{q} . The parameters $\theta(\mathbf{q})$ of this model depend explicitly on the query \mathbf{q} , meaning this model should *ideally* be built from the distributed data:

$$D(\mathbf{q}) = \{(\mathbf{x}, y) \in D : \mathbf{x} \in \mathcal{B}\}. \quad (1)$$

The data space \mathcal{B} is a d -dimensional space embedded in \mathcal{X} :

$$\mathcal{B} = \{\mathbf{x} \in \mathcal{X} : \wedge_{i=1}^d q_i^{\min} \leq x_i \leq q_i^{\max}\}, \quad (2)$$

where the query \mathbf{q} specifies the min q_i^{\min} and max q_i^{\max} boundaries for each dimension in \mathcal{X} . In this paper, we adopt a $2d$ -dimensional range-based vectorial representation of a hyper-rectangle DPA query:

$$\mathbf{q} = [q_1^{\min}, q_1^{\max}, \dots, q_d^{\min}, q_d^{\max}]. \quad (3)$$

Note that this does not affect the analysis and generalization of our approach to define the distributed dataset $\mathcal{D}(\mathbf{q}) \equiv \bigcup_{i=1}^N \mathcal{D}_i(\mathbf{q})$. A query requested by a DPA application represented as a $2d$ -dimensional vector results in selections of data samples across data dimensions. Range queries are integral part of real world DPA applications (Savva et al., 2020c), e.g., calculating average pollution level across regions for city planning, selecting data from geo-spatial databases with GPS signals mining duration of individuals staying in locations, crime index indicators in city regions, e.g., neuroDB (Zeighami et al., 2022). In this work, we deal with multidimensional data vectors representing multivariate contextual data for DPA applications (see our example in Section 1.1).

The lowest and highest boundary values $\{q_j^{\min}, q_j^{\max}\}_{j=1}^d$ are drawn from an unknown underlying joint probability distribution $P(q_1^{\min}, q_1^{\max}, \dots, q_d^{\min}, q_d^{\max}) = P(\mathbf{q})$. The geometric representation of $\mathcal{D}(\mathbf{q})$ is a hyper-rectangle defined by the lowest and highest boundary values linked to an interval range query. Other query types could be adopted to represent a query \mathbf{q} , which do not spoil our data retrieval from \mathcal{D} , e.g., k -nearest neighbors queries (Ramswamy et al., 2000; Angiulli et al., 2006; Hajizadeh et al., 2022) or radius query (Anagnostopoulos and Triantafillou, 2017b; Savva et al., 2020c,b). The latter refers to as: $\mathcal{D}(\mathbf{q}) = \{\mathbf{x} \in \mathcal{D} : \|\mathbf{q}' - \mathbf{x}\|_p \leq \psi\}$ under norm $L_p = \|\mathbf{x} - \mathbf{x}'\|_p = (\sum_{i=1}^d |x_i - x'_i|^p)^{1/p}$. The geometrical representation of a radius query is a hyper-sphere in d -dimensional space defined by center $\mathbf{q}' = [q'_i]_{i=1}^d$ with $q'_i = \frac{1}{2}(q_i^{\max} + q_i^{\min})$ and radius $\psi \in \mathbb{R}_+$. If $p = \infty$, the query represents a hyper-rectangle centered at \mathbf{q}' with extent 2ψ on each dimension; for $p = 2$, the query region is a hyper-sphere.

We focus on query-based data selection for DPA represented by ranges over data dimension, e.g., ranges in date and time (temporal window of interest), city regions, vehicle' parking occupancy duration, vehicle detection indicators, and sensor types (e.g., magnetic, inductive). Query ranges can also include image features like texture, color, and shape represented as d -dimensional intervals for image clustering and classification tasks, e.g., Phamtoan and Vovan (2021). Selection over other types of data like sub-graphs and moving images is beyond of the scope of this work and is left our future work.

Finally, it is worth noting that in a DML environment, the dataset $\mathcal{D}(\mathbf{q})$ is distributed across nodes and depends on the query \mathbf{q} . Given two different queries \mathbf{q}_1 and \mathbf{q}_2 , the required datasets $\mathcal{D}(\mathbf{q}_1)$ and $\mathcal{D}(\mathbf{q}_2)$ for building the models $f(\mathbf{x}; \theta(\mathbf{q}_1))$ and $f(\mathbf{x}; \theta(\mathbf{q}_2))$, respectively, can differ. Additionally, the nodes containing the corresponding data for the two queries can also be different and dependent on the queries. We summarize our notations in Table 10 and abbreviations in Table 11.

3.2. Problem formulation

Given a DPA query \mathbf{q} , we seek the subset of nodes $\mathcal{N}' \subset \mathcal{N}$ that can be participating in training a model f over their distributed data. However, nodes may have different data, where only the relevant ones are needed to be accessed for training f . Consider the following example, with a network of $n = 10$ nodes, each one having its own local datasets \mathcal{D}_i , and a query \mathbf{q} that needs to access relevant data. We have three main cases for handling this query:

(C1) Centralized Case: One could gather all the data $\mathcal{D} \equiv \bigcup_{i=1}^n \mathcal{D}_i$ from all nodes in a central server and train a global model f_G over \mathcal{D} for the query \mathbf{q} , as in Peng et al. (2022). We then obtain the average prediction error e_G corresponding to f_G . This means that we do have

access to all participating nodes' data, which violates our principles for query-centric DPA in terms of data privacy.

(C2) Individual Case: Each node n_i can individually and locally train its local model f_i over local dataset \mathcal{D}_i . In this case, we obtain the (local) average error e_i for each node $n_i \in \mathcal{N}$ given the query \mathbf{q} . Evidently, in this case, no data are shared. On a first thought, one might claim that since the central server possesses complete knowledge, its predictions would be *better* (in terms of error) compared to those of each node. This is not always the case. It depends on the local data densities of each node, which are unknown in advance given a query. As it is evidenced in Fig. 2, the performance of the global model f_G in terms of accuracy is better than some of the nodes' local models, i.e., from a sub-set of nodes $\mathcal{N}_0 \subset \mathcal{N}$, whose local errors $e_j > e_G$ for nodes $n_j \in \mathcal{N}_0$. On the other hand, there are some nodes, whose models perform better than the global one. These are the *most suitable* nodes $\mathcal{N}_1 \subset \mathcal{N}$ with local errors $e_k < e_G$, $n_k \in \mathcal{N}_1$. This means that there exists a subset of nodes whose models are more accurate than the global one. However, to identify this subset of nodes, one must first engage all the nodes *in advance* and train local models for each of them. Additionally, a global model would need to be built by transferring all the data and then selecting those nodes in \mathcal{N}_1 . Evidently, this approach is neither practical nor applicable since we need to identify those nodes in advance given a query without training a global model.

Theorem 1. Let e_G and e_i denote the prediction error of central node/server and a local node $n_i \in \mathcal{N}$. It is not always the case that $e_G < e_i, \forall n_i \in \mathcal{N}$.

Proof. We prove Theorem 1 with two counterexamples. Consider the simple mean regression algorithm f^+ , i.e., given an input \mathbf{x} , its predicted output \hat{y} is the sample mean \bar{y} , and the k -nearest neighbors regression model f^* , i.e., the output is the average output of the k th closest input points to the input \mathbf{x} . Consider that each \mathcal{D}_i follows a Gaussian $\mathcal{N}(\mu_i, \sigma_i^2)$, with $\sigma_i^2 \rightarrow 0$ and $|\mu_i - \mu_j| \gg 0, i \neq j$. Evidently, the central node's data set $\mathcal{D} = \bigcup_{i=1}^n \mathcal{D}_i$ follows the mixture $\mathcal{N}(\mu, \sigma^2)$ with $\mu = \sum_{i=1}^n a_i \mu_i$, $\sigma^2 = \sum_{i=1}^n a_i ((\mu_i - \mu)^2 + \sigma_i^2)$, with $a_i > 0, \sum_{i=1}^n a_i = 1$. If we were told that all inputs followed $\mathcal{N}(\mu_j, \sigma_j^2)$ for some $j, 1 \leq j \leq n$ then we should have engaged only node n_j , thus, yielding (i) $e_j < e_G$ in case of f^+ and (ii) $e_j = e_G$ in case of f^* , and avoiding engaging all nodes n_i or, even, constructing the global model in the central node by transferring all the data. Furthermore, consider that all \mathcal{D}_i follow exactly the same distribution; consequently, the central node's data follows the same distribution. Then, regardless of any knowledge on the densities of inputs, we could randomly select one node from \mathcal{N} , thus, yielding $e_i = e_G, \forall n_i \in \mathcal{N}$, and avoiding constructing the central node and/or engaging all nodes. \square

(C3) Model Aggregation Case: The query \mathbf{q} can be sent to all nodes in parallel. Each node n_i locally builds a model f_i and sends it to a central node. The central node, after receiving all models, aggregates them and finally sends the aggregated model $f_A = \frac{1}{n} \sum_{i \in \mathcal{N}} f_i$ to all nodes, where each one stores it locally. The performance of f_A is expected to be higher than that of the global model f_G . Some of the nodes have similar or lower performance compared to the global model on the central node, while other nodes achieve much higher performance, as shown in Fig. 2. Moreover, an aggregated model across only those nodes in \mathcal{N}_1 can potentially have higher performance than f_A (since models from nodes in \mathcal{N}_0 have been excluded), as shown in Fig. 2 (black dash-dot line). Therefore, the models that belong to the nodes in \mathcal{N}_1 could minimize the expected error of the desired model, while the rest of the models do not contribute to improving the predictive performance. However, the nodes in \mathcal{N}_1 cannot be known in advance and are unlikely to be the same for each query.

For each incoming query, all nodes need to communicate to obtain an average estimate. This implies that all nodes are equally considered available for providing a model per query. Ideally, given a query \mathbf{q} , it would be preferable to identify the subset $\mathcal{N}' \subset \mathcal{N}$ of nodes whose

average model's estimate $\hat{y}' = \frac{1}{|\mathcal{N}'|} \sum_{n_i \in \mathcal{N}'} \hat{y}_i$ would be (almost) the same as \hat{y} . This estimate \hat{y} could be either the prediction of f_G or f_A . More interestingly, it would be ideal if we could select the *minimum* subset of nodes whose average estimate is as close to \hat{y} as possible for each query. Evidently, we desire to obtain the *minimum* subset $\mathcal{N}' \equiv \mathcal{N}_1$ in advance for each query, without engaging all the nodes and/or transferring their data to a central location. This is a NP-hard problem as explained below.

Theorem 2. Given a query q , the problem of finding the minimum subset $\mathcal{N}' \subset \mathcal{N}$ of nodes, whose average estimate \hat{y}' results to same error as \hat{y} is NP-hard.

Before proceeding with the proof of Theorem 2, let us recall the Subset Sum Problem (SSP): Consider a pair (I, s) , where $I = \{i_1, i_2, \dots, i_n\}$ is a set of $n > 0$ positive integers and s is a positive integer. SSP asks for a subset of I whose sum is as large as possible, but not larger than s . SSP is NP-complete. Consider now the following problem, referred to as Minimum Subset Average Problem (MSAP): Given (I, s) , find the minimum subset I' with average s' such that $\lfloor s' \rfloor = s$ or $\lceil s' \rceil = s$.

Theorem 3. The MSAP is NP-hard.

Proof. If there is a polynomial-time algorithm for MSAP, then a polynomial-time algorithm can be developed for SSP. Suppose that there exists a polynomial algorithm, $A(I, s)$ that solves MSAP, i.e., $A(I, s)$ finds in polynomial time the minimum subset I' s.t. s . Then $A(I, s)$ can be used to solve the SSP with (I, ns) , $n = |I|$. In general, any solution $B(I, s)$ of SSP with (I, s) can be formulated as the Algorithm 1. If the polynomial-time complexity of $A(I, s)$ is a polynomial $P(n)$, then the complexity of $B(I, s)$ is $O(nP(n))$. But, this implies that there is a polynomial-time algorithm for SSP. Hence, no polynomial-time algorithm exists for MSAP. \square

ALGORITHM 1: $B(I, s)$

Input: I, s

Output: I'

for $1 \leq k \leq |I|$ **do**
 call $A(I, \frac{s}{k})$;
 If a subset I' of I with k elements is found, whose elements have an average k' such that $\lfloor k' \rfloor = s/k$ or $\lceil k' \rceil = s/k$ **Then**
 obviously their sum is s , so exit and return I'
end

Based on the outcome of Theorem 3, we can now proceed with the proof of Theorem 2.

Proof. Let $e = |\hat{y} - y|$ and $e' = |\hat{y}' - y|$. In order to show that the problem of finding the minimum subset \mathcal{N}' with $e' = e$ is NP-hard, it suffices to show that the problem $|\hat{y}| = |\hat{y}'|$ is NP-hard. Consider the set $I^R = \{|\hat{y}_i|_{i=1}^n\}$, $|\hat{y}_i| > 0, \forall i$. Since MSAP, which deals with set of positive integers, is NP-hard from Theorem 3 then a variant of MSAP with (I^R, \hat{y}) is, also, NP-hard. \square

Nonetheless, if one is able to find the minimum set \mathcal{N}' of nodes for a given query (let n be very small), then, this is meaningless. That is because, in order to obtain \mathcal{N}' for a given query, once has *firstly* to query all nodes and, consequently based on their estimates, produce \mathcal{N}' . Hence, one has to *predict* in advance the most appropriate subset \mathcal{N}' , which results to same or less error than that of \mathcal{N} . Furthermore, out of those nodes in \mathcal{N}' , we should select *those* nodes whose datasets satisfy the query boundaries B . Therefore, we propose a mechanism for proactive selection of nodes for each query, where these nodes will be engaged in training the model $f(x; \theta(q))$.

At any instance t , our mechanism predicts a subset $\mathcal{N}'_t \subset \mathcal{N}$ whose selected nodes provide a good model for query q_t . Good model relates

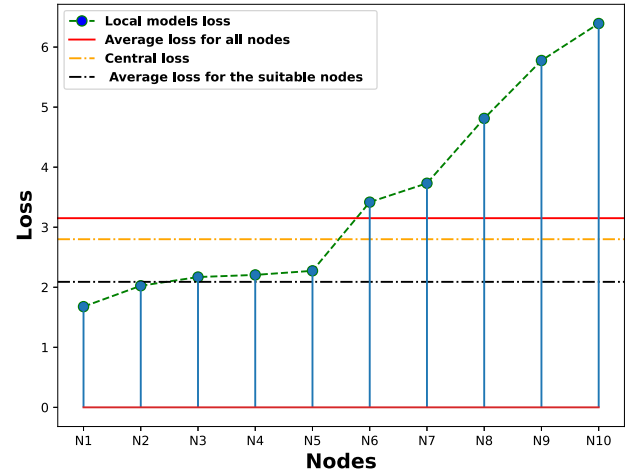


Fig. 2. Local models' prediction errors (loss) over ten nodes compared with the centralized model, the average model across all nodes, and the average model across only the most suitable nodes.

to (i) obtain a relatively small prediction error e_t as possible and (ii) achieve predictions \hat{y} close to \hat{y}_G (if the latter is able to be determined) denoting that $e_t = e_{G,t} + \epsilon$ for some tolerance $\epsilon > 0$ or, if possible, $e_t < e_{G,t}$.

Problem 1. Given a query q_t at time instance t , predict whether a node $n_i \in \mathcal{N}$ is suitable for contributing to train the model f requested by the query, i.e., if n_i is a candidate to be added to \mathcal{N}'_t , based on minimum sufficient information node n_i has a-priori conveyed.

In Problem 1, a suitable node selection decision should be achieved before model building given a DPA query. This entails predicting nodes' memberships to \mathcal{N}'_t for each query at time t . This in turn, requires sufficient statistical information about the nodes' datasets without disclosing or accessing the actual data. Our mechanism should exploit such information to determine whether a node's data (or even parts of data) are *relevant* to query boundaries B in question. This enables a node to be included in \mathcal{N}'_t , thus participating in the DML training process of model f .

4. Node & relevant data selection

4.1. Data relevance factors: Overview

We first introduce the factors that determine the notion of node's data relevance per query with a running example and then elaborate on the node selection mechanism based on ranking.

Consider a node n_i , which is randomly selected to participate in training model f given a query q . For illustration purposes only, Fig. 3 shows the node's data and the data region defined by the query (area in a rounded rectangle) projected onto a 2-dimensional plane (e.g., the axes could refer to the first two principal components of the data feature space).

The data samples enclosed in the region defined by the query are the *relevant* samples w.r.t. query boundaries B for training the model f . Using the entire node's data results in incorporating *irrelevant* samples for the query. Consequently, training the model f with *all* data leads to a drift from the most accurate model (Goetz et al., 2019). Our challenge is to eliminate the impact of irrelevant samples with respect to the query on the model's training stage, especially when we do not have actual access to the node's data. The identification of a node's relevant and irrelevant data *per* query, which determines the training samples for building f , is not trivial. Our mechanism should identify only the relevant data *per* selected node *per* query by acquiring the minimum

sufficient information from the data. Sufficient information can initially be conveyed by quantizing node's data. This can help separate relevant from irrelevant data per query.

We distinguish three factors determining the notion of data relevance w.r.t. query's boundaries over clustered data: **(F1)** The overlapping area between clusters' boundaries and the query's boundaries. **(F2)** The number of samples per cluster. **(F3)** The number of relevant samples per cluster. Each factor plays a specific role in assessing whether node n_i 's data are relevant for model training given a query.

In our example in Fig. 3, we assume that node n_i has $L_i = 6000$ samples clustered into six clusters $\{C_1, \dots, C_6\}$ using, for example, k -means clustering algorithm. The samples are distributed per cluster as $\{1000, 800, 900, 1200, 1100, 1000\}$, respectively. As shown in Fig. 3, the areas of clusters C_1 , C_2 , and C_3 have a higher overlap with the query's boundaries. These clusters, hereinafter referred to as *supportive clusters*, contain relatively many relevant samples w.r.t. the query. This contrasts with clusters C_4 , C_5 , and C_6 , which have either very small or zero overlap with the query's boundaries.

Considering factor **F1**, we determine the membership of node n_i based on the degree of overlap per cluster, *without* actually accessing the data. It is sufficient to obtain the minimum and maximum values of each dimension per cluster and compare these against the query's boundaries.

Considering factor **F2**, the number of samples per cluster can further support the decision on selecting node n_i . This information can be easily obtained from the clustering process. A supportive cluster with a few samples and relatively high overlap with the query's boundary could be mistakenly considered more relevant for model building than a supportive cluster with more samples and medium overlapping rate. For instance, C_1 contains more samples than the other two supportive clusters, C_2 and C_3 . Therefore, C_1 has the highest probability of providing more relevant samples for model building. This probability can be empirically calculated as the proportion of samples in each supportive cluster relative to the total number of samples across all supportive clusters. In our case, we obtain $(p_1, p_2, p_3) = (0.37, 0.29, 0.34)$ for C_1 , C_2 , and C_3 , respectively.

By considering both factors **F1** and **F2**, we can order the supportive clusters based on their probabilities, from the most supportive, C_1 , to the moderately supportive, C_3 , and finally to the least supportive, C_2 . However, as shown in Fig. 3, *all* the samples in C_2 are relevant w.r.t. the query. Therefore, C_2 should be ranked first, not C_1 . This indicates that the information from factors **F1** and **F2** alone is insufficient for accurately determining node n_i 's participation.

Consequently, we introduce factor **F3**, which focuses only on the samples from a supportive cluster that fall within the query's boundaries (rather than considering all samples from the cluster). A cluster's high density does not necessarily mean its samples are relevant to the query. **F3** aims to exclude irrelevant samples from supportive clusters. In our example, 25.5%, 87.5% and 22.2% of the samples in supportive clusters C_1 , C_2 , and C_3 , respectively, fall within the query's boundary and are thus relevant. These percentages help revise the ranking probabilities based on the proportion of relevant samples. Consequently, the updated probabilities are $(p_1, p_2, p_3) = (0.22, 0.60, 0.18)$, indicating that C_2 is now the most supportive and relevant cluster, while C_3 is the least relevant. These factors lead us to a more accurate ranking of supportive clusters for node n_i , and hence a more precise node selection based on query-data relevance. Importantly, the determination of relevant samples relies on all factors work without disclosing nodes' data.

4.2. Data relevance based on factor F1

Consider a query q which defines the boundary B over the data, and a node n_i with a local dataset D_i . The overlap between the query's boundary and node's data indicates an estimation of the percentage of data from the entire dataset D_i required for accessing and training

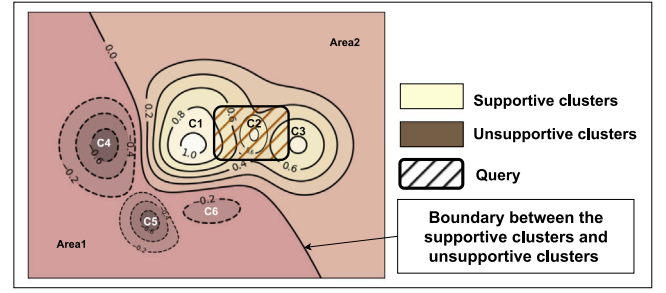


Fig. 3. An illustration example of the impact of supportive clusters per query on predicting the relevant data on a node (data are projected onto a 2-dimensional space for illustration purposes only).

the model f . This is associated with node n_i 's contribution to model training.

As discussed in Section 4.1, **F1** assesses the participation of a node and the corresponding relevance of its data from the supportive clusters. Specifically, node n_i has first quantized its input data space into K clusters $\cup_{k=1}^K \{C_k\} \equiv D_i$, associated with d -dimensional cluster heads $\{c_k \in \mathcal{X} \subset \mathbb{R}^d\}_{k=1}^K$, which minimizes the quantization error:

$$\min_{\{c_1, \dots, c_K\}} \sum_{k=1}^K \sum_{(x,y) \in D_i} \|x - c_k\|^2. \quad (4)$$

Our goal in this step is to find the number of clusters K' out of K that have a high overlap with the query boundary across all dimensions. A high overlap with many clusters K' in node n_i indicates that this node has data patterns similar to those requested by the query, which is expected to increase the chance of that node being selected as a participant in the learning process.

Let us define the $2d$ -dimensional vector:

$$c'_k = [x_{1,k}^{\min}, x_{1,k}^{\max}, \dots, x_{d,k}^{\min}, x_{d,k}^{\max}]^T, \quad (5)$$

which contains the minimum and maximum values for each dimension of all the data inputs $x \in C_k$ belonging to the cluster C_k , $k = 1, \dots, K$. Specifically, $x_{j,k}^{\min} = \min\{x_{j,k} : x \in C_k\}$ and $x_{j,k}^{\max} = \max\{x_{j,k} : x \in C_k\}$, $j = 1, \dots, d$. The vector c'_k represents the (hyper) rectangle boundary of the cluster C_k .

It might be the case that the underlying distribution of the data can change due to concept drift in, e.g., the input domain. This leads to potential changes in some of the cluster heads c_k of a node n_i , and in turn, updates the corresponding boundary vectors c'_k . We then rely on the principles of incremental learning of the current cluster heads upon receiving a new batch of input vectors $\{x^{(r)}, x^{(r+1)}, \dots\}$ by adopting the following update rule upon an incoming $x^{(r)}$:

$$c_k^{(r+1)} = \begin{cases} c_k^{(r)} + \xi(x^{(r)} - c_k^{(r)}) & k = \arg \min_k \|x^{(r)} - c_k^{(r)}\|^2, \\ c_k^{(r)} & \text{otherwise.} \end{cases} \quad (6)$$

This means, if node n_i receives a data $x^{(r)}$ at time instance r , then the closest cluster-head $c_k^{(r)}$ is updated by update rate $\xi \in (0, 1)$ towards the new data $x^{(r)}$.

Given an arrival of \mathcal{T} new data points $\{x^{(r)}, x^{(r+1)}, \dots, x^{(T)}\}$, for those cluster-heads which have been updated using (6), the corresponding boundary vectors c'_k are updated based on the membership of the newly incoming data points to their cluster-heads. Given the query q and the vectorized k th cluster boundary c'_k , we calculate the percentage of overlap $h_k(q) \in [0, 1]$ between these two hyper-rectangles (across all dimensions):

$$h_k(q) = \frac{1}{d} \sum_{j=1}^d h_{j,k}(q), \quad (7)$$

where

$$h_{j,k}(\mathbf{q}) = \begin{cases} \frac{q_j^{\max} - x_j^{\min}}{\max(x_j^{\max}, q_j^{\max}) - \min(x_j^{\min}, q_j^{\min})} & \text{if } q_j^{\max} \leq x_j^{\max}, \\ \frac{x_j^{\max} - q_j^{\min}}{\max(x_j^{\max}, q_j^{\max}) - \min(x_j^{\min}, q_j^{\min})} & \text{if } x_j^{\max} \leq q_j^{\max}, \\ 0 & \text{if } q_j^{\max} \leq x_j^{\min}, \\ & \text{or } x_j^{\max} \leq q_j^{\min}. \end{cases} \quad (8)$$

A cluster C_k is considered supportive w.r.t. query \mathbf{q} if and only if $h_k(\mathbf{q}) \geq \epsilon$ based on a DPA application specific overlapping threshold $\epsilon > 0$. Hence, for node n_i , we obtain $K' \leq K$ supportive clusters:

$$C' \equiv \cup \{C_k : h_k(\mathbf{q}) \geq \epsilon\}. \quad (9)$$

The overall support of the supportive clusters towards the query \mathbf{q} is then defined as:

$$r_i(\mathbf{q}) = \frac{1}{K'} \sum_{k=1}^{K'} h_k(\mathbf{q}). \quad (10)$$

So far, we argue that the samples belonging to the supportive clusters are considered relevant for the query \mathbf{q} , i.e., should node n_i be selected in \mathcal{N}' , then the samples in:

$$D'_i = \{(\mathbf{x}, y) \in D_i : \mathbf{x} \in \cup_{k=1}^{K'} C_k\}, \quad (11)$$

can be used for model training.

4.3. Data relevance based on factor F2

The overlapping degree $h_k(\mathbf{q})$ is not enough to decide whether node n_i is suitable to participate in model training, as discussed in Section 4.1 in terms of F2. For instance, assume two nodes n_1 and n_2 both have $K_1 = K_2 = 3$ clusters, with node n_1 having $K'_1 = 1$ supportive cluster and node n_2 having $K'_2 = 2$ supportive clusters with respect to ϵ . We could rank the nodes based on their percentage of supportive clusters. For example, we obtain $\left(\frac{K'_1}{K_1}, \frac{K'_2}{K_2}\right) = \left(\frac{1}{3}, \frac{2}{3}\right)$ for n_1 and n_2 , respectively. Hence, n_2 can be considered more suitable than n_1 . However, the number of samples may differ across clusters. If the supportive cluster of node n_1 had more samples than the total number of samples in the two supportive clusters of node n_2 , then n_1 should have been ranked first and not n_2 . This will not happen unless the number of samples per supportive clusters has been taken into consideration.

The number of training samples is important to obtain a good model fit. Especially, in DPA models, by decreasing the sample size yields a model unable to capture the relationship between the input and output accurately, thus, leading to lower predictive performance as evidenced by Wu et al. (2023). Let $|C_k|$, $k = 1, \dots, K'$ be the number of samples of a supportive cluster $C_k \in C'$ of node n_i . Then, we extend (10) by adding the size of the supportive clusters:

$$r'_i(\mathbf{q}) = \sum_{k=1}^{K'} h_k(\mathbf{q}) a_k(\mathbf{q}), \text{ with } a_k(\mathbf{q}) = \frac{|C_k|}{\sum_{k=1}^{K'} |C_k|}, \quad (12)$$

such that $\sum_k a_k(\mathbf{q}) = 1$.

So far, given factors F1 and F2, we have obtained the overlapping degree $h_i(\mathbf{q})$ and the sizes of supportive clusters for node n_i with respect to a query \mathbf{q} . These indicators will be used to rank the suitability of n_i for model building, by using only the selected samples in the dataset $D'_i \subseteq D_i$, as derived in (11).

At that stage, the minimum sufficient statistic, which can be provided by each node, is the cluster boundary vectors $\{\mathbf{c}'_k\}$ associated with the clusters C_k . This can be shared among node n_i 's neighborhood \mathcal{N}'_i in advance. Based on this, a node from a neighborhood, which receives a query \mathbf{q} , plays the role of a 'leader' to locally determine the suitability of each neighbor node without accessing the neighbors' data. Specifically, each neighboring node shares its cluster boundary vectors in advance,

so there is no need for further communication with the leader node for each query. The leader node can efficiently compute the metric $h_j(\mathbf{q})$ for each neighboring node n_j upon receiving a query \mathbf{q} . This computation can be efficiently performed locally in $O(Kd)$ time.

4.4. Data relevance based on factor F3

The samples in D'_i for a node n_i are determined based on the boundaries of its supportive clusters, which have been identified w.r.t. the overlap between the query and cluster boundaries. However, according to factor F3 in Section 4.1, not all the samples in a supportive cluster can be considered relevant to the query boundaries. The fact that a supportive cluster overlaps with the query boundaries does not imply that all its samples are relevant. Relevance depends on the local density and distribution of samples within the supportive cluster itself.

To gain further insights into the relevance of the samples in supportive clusters, each node can include in its sufficient statistic the ratio of samples in supportive clusters that also falls within the query boundary. This means that each neighboring node must locally examine whether each input sample \mathbf{x} from (\mathbf{x}, y) in D'_i satisfies the query boundary B as well. In this case, when the leader node receives a query \mathbf{q} , it requests each neighbor to identify the subset of their dataset that is relevant to the query:

$$D''_i = \{(\mathbf{x}, y) \in D'_i : \mathbf{x} \in B\}. \quad (13)$$

That is, each node locally can estimate the size of relevant samples out of all the samples in a supportive cluster $|C'_k|$ such that $C'_k = \{(\mathbf{x}, y) \in C_k : \mathbf{x} \in B\}$. Hence, the node can return to the leader node the revised support information:

$$r''_i(\mathbf{q}) = \sum_{k=1}^{K'} h_k(\mathbf{q}) b_k(\mathbf{q}), \text{ with } b_k(\mathbf{q}) = \frac{|C'_k|/|C_k|}{\sum_{k=1}^{K'} |C'_k|/|C_k|}, \quad (14)$$

such that $\sum_k b_k(\mathbf{q}) = 1$.

In this case, the sufficient statistic requires $O(\sum_{k=1}^{K'} |C_k|d)$ time to be calculated on each member node for the query \mathbf{q} , which needs to be sent to the leader.

4.5. Ranking of suitable nodes

Based on factors F1 and F2, only the leader node locally estimates the metric $h_i(\mathbf{q})$ in (12) for each member n_i given a query. This requires that in advance each member has sent over the cluster vector boundaries to the leader once. This comes with $O(1)$ communication in the neighborhood of the leader and only local processing on the leader once, independent on the number of the incoming queries $\{\mathbf{q}_1, \dots, \mathbf{q}_T\}$ up to horizon T . By incorporating factor F3, the leader node requires each member to estimate the metric in (14) locally per query. Then, each member sends this over to the leader. This requires $O(T)$ communication and local processing on each member.

Taking into account all factors, we rank the suitability of a node n_i for a query \mathbf{q} . Nodes are sorted by $\{r'_1, \dots, r'_N\}$ by the leader if only the support in (12) is used, resulting in no further communication with the leader. Alternatively, if the support in (14) is used, the leader, in collaboration with its neighbors, sorts the nodes by $\{r''_1, \dots, r''_N\}$. In both cases, the leader determines a subset of the top- ℓ nodes ($\ell < N$) to be engaged in model learning given a query. The number ℓ of top-ranked nodes can be decided by selecting those nodes whose support r' (or r'') satisfies a specific condition $\mathcal{A}(r')$:

$$\mathcal{N}' = \{n_i \in \mathcal{N} : \mathcal{A}(r'_i) = \text{TRUE}\}, \quad (15)$$

with $\ell = |\mathcal{N}'|$. The condition depends on the DML policy adopted, which will be elaborated in Section 5. A node $n_i \in \mathcal{N}'$ participates in the distributed learning using its own *relevant* samples $D''_i \subseteq D'_i \subset D_i$, based on the adopted support metric r'_i or r''_i . Therefore, the relevant

dataset $D(q)$ for training the model f refers to the distributed relevant datasets of the selected nodes in \mathcal{N}' , i.e., $D(q) \equiv \cup_{n_i \in \mathcal{N}'} D_i''$.

The query-driven distributed learning minimization objective seeks the distributed model f over *only* the relevant data $D(q)$ across the *selected* nodes in \mathcal{N}' that collaboratively minimize the loss:

$$\begin{aligned} \mathcal{J}(f; q) &= \frac{1}{|D(q)|} \sum_{(x,y) \in D(q)} \mathcal{L}(f(x; \theta(q)), y) \\ &= \frac{1}{|D(q)|} \sum_{n_i \in \mathcal{N}'} \sum_{(x,y) \in D_i''} \mathcal{L}(f(x; \theta(q)), y). \end{aligned} \quad (16)$$

Note that in our case, minimizing (16) specializes the generic distributed machine learning objective in Verbraeken et al. (2020). Our aim is to minimize (16) among only the selected nodes in \mathcal{N}' as expressed in Problem 1. Hence, in our case, the distributed model f is trained over the predicted relevant samples in $D(q)$ as requested by the DPA q . In turn, the derived model f is tailored to the query q specifications and should not be confused with a derived model which would be trained over *all* nodes' data (relevant and irrelevant) regardless of any DPA query. In our case, each selected node $n_i \in \mathcal{N}'$ participating in the minimization of (16) contributes to the model training with only its (potential) relevant data in D_i'' and not with all its available data D_i . Section 6.3.5 elaborates on the impact of this data selection decision on the generalization capacity of the tailored model f and the corresponding implications.

In the next section, we provide the DML mechanisms to collaboratively train the model f tailored to a DPA query q across the selected nodes over their relevant data in a distributed fashion to minimize (16). Such DML mechanisms span across the spectrum of distributed learning (e.g., federated learning) as elaborated below.

5. Query-centric DML mechanisms

Given the selected nodes \mathcal{N}' for a query q , we elaborate on the DML mechanisms based on the interactions among the selected participants. A node is elected as the leader based on specific criteria. In this work, a node becomes the leader if it receives a query q . Consequently, the remaining nodes are considered members for this query. Note that any member node for a query q can be elected as a leader for a different query q' if it receives that query (see example in Section 1.1). The same holds true for an already leader, which can be appointed as a member w.r.t. another query received by another node.

Let us focus on a specific query q received to a leader node n_0 . The leader node n_0 can initiate a (minimum) spanning tree to make aware all its members about its appointment for the received query q .

5.1. Best node model learning

According to node ranking, the leader n_0 selects only the top-ranked node n_i using the Best Node (BN) learning policy. This selection is made either directly by calculating the supports $\{r_j''\}$ or by collaborating with the member nodes to calculate the supports $\{r_j''\}$, as discussed in Section 4. For simplicity, we refer to both variants of supports, r_j'' and r_j' , unless otherwise specified. The selection condition $\mathcal{A}(r_i'')$ for the top-ranked node n_i is satisfied if and only if $r_i'' = \max\{r_j'' : n_j \in \mathcal{N}'\}$. In BN (see Fig. 5(a)), it is assumed that node n_i has the most relevant sample among all candidate nodes. Therefore, the leader assigns node n_i to build a model $f_i(x; \theta_i(q))$ locally, using n_i 's relevant data D_i'' .

The model f_i is sent to the leader node n_0 as requested, resulting in $O(1)$ communication with the BN policy. There might be a case where the top-ranked node has nearly all of the relevant data with respect to the query, i.e., $r_i'' \rightarrow 1$. In this case, we achieve results similar to those in centralized training with respect to the query, where $D_i'' \simeq D(q)$. This similarity is also reflected by the loss function of the BN, i.e., we obtain that:

$$\mathcal{J}(f; q) = \mathcal{J}(f_i; q) = \frac{1}{|D_i''|} \sum_{(x,y) \in D_i''} \mathcal{L}(f_i(x; \theta_i(q)), y). \quad (17)$$

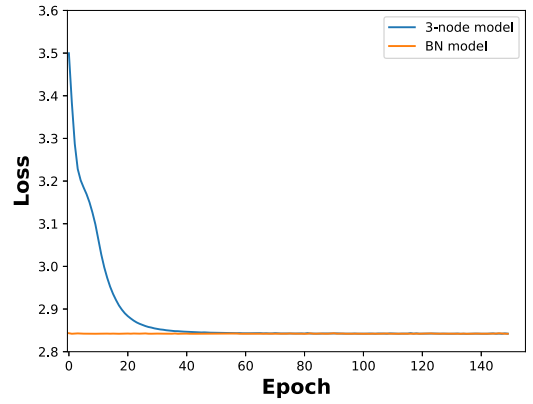


Fig. 4. Comparison of BN model (top-1 node) vs. incremental model engaging the top-3 nodes.

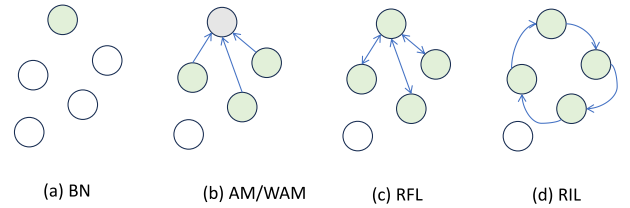


Fig. 5. The spectrum of the query-centric DML mechanisms from (a) Best Node (BN), to (b) Aggregation (AM/WAM), (c) Ranking-based Federated Learning (RFL), and (d) Ring-based Incremental Learning (RIL).

Further (incremental) training of f_i with additional candidate nodes (e.g., the 2nd or 3rd ranked nodes) could lead to overfitting, potentially worsening the model's performance. To avoid this problem, particularly when the top-ranked node has a relatively high support r_i'' , we should be cautious about including *unnecessary* additional candidate nodes. For illustration, Fig. 4 shows the case of engaging more than just the top-ranked node with $r_i'' = 0.9$ for training a deep neural network given a query. It demonstrates that there is no significant improvement in performance after incrementally training the model across the top-3 candidate nodes.

5.2. Aggregate/weighted aggregate model learning

Let us consider a case where more than one node has a relatively high support (ranking) value in the candidate set. In this case, more than one node can participate in DML by providing access to its relevant data. Given the supports r_i'' , the leader node n_0 selects nodes under a cut-off condition over r_i'' . This cut-off is a separating statistic between low and high ranking values, derived from the principle of outliers detection using the Median Absolute Deviation (MAD). MAD is defined as the median of the absolute deviations from the median $\bar{r} = \text{median}\{r_i''\}$, i.e., $\text{MAD} = \text{median}\{|r_i'' - \bar{r}|\}$. MAD is the most robust measure of dispersion, separating relatively high support values under a cut-off, which is normally set to 2.5 times the MAD, as suggested in Yang et al. (2019).

In the Aggregate Model (AM) learning, the selection condition $\mathcal{A}(r_i'')$ for a node n_i to be selected is:

$$\mathcal{N}' \equiv \{n_i \in \mathcal{N} : \frac{|r_i'' - \bar{r}|}{\text{MAD}} > 2.5\}. \quad (18)$$

The model f derives from aggregating the locally trained models f_i from the selected nodes $n_i \in \mathcal{N}'$ determined in (18). The AM mechanism essentially takes advantage of the local data variety over the selected nodes (see Fig. 5(b)). The leader n_0 communicates once with the selected nodes enabling them to locally build their models $\{f_i\}$

over their relevant data $\{D_i''\}$, individually. Then, the selected nodes returns their local models (parameters) $\{\theta_i\}$ to the leader. The latter aggregates the local models with equal importance:

$$f(\mathbf{x}; \theta(\mathbf{q})) = \frac{1}{|\mathcal{N}'|} \sum_{n_i \in \mathcal{N}'} f_i(\mathbf{x}; \theta_i(\mathbf{q})). \quad (19)$$

Note that we obtain $O(|\mathcal{N}'|)$ communication with the AM policy. However, selected nodes could have different data distributions and ranges. Hence, some local models can perform better or worse than other, thus, it is not well argued to aggregate the local models equally. The model f is derived by a weighted aggregation of the local models taking into account the individual contribution of each models based on the relative ranking (support) given the query. We then obtain the Weighted Aggregate Model (WAM):

$$f(\mathbf{x}; \theta(\mathbf{q})) = \sum_{n_i \in \mathcal{N}'} \hat{r}_i f_i(\mathbf{x}; \theta_i(\mathbf{q})). \quad (20)$$

where $\hat{r}_i = \frac{r_i''}{\sum_{n_k \in \mathcal{N}'} r_k''} \in [0, 1]$ and $\sum_{n_i \in \mathcal{N}'} r_i'' = 1$. The loss function of WAM is then:

$$\mathcal{J}(f; \mathbf{q}) = \sum_{n_i \in \mathcal{N}'} \frac{\hat{r}_i}{|D_i''|} \sum_{(\mathbf{x}, y) \in D_i''} \mathcal{L}(f_i(\mathbf{x}; \theta_i(\mathbf{q})), y). \quad (21)$$

We obtain the loss of AM by setting $\hat{r}_i = 1, \forall i$ in (21).

5.3. Ranking-based federated learning

Under the data relevance notion, each selected node contributes to minimizing the expected loss of the model f by adopting the principles of Federated Learning (FL) introduced in McMahan et al. (2016). We extend the FL paradigm in our context by incorporating the weight of the relative ranking of the nodes selected in the AM/WAM mechanism, participating in a holistic (*federated*) model optimization as follows.

Firstly, the leader node n_0 selects the most suitable nodes based on (18). Then, the leader initiates a FL-based optimization process for \mathcal{T} rounds across the selected nodes in \mathcal{N}' . We modify the sampling of the selected nodes at each round of the FL optimization by leveraging the relative ranking of the selected nodes. Specifically, at each round $\tau = 1, \dots, \mathcal{T}$, the leader selects κ nodes out of the ℓ selected nodes in \mathcal{N}' based on the ranking probabilities $(\hat{r}_1, \dots, \hat{r}_\ell)$. Within round τ , each sampled node locally trains its model $f_i^{(\tau)}(\mathbf{x}; \theta_i^{(\tau)}(\mathbf{q}))$ over its relevant samples D_i'' .

At the end of the round τ , the model parameters $\{\theta_i^{(\tau)}(\mathbf{q})\}_{i=1}^\kappa$ of the κ sampled nodes are sent to the leader node. The leader aggregates the received models using the relative ranking as weights. This results in giving more attention to models trained with sampled from nodes with relatively more relevant data than the rest of the nodes (see Fig. 5(c)). The overall objective of the Ranking-based FL (RFL) over the relevant data $\{D_i''\}$, $n_i \in \mathcal{N}'$ involving the ranking-based weighted aggregation by the leader is given by:

$$\min_{\theta(\mathbf{q})} \sum_{n_i \in \mathcal{N}'} \sum_{(\mathbf{x}, y) \in D_i''} \frac{\hat{r}_i}{|D_i''|} \mathcal{L}(f(\mathbf{x}; \theta(\mathbf{q})), y). \quad (22)$$

The federated model f obtained by (22) at the end of round \mathcal{T} is used by leader node n_0 to serve the query \mathbf{q} .

In RFL, we require $O(\mathcal{T}\ell)$ total communication rounds with the selected nodes per query. This derived from the principles of FL, which gradually optimizes the f 's predictive performance w.r.t. our objective in (22).

5.3.1. WAM & RFL loss functions

In AM and WAM learning policies, a selected node n_i locally optimizes its local model parameter $\theta_i(\mathbf{q})$, $\forall i$, over the query-defined relevant samples such that:

$$\theta_i^*(\mathbf{q}) = \arg \min_{\theta_i(\mathbf{q})} \sum_{(\mathbf{x}, y) \in D_i''} \mathcal{L}(f_i(\mathbf{x}; \theta_i(\mathbf{q})), y). \quad (23)$$

Then, each optimized model is aggregated through (19) or (20) for AM or WAM, respectively, in the leader node n_0 in order to serve the incoming query \mathbf{q} . Hence, no extra communication among selected nodes is required. On the contrary, in RFL, all selected nodes try to optimize a FL model f , i.e., seek the optimal federated parameter $\theta(\mathbf{q})$ across all the relevant samples as provided in (22). The prediction capacity of each locally optimized model aggregated in the leader node per round is weighted based on the relative ranking of the participating nodes.

5.3.2. FL & RFL loss functions

The weighting model factor in (22) depends on the selected node ranking and the associated number of relevant samples within the node based on the query. This departs from the conventional FL paradigm by McMahan et al. (2016), where the weighting factor refers to all the samples residing on a node (including relevant and irrelevant samples). Moreover, the ranking value attempts to give more importance to the model parameters of those selected nodes with higher relevance to the query during the FL training process. Our proposed query-centric model averaging scheme in (22) is aligned with the averaging convergence scheme in Li and Song (2022), such that our averaging weights $\left\{ \frac{\hat{r}_i}{|D_i''|} \right\}$ are invariant in training time. Note that the ranking probabilities $(\hat{r}_1, \dots, \hat{r}_\ell)$ relate to random sampling without replacement such that κ selected nodes are sampled out of the ℓ suitable nodes in \mathcal{N}' .

5.4. Ring-based Incremental Learning

In the Ring-based Incremental Learning (RIL) mechanism, given a query \mathbf{q} , the leader node n_0 selects the most suitable nodes as in (18) and then logically arranges these nodes to have a kind of dependency among them to incrementally train the model f over their datasets.

First, the nodes form a logical clockwise ring topology (initiated by the leader). The nodes are linked from the one with the relatively highest support to the node with the relatively least support following the decreasing order $r_1'' \rightarrow r_2'' \rightarrow \dots \rightarrow r_\ell''$, $\ell = |\mathcal{N}'|$ (see Fig. 5(d)). A node $n_i \in \mathcal{N}'$ receives the model update from the previous one n_{i-1} to further train and passes this model to the next node n_{i+1} .

In the incremental learning literature, the benefits of incremental learning are as follows. (i) By increasing the number of samples in model training, the accuracy and generalization ability of the model will be greatly improved. In our context, the model f is incrementally trained over the relevant samples from the suitable node in the ring. (ii) Based on the current model $f^{(m)}$ at the stage/node n_m , the performance of the model will be further optimized by new training samples from the next nodes $n_{\kappa}, \kappa > m$. (iii) After a stable model is obtained, the new training samples will not affect the performance of the model. This means that the predictions of new samples by two successive latest models $f^{(m)}$ and $f^{(m-1)}$ are almost the same.

Based on the above-mentioned benefits, the loss of the ring-based incremental learning of $f^{(m)}$ at node $n_m \in \mathcal{N}'$ is:

$$\begin{aligned} \mathcal{J}(f^{(m)}; \mathbf{q}) = & \sum_{(\mathbf{x}, y) \in D_{(m)}''} \mathcal{L}(f^{(m)}(\mathbf{x}; \theta_{(m)}(\mathbf{q}), y)) \\ & + \xi_{(m)} \sum_{(\mathbf{x}, y) \in D_{(m)}''} \left(f^{(m)}(\mathbf{x}; \theta_{(m)}(\mathbf{q}), y) \right. \\ & \left. - f^{(m-1)}(\mathbf{x}; \theta_{(m-1)}(\mathbf{q}), y) \right)^2. \end{aligned} \quad (24)$$

$f^{(m-1)}$ denotes the model at stage $m-1$. ξ is a weighting factor representing the importance between the two predictions of two consecutive variants of the model during incremental learning. As the model is incrementally trained by accessing relevant samples from the suitable nodes, the factor $\xi_{(m)}$ increases. This indicates that the derived (final) model should be more generalizable over the relevant spaces of the models with higher support values.

Table 1
Summary of Complexities.

Node Selection		
Mechanism	Computation	Communication
Factor F1	$O(dK)$	N/A
Factor F2	$O(dK)$	$O(\ell)$
Factor F3	$O(d(K + C'))$	$O(\ell)$
DML Mechanism		
	Computation	Communication
BN	$O(1)$ (leader)	$O(1)$
A/WAM	$O(\ell)$ (leader)	$O(\ell)$
RFL	$O(\ell)$ (leader)	$O(\mathcal{T}\ell)$
RIL	$O(\ell)$ (leader)	$O(v\ell)$

5.4.1. Passes & ring structure in RIL

The model f can be (re)trained incrementally over more than one pass $v \geq 1$ of the ring. That is, the final model at the end of the first round, i.e., the model $f^{(\ell)}$ reaching the node with the highest support $r_1'' = \max\{r_i''\}_{i=1}^{\ell}$, can undergo another pass of the ring.

We provide a process where the leader node n_0 can create the logical ring of the selected nodes in \mathcal{N}' . Based on the ranking list indexed by the selected nodes, i.e., $\Lambda = \{(r_1'', n_1), \dots, (r_{\ell}'', n_{\ell})\}$, the leader sends Λ to the top-ranked node n_1 . Node n_1 is appointed as the *head* of the ring. It receives the list and is advised to connect with node n_2 , sending the sublist $\Lambda \setminus \{(r_1'', n_1)\}$. Node n_2 then sends the updated sublist $\Lambda \setminus \{(r_1'', n_1), (r_2'', n_2)\}$ to node n_3 , and this process continues until node n_{ℓ} connects back to the head node n_1 . When node n_1 receives a connection message from n_{ℓ} , the ring is complete and each node knows its *next* node. The incremental learning process then begins. At each stage, node n_m locally trains the model incrementally based on (24) and then sends the updated model to the next node n_{m+1} .

The suitable node in \mathcal{N}' requires $O(\ell v)$ communication for incremental training after v passes. The leader node n_0 uses the incrementally trained model $f^{(\ell, v)}$ after $v \geq 1$ passes to serve the query q .

Table 1 summarizes the computational and communication complexities of the node selection mechanisms and the DML mechanisms engaging the leader node (receiving a DPA query) and the involved ℓ selected nodes.

6. Experimental evaluation

6.1. Experimental setup

6.1.1. Datasets

We assess the performance and efficiency of our mechanisms using two publicly available datasets to create realistic environments. The dataset DS1¹ contains multivariate data with data heterogeneity among nodes. DS1 includes weather observations from $N = 13$ European weather stations, which correspond to the edge nodes in our context. The dataset DS2² contains hourly air pollutant data from the Beijing Municipal Environmental Monitoring Center, covering $N = 12$ air-quality monitoring stations. These stations correspond to the edge nodes. The meteorological data for each air-quality station are matched with the nearest weather station from the China Meteorological Administration.

We chose DS1 and DS2 because they meet our study criteria, which include data collected from different locations. We further applied the data manipulation techniques Torra (2023), Arjovsky et al. (2020), Aubin et al. (2021) elaborated in on each node to achieve non-independently and identically distributed (non-i.i.d.) data over

the nodes via distribution drifts where correlated features shuffled between samples (the features contain both causes and effects of a target variable). Our target is to assess the impact on non-iid data and query access patterns in selecting nodes with relevant data for each DPA query. We evaluate this impact via the final DML model prediction accuracy, the node selection accuracy, and percentage of relevant data accesses as will be elaborated later. For both datasets, we conduct the same experiments following identical procedures to evaluate the best and worst-case scenarios for the proposed node selection mechanism and comparison mechanisms. In addition, in order to assess scalability of the node selection mechanisms in terms of number of nodes along with a fair comparison with the MAB-based mechanisms Puthiya Parambath et al. (2021) and Puthiya Parambath et al. (2024) dealing with a relatively large number of nodes for selection decisions (elaborated later), we adopted data augmentation techniques (including data blending via interpolation between samples) in Yan and Zhu (2023) and Alawani et al. (2024) over DS1 and DS2. As a result, we obtained non-i.i.d. data shared among $N = 1000$ nodes. Note, the developed code and datasets are publicly available here.³

6.1.2. Query workloads

We define $|Q_1| = 1100$ and $|Q_2| = 400$ DPA queries in our query workloads Q_1 and Q_2 for DS1 and DS2 dataset, respectively. Each query $q \in Q$ is randomly generated across the entire data space, following the query workload method described in Savva et al. (2019). The process of query generation under query distribution $P(q)$ is introduced in Anagnostopoulos and Triantafillou (2017a), Savva et al. (2019, 2020a), Kolomvatsos and Anagnostopoulos (2020), and Savva et al. (2020b). At time t , a DPA query q is uniformly distributed across the data space. For dimension x_j , the query center $q_j' = x_j^{\min} + (x_j^{\max} - x_j^{\min})\psi$ is uniformly sampled in $[x_j^{\min}, x_j^{\max}]$, with $\psi \sim U(0, 1)$. Thus, the boundaries for this dimension are $q_j^{\min} = \max(x_j^{\min}, q_j' - \frac{\zeta_j}{2})$ and $q_j^{\max} = \min(x_j^{\max}, q_j' + \frac{\zeta_j}{2})$, with uniformly sampled query length $\zeta_j = \psi'\sigma_j$ with $\psi' \sim U(0, 1)$ and σ_j being the standard deviation of the dimension x_j . Hence, the DPA generated queries cover the *entire* data space and avoids the exclusion of various sub-regions. To provide a better understanding of the query representation, a typical DPA range query in a smart city urban analytics application requesting parking sensors' data could be: $q = \{[10:30am, 11:00am], [30 \text{ min.}, 60 \text{ min.}], [60\%, 80\%]\}$, whose interpretation is elaborated on our example in Section 1.1. Such range queries are widely supported by modern database management systems as core part of aggregate queries (expressed in declarative languages like SQL) with range selection predicates (Idreos et al., 2015). Our DPA queries are queries with conjunctive selection predicates whose efficient computation has been a major research interest with methods applying sampling, synopses, and ML models to compute such queries; the interested reader could refer to Savva et al. (2020c) and there references therein for more information.

The queries are randomly assigned to nodes, with each node receiving a query with equal probability $\frac{1}{N}$. This random assignment facilitates the leader election process to identify leader nodes and their corresponding members for each issued query.

6.1.3. ML models for DPA

Each node quantizes its data in advance using a vector quantization method into K clusters. In our experiments, we used the k -means clustering algorithm, modified with the update rules for cluster heads under data changes. To avoid bias, all nodes use the same number of clusters K . The data space quantization determines the supportive clusters for node ranking purposes. Other quantization algorithms that can dynamically update cluster heads in evolving data subspaces, such as those discussed in Gu et al. (2022), may also be incorporated into our framework.

¹ https://github.com/florian-huber/weather_prediction_dataset

² <https://www.kaggle.com/datasets/sid321axn/beijing-multisite-airquality-data-set>

³ <https://anonymous.4open.science/r/JNCA-752C/README.md>

Our overarching goal is to train and examine the prediction accuracy of the models using only the *relevant data* predicted by incoming queries, rather than all available data. This helps in reducing the likelihood of training models on irrelevant data samples. We evaluate the predictive performance of these models compared to (i) the *ideal case*, i.e., accessing only the training data perfectly matched with the query boundaries in a centralized fashion, and (ii) the *ground truth*, i.e., accessing the predicted relevant data w.r.t. query boundaries in a distributed fashion.

To gain a deeper understanding of how our mechanism functions, we analyze its impact on the performance of ML and DL models for DPA queries. The rationale behind this selection of models is to cover the spectrum of relatively simple *but* yet powerful predictive models widely used in exploratory analytics (Savva et al., 2020c), query-driven big data analytics (Anagnostopoulos and Triantafillou, 2017b; Savva et al., 2019), decentralized data mining analytics (Savva et al., 2020b), and aggregate query-based predictive analytics (Anagnostopoulos and Triantafillou, 2017a; Savva et al., 2020a) over distributed contextual data. Specifically, we conducted experiments with three different predictive models derived from the above-mentioned applications of analytics families: Multivariate Linear Regression (LR), Non-linear/Polynomial Regression (PR), and Deep Neural Network (DNN) models. In addition to the wide adoption of these families of models for predictive analytics, their choice is justified by several key factors: (i) LR is computationally efficient and ideal for large datasets, making it a solid choice for initial data exploration and benchmarking (Savva et al., 2019, 2020c). (ii) PR provides the flexibility to capture more complex relationships in data selected by range and aggregate queries in large-scale databases without a significant increase in computational cost (Anagnostopoulos and Triantafillou, 2017a; Savva et al., 2020c). (iii) DNN models are well-suited for modeling highly complex and non-linear query access patterns for mining data regions in large-scale datasets (Savva et al., 2020b; Long et al., 2022) along with incremental regularization of compressed DNN models in data mining tasks (Long et al., 2023). Finally, LR and PR are straightforward to interpret in exploratory analytics (Savva et al., 2020c) aiding in the clear understanding the relevant data learnt by query access patterns along with developing re-usable ML models for DML training efficiency in edge computing environments, e.g., Long et al. (2024). The use of these diverse models, ranging from simple to complex, ensures a comprehensive evaluation of our mechanism across different applications and domains in the realm of DPA.

6.1.4. Model configuration, train & test datasets

The tuned hyperparameters of these predictive models are provided in Table 3 and are assessed using the Mean Squared Error (MSE). For fairness, the prediction error is evaluated using *ground truth* testing data for each query, which are *not* involved in any training process. Specifically, given a query q , we ideally aim to access the (distributed) data $D(q)$ defined in (1). These are the *actually* relevant data on which the model f should be trained according to the query specification. A sample of these data is used as the *test* dataset for evaluating the MSE of each DML model. Moreover, we train a DML model using our proposed mechanisms, which involves access *only* to the predicted relevant data from the selected nodes, i.e., data samples that have been identified as relevant by our mechanism to train our models. We then test the final DML model with the ground truth data for each query in question as explained above.

Consequently, given a query q , we *train* a DML model f over the *predicted* relevant distributed data $\cup_{i=1}^{\ell} \{D_i''(q)\}$ across all the ℓ selected nodes based on our DML mechanisms and, we *test* the model f 's MSE over the *actually* relevant data $D(q)$ for that query, i.e.,

$$MSE = \frac{1}{|D(q)|} \sum_{(x,y)} (y - f(x; \theta(q)))^2. \quad (25)$$

Furthermore, in order to report on the statistical significance of the performance of the methods compared with our mechanism we

Table 2

Experimental parameters.

Parameters	DS1	DS2
Nodes N	(13, 1000)	(12,1000)
Dimensionality d	11	18
Query workload size $ Q $	1100	400
Environment	Static, Dynamic (concept drift).	
Number of clusters K	(5, 10, 15)	
Overlapping threshold ϵ	(0.45, 0.55, 0.65, 0.75)	
Statistical level p -value	0.05	

Table 3

Predictive Model & DML Mechanisms Hyper-parameters.

Model	LR	PR	DNN
Local RFL rounds \mathcal{T}	20	20	20
Validation split	0.2	0.2	0.2
Learning rate	0.03	0.001	0.001
Number of RIL passes ν	5	5	5
RIL weighting factor ξ	0.2	0.2	0.2
Activation function	ReLU	ReLU	ReLU
Optimizer	Adam	SGD	Adam
Loss	MSE	MSE	MSE

Table 4

Performance Metrics.

Metric	Range
MSE	\mathbb{R}_+
Node selection accuracy α	[0,1]
Node selection frequency	[1, Q]
Percentage of relevant data accessed	[0,100]%
Percentage of node's data accessed	[0,100]%

adopt the non-parametric statistical Friedman's test (Grinberg et al., 2020). This is a robust test that handles non-normal distributions and outliers. Friedman's test, as an extension of the Wilcoxon signed-rank test (Grinberg et al., 2020), is widely used in regression analysis (a.k.a. two-way analysis of variance by ranks). Friedman's test has been carried out in order to identify the best and worst-performing methods, and statistically significant differences between pairs of models based on the predicted response values. By assigning ranks on prediction residuals in using Friedman's test, the average prediction performance by all the models is first computed for each test dataset. The differences between the performances of all the models and the average are then calculated, and are subsequently ranked. We examine the reported p -value compared with the statistical level of 0.05 to assess the significance in the prediction accuracy among the methods in comparison. In our experiments, we provide the test result value p compared against 0.05.

Tables 2, 3, and 4 list the experimental parameters, the hyper-parameters, and the performance metrics used in this paper.

6.2. Baselines & mechanisms under comparison

We evaluate our query-centric DML mechanisms: Best Node Model (BN), Aggregate Model (AM), Weighted Aggregate Model (WAM), Ranking-based Federated Learning Model (RFL), and Ring-based Incremental Learning Model (RIL) in producing accurate models for DPA. These mechanisms are assessed w.r.t. factors F1 and F2 (F1-2) and all factors (F1-3), as defined in Eqs. (12) and (14), respectively, over datasets DS1 and DS2.

We compare the performance of our mechanisms with the following baselines and relevant methods found in the literature: (i) the **Global Model (GM)**, where all the data are transferred in a centralized location used for centralized model learning per query (i.e., the Cloud) in a query-driven fashion (Anagnostopoulos, 2020). (ii) the **Random Model (RM)**, where ℓ nodes are randomly selected to be engaged in the DML process per query (Ye et al., 2020). (iii) the **Game Theory**

selection (GT) and (iv) the **Fair Selection (FS)** mechanisms introduced in Hammoud et al. (2022). In addition, we assess the scalability of our paradigm with comparing with the query-driven Reinforcement Learning and contextual Multi-armed Bandit (MAB) mechanisms for node selection: **Max Utility-based MAB (M-MAB)** (Puthiya Parambath et al., 2021) and **non-stochastic MAB with Transformer-based Experts and Feedback (TEF)** (Puthiya Parambath et al., 2024). M-MAB and TEF refer to online learning paradigms which are capable of selecting the subset of the most suitable nodes for engagement in DPA based on the trajectory of incoming DPA queries (contexts) as elaborated later. We chose TEF and M-MAB for comparative assessment since these mechanisms excel in incremental learning from a relatively high number of queries comparable to the number of nodes (using $N = 1000$ nodes in our comparison evaluation).

In GT mechanism, each node n_i builds its own independent local model f_i in advance according to its local dataset D_i . Moreover, in GT it is assumed that each node could exchange its model with all the other nodes in the network. When a node n_i receives the trained model f_0 from the leader node n_0 , it tests the f_0 model's performance locally against its data D_i and returns the results to the leader. The leader then targets those nodes that obtained accuracy lower than a pre-defined threshold. The rationale behind this is that a node could have data with different patterns than the leader's data. Therefore, the leader node selects those nodes n_i that have local models with relatively low accuracy in order to engage them for obtaining a global model. The learning process iteratively involves all the selected nodes' data for training the model.

The FS mechanism is a modification of the GT mechanism. While FS follows a similar approach, it incorporates additional fairness. Rather than always involving the same set of nodes, the FS mechanism ensures rotation by selecting the less frequently selected nodes at every round. This rotation ensures that over time, all nodes, irrespective of their initial performance or data, have an equal opportunity to participate and influence the global model. In essence, both the GT and FS algorithms emphasize fairness by ensuring that all nodes, regardless of the statistical patterns of their data or their models' initial performances, play an active and equitable role in the DML process.

The incremental learning mechanisms TEF and M-MAB are based on predicting the currently best sub-set of nodes to be selected via a trajectory of incoming DPA queries q_1, q_2, \dots, q_t . When the leader node is receiving a DPA query q_t (which plays the role of a context in these mechanisms), then the node recommends a sub-set of the most suitable nodes. The node ranking values are treated as the rewards to TEF and M-MAB for estimating the probability distribution of nodes being included in \mathcal{N}' . At every round t , TEF and M-MAB update such probability distribution w.r.t. current rankings $\{r_k\}_{k=1}^N$.

TEF is based on the MAB framework, which models the trade-off between exploration and exploitation over a sequence of actions (sub-set of suitable nodes in our case). TEF considers the query q_t as the context up to t , i.e., the sequence (q_1, q_2, \dots, q_t) . Such context is used for identifying the sub-set \mathcal{N}' of the most suitable nodes. Given this context, the TEF uses the top-rank r'' as the reward as explained in Puthiya Parambath et al. (2024). TEF then estimates the current distribution over the set of nodes in \mathcal{N}' , which is the basis for recommending the most suitable node by sampling from this distribution.

M-MAB deals with the *countably many arms* in the MAB framework, i.e., the cases where there are relatively many candidate sub-sets of suitable nodes available at every query. M-MAB introduces a selection strategy based on the maximum utility of the most suitable nodes in light of reducing the sizes of the sub-sets of suitable nodes to manageable sizes. This is achieved by selecting nodes with maximum utility with respect to the currently executing query. M-MAB adopts Zooming LinUCB (linear Upper Confidence Bound) (Kleinberg et al., 2019) to combine the upper confidence bound technique with an adaptive refinement step reducing the size of \mathcal{N}' . For each currently

executing query, M-MAB selects suitable nodes using LinUCB (Li et al., 2010), where the reward is a linear function of the query q .

Once the nodes for each query are predicted then, we compare the performance of the ML models for DPA trained by the DML mechanisms BN, AM, WAM, RFL, and RIL.

6.3. Performance evaluation

6.3.1. Node selection accuracy

We assess the efficiency of the node selection mechanisms by comparing the set of the most appropriate nodes \mathcal{N}' , predicted by our approach, with the ground truth (ideal) nodes \mathcal{N}^* given a random query q . In real-life scenarios, the ideal nodes \mathcal{N}^* are entirely unknown unless we access all nodes' data and train models in advance for each query; hence, they are referred to as ideal nodes. We introduce the metric *node selection accuracy*, defined as the ratio:

$$\alpha = \frac{|\mathcal{N}' \cap \mathcal{N}^*|}{|\mathcal{N}^*|} \in [0, 1]. \quad (26)$$

If the predicted selected nodes are included in the ideal nodes set, i.e., we accurately predict the most suitable ones for a query, then $\alpha \rightarrow 1$.

Additionally, we compare the node selection accuracy of our mechanism with that of a random node selection mechanism to highlight the importance of node selection and engagement per DPA query w.r.t. data relevance. In the random node selection, nodes are chosen entirely at random, meaning that the predicted set of nodes \mathcal{N}' is a random subset of the nodes \mathcal{N} .

It is worth noting that our mechanism implicitly controls the number of participants through the overlapping threshold ϵ , which determines the supportive clusters. This slightly influences the node selection accuracy, as shown in Figs. 6(a) and 6(b). Specifically, we present the metric α for our mechanism and the random selection with different overlapping thresholds $\epsilon \in \{0.45, 0.55, 0.65, 0.75\}$ over datasets DS1 and DS2. By assuming a relatively high ϵ (e.g., 0.75), we aim to select a few of the highest-ranking nodes using their supportive clusters (out of $K = 5$ clusters in the experiments shown in Figs. 6(a) and 6(b)). Conversely, a relatively low ϵ value (e.g., 0.45) results in selecting almost all nodes with acceptable ranks.

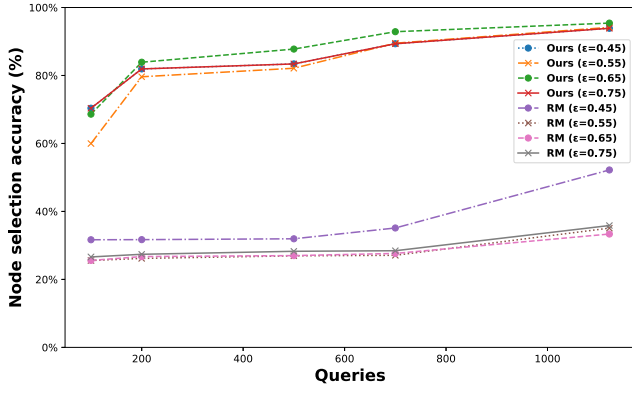
Our selection mechanism achieves high selection accuracy ($0.7 \leq \alpha \leq 0.95$) across the range of ϵ values. In contrast, random selection yields a selection accuracy between 0.3 and 0.5 across the same range of overlapping thresholds. This clearly indicates that random selection of nodes per query is not a viable solution in our context, as outlined in our rationale. Similar results are observed for $K \in \{10, 15\}$ in both datasets.

6.3.2. Static & dynamic data environments

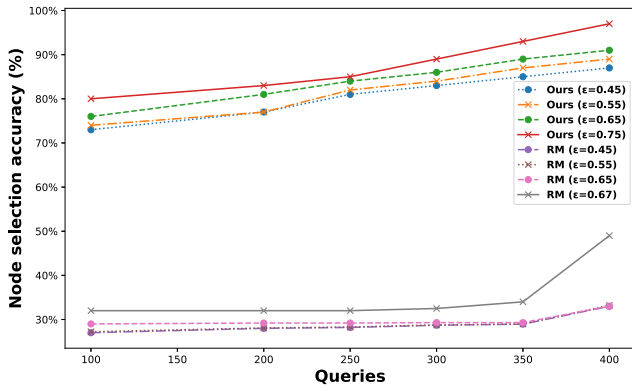
We further demonstrate the effectiveness of our mechanism w.r.t. dynamic data updates. Specifically, we investigate how the node selection accuracy is affected by dynamic environments and assess the robustness of our proposed mechanism in these scenarios. We conduct experiments in both fixed and dynamic environments regarding data updates over the nodes.

In a fixed data environment, we assume that updates on the underlying data distribution over nodes occur slowly. That is, the underlying data distribution does not significantly change over time, or changes are negligible. In contrast, dynamic data environments, such as those involving time-series data from surveillance applications as described in Kolomvatsos et al. (2022), exhibit frequent data updates in the underlying distributions. In these cases, data updates occur rapidly, with potential concept drifts where mean values may shift and/or new patterns may emerge.

Static Data Environment: Consider the experiment conducted in a static environment with non-i.i.d. data over the nodes from datasets DS1 and DS2. In this scenario, the nodes' data distributions remain



(a) Node selection accuracy α of our mechanism and the RM method with different overlapping threshold ϵ having $K = 5$ clusters per node; DS1 dataset.



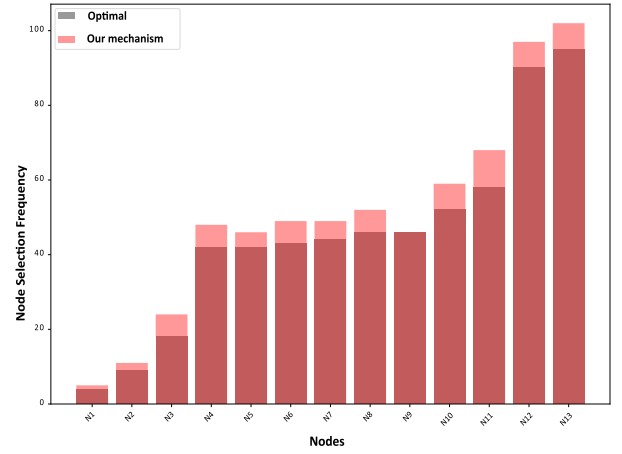
(b) Node selection accuracy α of our mechanism and the RM method with different overlapping threshold ϵ having $K = 5$ clusters per node; DS2 dataset.

Fig. 6. Node selection accuracy α for DS2 and DS1 datasets.

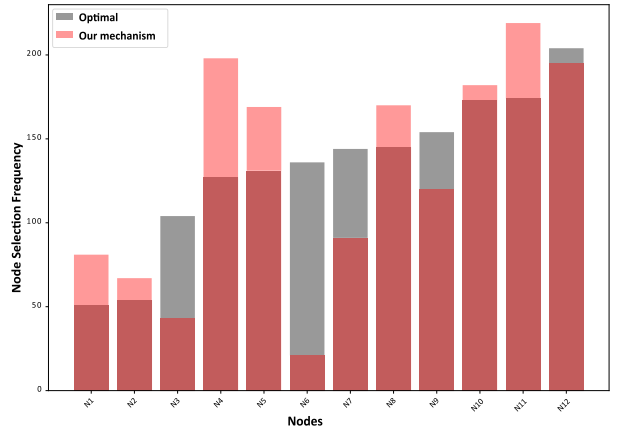
unchanged. Figs. 7(a) and 7(b) illustrate the disparity between the *node selection frequency* with which a node is predicted to be a participant by our mechanism and that of the optimal (ideal node) selection. This selection frequency indicates the number of times a node has been selected for a DML process out of the total number of queries. The results indicate that the difference between our mechanism and the optimal selection is relatively small for each node across all the queries considered. This suggests that our mechanism performs closely to the ideal node selection, demonstrating its effectiveness in static data environments.

Dynamic Data Environment: We consider a dynamic environment, where the underlying non-i.i.d. nodes' data distributions. This case influences the node selection mechanism w.r.t. data clusters. For details on how the cluster heads and the associated cluster boundary vectors are updated in response to changes in the underlying data refer to Section 4. We performed four sequential changes (concept drifts) of the data distribution changing progressively 25% of the selected nodes' data distributions parameters (including mean and skewness values). For each change, we applied our node selection mechanism to predict the suitable set of nodes per query. We then compared their node selection frequencies with the frequencies of the nodes in the optimal case.

As we can observe in Figs. 8(a) and 8(b), in each progressive change (from 25% to 100% total changes), our mechanism remains robust in selecting the most suitable nodes per query by gradually and incrementally updating the nodes' cluster-heads, and therefore,



(a) Static data environment: node selection frequency for our mechanism and the optimal (ideal node selection) one with $\epsilon = 0.65$ and $K = 5$ clusters per node; DS1 dataset.



(b) Static data environment: node selection frequency for our mechanism and the optimal (ideal node selection) one having $\epsilon = 0.65$ and $K = 5$ clusters per node; DS2 dataset.

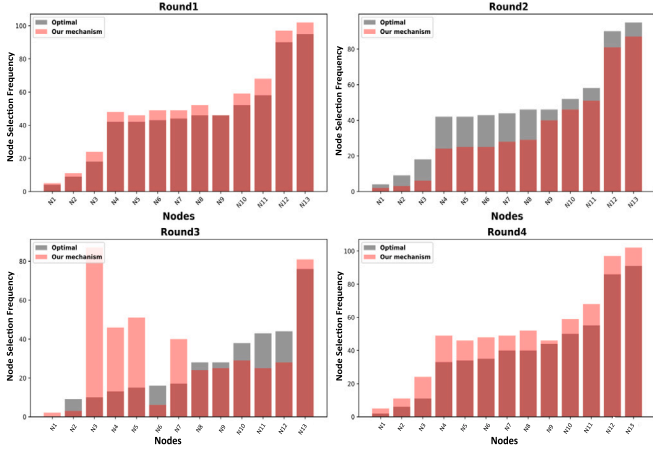
Fig. 7. Node selection frequency for our mechanism and the optimal one in static data environments.

updating the cluster boundary vectors reflecting these data changes. We obtain similar results and behavior of our mechanism over the datasets (similar results are obtained for $K \in \{10, 15\}$ in both datasets).

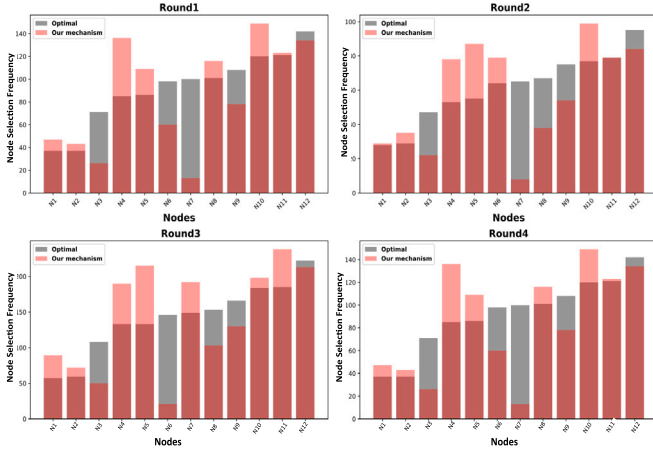
6.3.3. Relevant data access rate

It is so far evidenced that our mechanism is capable of engaging the most suitable nodes to be involved in the DML process under static and dynamic data environments. We therefore examine the *amount* of relevant data accessed locally, which drives the efficiency of our mechanism.

After predicting the most suitable nodes, we examine the impact of the number of clusters K on the amount of relevant data used by the selected nodes during DML training. Our mechanism aims to reduce the amount of irrelevant data samples by increasing the number of clusters K in the nodes. An increase in K enhances the segregation between relevant and irrelevant samples. When K is relatively small, each cluster may contain a higher percentage of irrelevant samples. Conversely, increasing the number of clusters helps reduce the access to irrelevant samples within the supportive clusters.



(a) Dynamic data environment: node selection frequency for our mechanism and the optimal (ideal node selection) one having $\epsilon = 0.65$ and $K = 5$ clusters per node over four sequential phases of data updates; DS1 dataset.



(b) Dynamic data environment: node selection frequency for our mechanism and the optimal (ideal node selection) one having $\epsilon = 0.65$ and $K = 5$ clusters per node over four sequential phases of data updates; DS2 dataset.

Fig. 8. Node selection frequency for our mechanism and the optimal one in dynamic data environments.

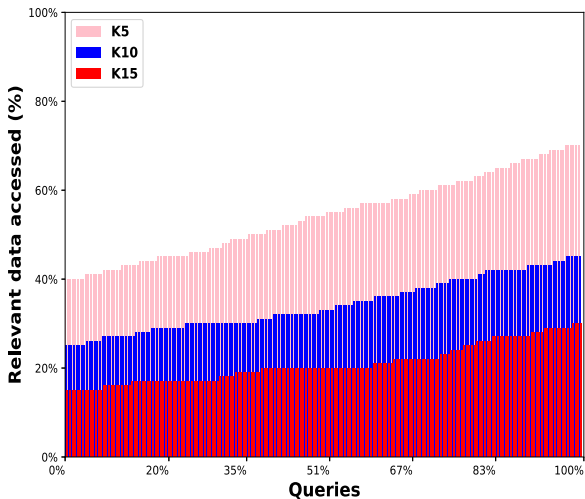


Fig. 9. Percentage of relevant data accessed per proportion (%) of DPA queries in our mechanism adopting F1-3 factors; DS1 dataset.

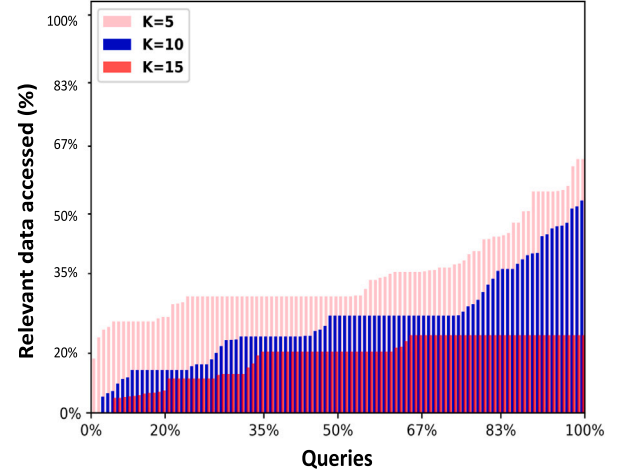


Fig. 10. Percentage of relevant data accessed per proportion (%) of DPA queries in our mechanism adopting F1-3 factors; DS2 dataset.

Figs. 9 and 10 illustrate how varying the number of clusters helps filter out irrelevant data samples accessed by the nodes over the DS1 and DS2 datasets, based on the three factors F1, F2, and F3. The vertical axis represents the percentage of data accessed across issued queries per node (on average). The bars indicate the percentage of data needed for each query from all selected nodes, using different K values per node. Evidently, for most queries, less than 20% to 40% of the total data are accessed, demonstrating the selective capability of our mechanism to involve *only* the required data per query. Accessing all available data typically deteriorates model performance, as will be shown later.

Table 5 summarizes the percentage of data accessed relative to number of clusters K for our mechanism, compared with benchmark methods across both DS1 and DS2 datasets in static and dynamic data environments. It is worth noting that when using the three factors F1, F2, and F3 (F1-3 in Table 5) for data selection per query per node, we achieve a high probability of accessing all relevant data per query from the supportive clusters (but not all the data in each supportive cluster). This effectiveness is particularly pronounced with higher K values. In contrast, the GT, FS, RM, and central GM mechanisms access all available data. Moreover, in Table 5 we show the percentage of data accessed by nodes selected using TEF, M-MAB and our method (using all factors F1-3 with $K = 15$ clusters) for $N = 1000$ nodes in both DS1 and DS2. Both TEF and M-MAB methods attempt to predict the most suitable nodes in light of increasing the probability of accessing relevant data over a relatively high number of nodes. However, our paradigm outperforms the MAB methods by being significantly selective in accessing all the relevant data thus avoiding redundant access of irrelevant data used to train the DML models. Our mechanism achieves 65% and 29% less redundant data access compared to TEF and M-MAB, respectively. This is attributed to the filtering mechanisms introduced in factors F1-F3 focusing explicitly on the characteristics of each incoming DPA query avoiding the summarization statistics over the queries' patterns trajectories as achieved by TEF and M-MAB.

6.3.4. Comparative assessment

We compare the prediction error (MSE) over the test datasets in (25) of the models LR, PR, and DNN trained via the selected nodes by our mechanism against the methods GM, RM, GT and FS across all the query workloads Q_1 and Q_2 for the dataset DS1 and DS2, respectively. Moreover, we compare our mechanism adopting the three factors (F1-3) with the M-MAB and TEF models under $N = 1000$ nodes to study the scalability performance. Once the most suitable nodes have been

Table 5

Percentage of data accessed per query across node selection mechanisms in static and dynamic environments over DS1 and DS2.

Mechanism		(DS1)		(DS2)	
		Static	Dynamic	Static	Dynamic
GM		100%	100%	100%	100%
F1-3	$K=5$	39.88%	55.20%	69.13%	75.55%
	$K=10$	32.17%	52.67%	67.22%	64.90%
	$K=15$	20.92%	51.04%	65.27%	62.50%
RM		100%	100%	100%	100%
GT		100%	100%	100%	100%
FS		100%	100%	100%	100%
MAB Models		$N = 1000$		$N = 1000$	
		Static	Dynamic	Static	Dynamic
TEF		67%	79%	72%	79%
M-MAB		71%	78%	77%	81%
F1-3($K = 15$)		19%	29%	55%	57%

predicted by each method, we apply the DML mechanisms BN, AM, WAM, RFL, and IRL over these nodes to train the predictive models.

Tables 6 and 8 show the performance of the predictive models w.r.t. MSE across all DPA queries in static data environments over DS1 and DS2 datasets. Both tables show the comparison of the performance of the models having built based on nodes selected based on our mechanism in the cases: adopting F1 factor and F1-3 factors for different

K values compared with the central GM, GT, FS, and RM approaches. Moreover, the TEF and M-MAB methods are compared against our F1-3 mechanism (with $K = 15$ clusters) selecting sub-sets over $N = 1000$ nodes. The selected nodes trained the different predictive models (LR, PR, and DNN) according to the query-centric DML mechanisms BN, RFL, RIL, AM, and WAM.

From the results, our mechanism consistently outperforms other node selection mechanisms even in the case with $N = 1000$ over the same queries ($p < .05$). Notably, in terms of the query-centric DML mechanisms, RFL demonstrates the closest approximation to the GM training outcome, while BN mostly shows good results for GT, FS and RM, however, with no comparable accuracy results ($p < .05$). Specifically, the disparities between GM error and our results are on average 3.8% across all predictive models. This indicates that our mechanism achieves comparable accuracy under circumstances of not allowing data transfer to central locations including violation of data privacy apart from unconditional access to data. Moreover, given the unanimous agreement across all models regarding RFL's superiority, it is considered as the preferred choice for static data environments, especially when prioritizing model performance.

However, one must remain aware of potential communication overhead with RFL compared to the other query-centric DML schemes, e.g., BN and A/WAM. While RFL requires \mathcal{T} training rounds among selected nodes, this results in more communication compared to BN, which simply engages a single node. RFL also surpasses the communication rounds in both AM and WAM, as they only demand one round of interaction between selected nodes. Moreover, RFL entails a higher

Table 6

MSE (loss) of predictive models based on all the node selection mechanisms over DS1 across all the query-centric DML mechanisms (static data environment).

DS1 Static Environment							
Node Selection		Learning	BN	RFL	RIL	AM	WAM
		Models					
Ours (F1)		LR	36.08	32.44	37.67	55.44	35.83
		PR	40.38	32.71	35.12	61.56	36.74
		DNN	38.05	31.03	33.56	56.98	34.60
F1-3	Ours $K = 5$	LR	35.6	31.81	35.25	53.57	34.12
		PR	39.23	31.83	34.61	60.27	33.01
		DNN	35.71	31.96	31.45	56.71	33.17
	Ours $K = 10$	LR	34.98	30.74	35.82	51.22	33.98
		PR	38.67	30.15	33.87	58.23	32.33
		DNN	33.56	29.79	30.48	54.29	30.11
	Ours $K = 15$	LR	33.20	30.87	32.05	51.22	34.98
		PR	32.77	31.77	37.52	57.96	33.89
		DNN	31.43	28.53	30.68	55.75	31.51
GM	LR (27.91)		–	–	–	–	
	PR(29.39)		–	–	–	–	
	DNN (27.85)		–	–	–	–	
GT	LR		57.45	85.43	88.78	97.59	89.10
	PR		63.50	72.57	73.33	98.04	91.61
	DNN		46.01	72.88	73.67	82.04	76.69
FS	LR		60.49	87.82	89.18	95.22	90.46
	PR		63.88	78.40	79.67	83.32	80.41
	DNN		53.02	72.48	74.25	78.39	74.95
RM	LR		73.94	98.82	97.39	112.87	98.98
	PR		73.53	85.50	71.20	93.61	87.26
	DNN		87.53	77.59	79.48	84.51	81.30
DS1 Static Environment; MAB Mechanisms ($N = 1000$)							
TEF	LR		53.44	48.42	47.49	42.44	48.66
	PR		43.43	45.15	41.32	53.51	57.44
	DNN		47.55	47.69	49.45	54.61	51.35
M-MAB	LR		63.54	68.62	67.69	52.46	58.76
	PR		48.40	57.52	50.52	49.15	58.84
	DNN		49.95	50.66	55.76	59.99	62.33
F1-3($K = 15$)	LR		28.04	28.28	27.95	28.70	29.98
	PR		30.23	20.20	29.90	33.71	32.25
	DNN		27.94	29.57	27.88	26.51	28.50

Table 7

MSE (loss) of predictive models based on all the node selection mechanisms over DS1 across all the query-centric DML mechanisms (dynamic data environment).

DS1 Dynamic Environment						
Node Selection	Learning Models	BN	RFL	RIL	AM	WAM
Ours (F1)	LR	39.29	33.46	35.47	57.53	34.22
	PR	42.00	34.76	35.55	65.81	35.25
	DNN	44.72	33.16	34.51	49.35	33.26
F1-3	Ours $K = 5$	LR	38.95	32.78	34.67	34.56
		PR	41.28	33.23	34.92	33.02
		DNN	43.54	33.40	31.19	32.36
	Ours $K = 10$	LR	37.53	32.04	33.05	33.88
		PR	39.65	33.26	34.30	32.72
		DNN	37.71	32.02	30.67	30.04
	Ours $K = 15$	LR	37.03	30.43	32.34	33.09
		PR	37.33	31.46	32.07	33.93
		DNN	36.02	31.52	29.56	31.22
GM	LR (27.91)	–	–	–	–	–
	PR(29.39)	–	–	–	–	–
	DNN (27.85)	–	–	–	–	–
GT	LR	85.22	96.15	101.27	111.18	96.94
	PR	72.25	79.27	83.81	86.92	82.53
	DNN	71.18	74.51	78.66	82.45	75.12
FS	LR	88.61	95.03	98.18	102.22	94.21
	PR	61.05	85.81	88.75	95.39	86.15
	DNN	69.63	73.75	76.68	88.81	74.62
RM	LR	65.79	128.21	132.79	130.31	93.27
	PR	70.81	93.67	96.24	97.43	95.21
	DNN	68.63	80.24	86.07	93.04	88.43
DS1 Dynamic Environment; MAB Mechanisms ($N = 1000$)						
TEF	LR	56.64	68.62	57.59	52.54	58.78
	PR	47.73	43.35	40.44	40.41	47.54
	DNN	43.53	57.59	59.55	64.66	61.85
M-MAB	LR	77.54	70.72	77.98	72.76	78.88
	PR	78.70	77.56	70.77	79.57	78.47
	DNN	69.65	70.76	75.16	69.11	72.66
F1-3($K = 15$)	LR	33.34	28.38	30.95	28.09	29.33
	PR	29.53	29.52	29.45	30.15	30.11
	DNN	28.57	28.33	29.28	30.15	33.22

Table 8

MSE (loss) of predictive models based on all the node selection mechanisms over DS2 across all the query-centric DML mechanisms (static data environment).

DS2 Static Environment						
Node Selection	Learning Models	BN	RFL	RIL	AM	WAM
Ours (F1)	LR	42.53	35.26	40.84	61.35	52.49
	PR	43.77	36.25	41.41	70.49	53.04
	DNN	35.91	32.15	37.45	58.04	49.07
F1-3	Ours $K = 5$	LR	40.15	34.25	39.31	48.82
		PR	42.00	36.22	43.77	47.17
		DNN	35.01	31.75	36.92	47.13
	Ours $K = 10$	LR	39.20	33.26	38.42	47.43
		PR	42.53	35.97	42.21	48.00
		DNN	34.98	31.09	36.27	47.11
	Ours $K = 15$	LR	38.02	32.41	37.42	46.24
		PR	41.88	34.48	41.12	47.66
		DNN	33.87	30.78	35.98	46.08
GM	LR (30.57)	–	–	–	–	–
	PR (32.34)	–	–	–	–	–
	DNN (30.29)	–	–	–	–	–
GT	LR	65.45	79.99	82.33	92.24	84.05
	PR	67.23	88.61	88.38	98.77	92.87
	DNN	65.34	76.65	77.23	95.07	82.88

(continued on next page)

Table 8 (continued).

FS	LR	43.77	62.26	68.20	90.12	88.81
	PR	46.36	67.07	71.91	94.27	86.33
	DNN	42.78	60.21	69.22	93.41	85.12
RM	LR	51.17	90.65	95.17	111.12	93.82
	PR	53.34	99.31	102.07	107.33	98.67
	DNN	50.19	87.00	91.44	97.29	94.15
DS2 Static Environment; MAB Mechanisms ($N = 1000$)						
TEF	LR	55.66	67.76	59.59	62.74	57.77
	PR	49.91	50.30	50.44	55.41	57.99
	DNN	47.77	42.22	52.22	60.01	62.27
M-MAB	LR	75.44	72.11	70.01	70.16	73.77
	PR	74.50	67.66	67.98	77.87	69.45
	DNN	66.02	70.11	64.23	71.11	69.96
F1-3($K = 15$)	LR	31.14	30.81	32.13	34.31	33.28
	PR	34.44	33.01	34.98	33.35	38.65
	DNN	31.44	30.48	31.77	32.55	36.34

Table 9

MSE (loss) of predictive models based on all the node selection mechanisms over DS1 across all the query-centric DML mechanisms (dynamic data environment).

DS2 Dynamic Environment							
Node Selection		Learning	BN	RFL	RIL	AM	WAM
		Models					
Ours (F1)		LR	47.40	38.35	42.11	68.28	39.49
		PR	48.47	39.33	39.45	70.49	40.59
		DNN	49.16	37.77	37.13	59.30	39.32
F1-3	Ours $K = 5$	LR	46.60	37.37	41.59	63.78	38.46
		PR	47.27	38.81	33.19	65.29	39.62
		DNN	47.25	36.99	37.17	57.31	38.40
	Ours $K = 10$	LR	46.25	36.08	37.22	62.10	37.37
		PR	46.06	37.25	35.90	63.57	38.29
		DNN	45.02	35.05	36.88	56.37	38.67
	Ours $K = 15$	LR	46.55	33.98	35.18	62.77	37.03
		PR	45.20	36.82	33.71	62.32	36.14
		DNN	44.17	32.39	34.48	55.18	35.26
GM	LR (30.57)		–	–	–	–	
	PR (32.34)		–	–	–	–	
	DNN (30.29)		–	–	–	–	
GT	LR		64.18	79.99	82.32	114.00	84.75
	PR		65.24	88.61	92.24	116.00	85.15
	DNN		61.52	74.78	76.23	113.37	81.09
FS	LR		73.12	76.44	82.69	107.38	84.63
	PR		75.15	77.83	77.19	109.32	86.91
	DNN		73.13	74.46	75.63	111.50	83.41
RM	LR		71.00	90.65	95.17	111.12	93.82
	PR		53.34	99.31	102.07	107.33	98.67
	DNN		50.19	87.00	91.44	97.29	94.15
DS2 Dynamic Environment; MAB Mechanisms ($N = 1000$)							
TEF	LR		65.55	51.11	60.11	61.43	67.89
	PR		44.43	41.22	49.33	45.33	59.01
	DNN		57.88	38.33	42.44	51.21	52.88
M-MAB	LR		65.64	62.61	60.44	73.36	74.01
	PR		64.60	57.56	57.08	57.89	59.94
	DNN		56.22	61.12	67.07	66.20	58.76
F1-3($K = 15$)	LR		33.21	31.11	32.23	33.91	32.55
	PR		33.97	33.16	34.29	34.39	35.25
	DNN		31.11	30.44	32.66	32.12	34.77

communication compared to RIL, which the latter circulates a single model among the selected nodes forming a logical ring. Notably, our mechanism F1-3 achieves better performance in the case of $N = 1000$ nodes by adopting the RIL method. This indicates the efficiency of our mechanism in selecting and engaging the most suitable nodes out of a network with a relatively high number of nodes. Furthermore, TEF and M-MAB methods produce similar predictive performance across different DML mechanisms (like BN, RIL, and AM) achieving, however,

39.02% more error (on average) compared with our F1-3 mechanism ($p < .05$).

In DS1, static data environment (Table 6), we obtain a trade-off between accuracy and communication overhead by adopting RFL (and RIL for $N = 1000$) against other less communication intensive query-centric DML mechanisms. We can slightly sacrifice some of the obtained accuracy, e.g., around 1% less, by adopting RIL or WAM instead ($p < .05$). Similar trade-off is obtained in DS2, static data environment,

Table 10
Nomenclature.

Symbol	Definition
$\mathcal{N} = \{n_1, n_2, \dots, n_N\}$,	
$\mathcal{N}' \subset \mathcal{N}$, $\mathcal{N}^* \subset \mathcal{N}$	Set of nodes, selected nodes, ideal selected nodes.
$\ell = \mathcal{N}' \leq N$	Number of selected nodes.
$\mathbf{x} = [x_1, \dots, x_d]^T \in \mathcal{X}$, $y \in \mathcal{Y}$	Input and output.
d	Data dimensionality.
\mathbf{q}	DPA query.
B	Hyper-rectangle in \mathcal{X}
$D_i = \{(\mathbf{x}, y)_k\}_{k=1}^{L_i}$	Node n_i 's local dataset with L_i samples.
$f(\mathbf{x}; \theta(\mathbf{q}))$	DML model; $\theta(\mathbf{q})$ parameters.
$D(\mathbf{q}) = \cup_{i=1}^N D_i(\mathbf{q})$, $D'_i \subset D''_i \subset D$	Whole distributed data and relevant data per query/node.
\mathcal{L}	Loss function.
q_k^{\min}, q_k^{\max}	Boundaries per query.
e_j, e_G	Error of central and local nodes.
$\{C_1, \dots, C_K\}$	Set of K clusters.
$\mathbf{c}_k, \mathbf{c}'_k$	k th cluster-head & boundary.
$r_i(\mathbf{q}), r'_i(\mathbf{q}), r''_i(\mathbf{q})$	Ranking support for n_i .
$h_k(\mathbf{q}) \in [0, 1]$	Query-cluster overlapping.
$K' \leq K$	Number of supportive clusters.
$\epsilon > 0$	Overlapping threshold.
$t, \tau \in \mathbb{T}, \mathcal{T}$	Discrete time, training epochs.
α	Node selection accuracy.
ν	Ring passes (RIL).

Table 11
Acronyms & Abbreviations.

Acronym	Abbreviation
DPA	Distributed Predictive Analytics
DML	Distributed Machine Learning
MSAP	Minimum Subset Average Problem
BNM	Best Node Model
MAD	Median Absolute Deviation
AM	Aggregate Model
WAM	Weighted Aggregate Model
FL	Federated Learning
RFL	Ranking-based Federated Learning
DS	Dataset
ML	Machine Learning
DL	Deep Learning
LR	Linear Regression
PR	Polynomial Regression
DNN	Deep Neural Network
MSE	Mean Squared Error
RIL	Ring-based Incremental Learning
GM	Global Model
RM	Random Model
GT	Game Theory
FS	Fair Selection
RL	Reinforcement Learning
MAB	Multi-armed Bandits
M-MAB	Max Utility-based MAB
TEF	Transformer Experts/Feedback MAB

(Table 8). For $K = 15$, our selection mechanism adopting RFL and RIL achieves almost similar accuracy prediction levels with the GM. This indicates the capability of our mechanism to offer distributed learning over DPA queries without accessing the data in advance nor transferring the data in a central location.

In dynamic environments, similar to the static ones, our mechanism surpasses other techniques, with RFL, RIL, and WAM emerging as best performers for most of the predictive models. Tables 7 and 9

present the performance of the node selection mechanisms in dynamic data environments, where data have undergone changes described in Section 6.3.1. It is worth noting that in this context, RIL is considered as an efficient query-centric scheme to improve the model performance and reduce the communication rounds. However, this requires the communication dependency among nodes during the ring-based model training. Furthermore, our mechanism does not only select the most suitable nodes, it also reduces redundant access to irrelevant data as discussed in Section 4.1. Each query, on average, requires access only to 10% ~ 15% of data on a selected node. In DS2, WAM seems also another efficient candidate in some cases compared to RFL and RIL, e.g., for PR and DNN models when $K \in \{10, 15\}$ trading off accuracy with communication overhead. Furthermore, in the case of $N = 1000$ nodes, RFL is proved efficient for TEF and our mechanism, while RIL is a promising candidate for M-MAB mechanism. Even in this case, our mechanism outperforms TEF and M-MAB in terms of prediction accuracy ($p < .05$) while RFL scheme achieves 26.55% and 41.01% less error (on average) compared to TEF and M-MAB, respectively.

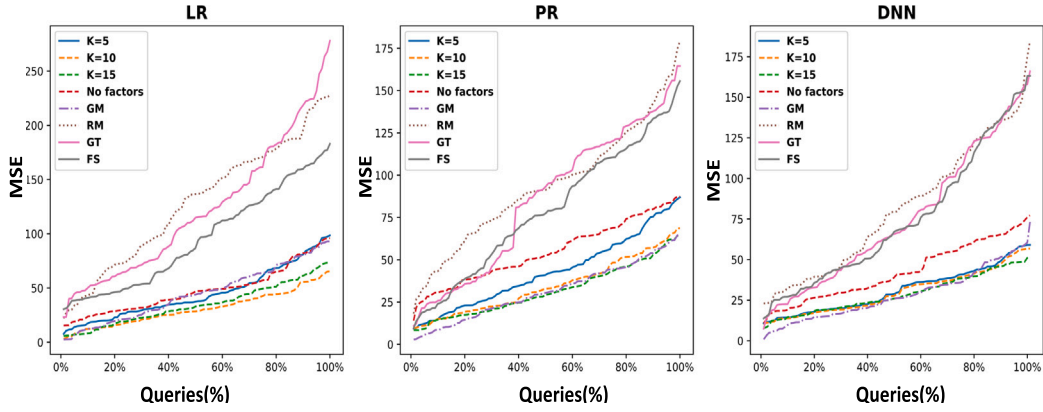
6.3.5. Discussion on generalization capacity of ML models for DPA

Our preemptive mechanism ensures that the subsequent training phase leverages the most relevant data out of all selected nodes' data. This yields each DML-trained model *tailored* to each DPA query. Such training process over relevant samples might affect the generalization capacity of the derived ML model for DPA, i.e., its ability to adapt to new, previously unseen data, yet drawn from the same distribution as the one used to train the model (the distribution of the relevant data). Our mechanism attempts to balance this potential lack of generalization by introducing the factors F1, F2, and F3 in Section 4. Recall that each factor controls the 'degree of relevance' of the data used for training with F2 and F3 attempting to predict the highest portion of relevant data given a query. As evidenced in our experimental results in Tables 6, 7, 8, and 9 (refer to performance of F1 and F1-3 variants), the derived models obtain almost similar MSE with the ideal GM method, which uses only the most relevant data for training given a query (ground truth). Evidently, by adopting our mechanism with all factors involved (i.e., F1-3 variant), the derived model's generalization capacity is decreased (due to overfitting).

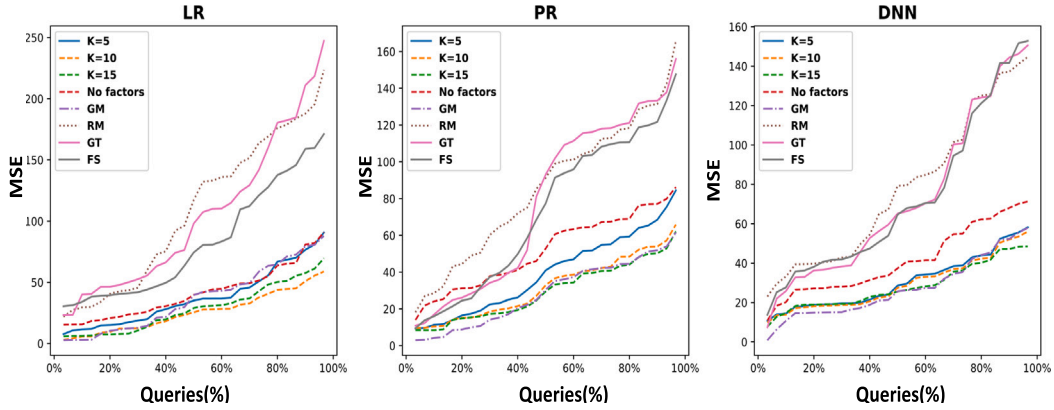
On the other hand, our mechanism's variants with factor F1 result in not aggressively filtering our irrelevant training data given a query. In these variants, the derived model is trained over a *mixture* of (predicted) relevant samples (as strictly requested by a DPA query specifications) and irrelevant samples, which could not be excluded due to the F1 factor. In this case, we obtain the following trade-offs: a highly tailored model (based on the F1-3 variant satisfying the DPA query) with low model generalization capacity, and a less tailored model (based on the F1 factor) with higher generalization capacity. Depending on the DPA application and analytics domain, once could balance between a generalized model and a model that fits the requirements of the DPA query. For instance, models for exploratory analytics applications, e.g., Savva et al. (2020c) can be obtained using our mechanism with F1 and/or F2 variants to allow for generalization, while models for aggregate predictive analytics in large-scale databases, e.g., Savva et al. (2020a), Anagnostopoulos and Triantafillou (2017b) can be obtained using our F1-3 variant, where model fitting over large-scale datasets is significantly required (closely match with the relevant data allowing for accurate predictions and pattern mining).

6.3.6. Discussion on comparative methods performance

Considering now the approaches under comparison, i.e., GT, FS, RM, TEF and M-MAB, it is required to access all the data in advance to establish the most suitable nodes. As stated by Goetz et al. (2019), selection of nodes is based on a value function, which reflects the usefulness of the data during each training round. Moreover, the MAB methods in Puthiya Parambath et al. (2021) and Puthiya Parambath et al. (2024) require online training to achieve convergence in determining the most



(a) Distribution of MSE vs. proportion (%) of queries using RFL across different models in static data environment (DS1 dataset).



(b) Distribution of MSE vs. proportion (%) of queries using RFL across different models in dynamic data environment (DS1 dataset).

Fig. 11. Distribution of MSE against proportion (%) of DPA queries (from the query workloads) using RFL in static and dynamic data environments (DS1).

suitable sub-set of nodes per DPA query. During this training period (exploration-exploitation rounds), these methods require full access to the data to quantify the reward value per round.

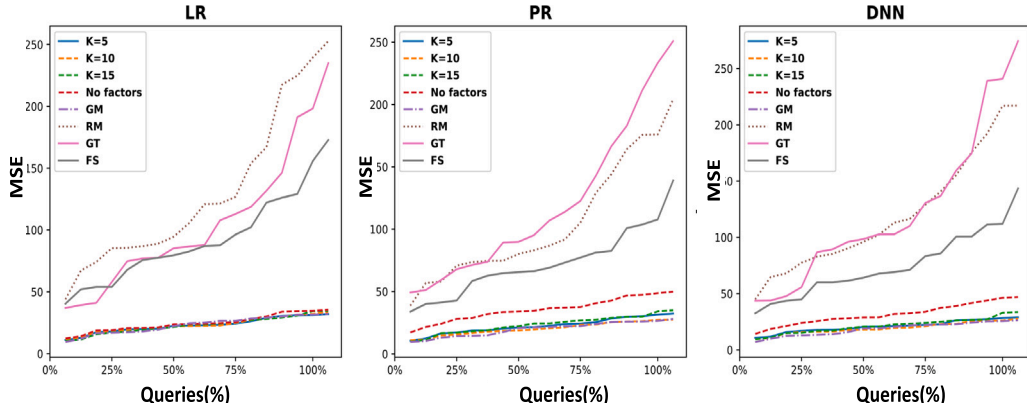
Our mechanism is based a pre-training model principle, i.e., the nodes are predicted to be suitable for a DPA query before attempting any DML process. Whereas, GT, FS, and TEF and M-MAB methods are based on a post-training model principle, i.e., the nodes are predicted to be suitable for a DPA query after training a predictive model (for TEF and M-MAB this holds up to the conference stage). A commonality among these approaches is their dependency on building the predictive models first; then, based on these models, they determine the suitable set of nodes. Notably, even after the node selection process, they do not focus explicitly on the relevant data requested by the queries. In contrast, in our mechanism, we prioritize both the selection of suitable nodes and the identification of relevant data before initiating the training stage.

In terms of data access, as shown in Fig. 10, the number of irrelevant samples has significantly reduced when considering the clustering approach compared to the comparison methods. In addition, we improved the model performance by surgically identifying the relevant data to be accessed only, for instance, in Table 6 the performance of RFL over DNN is increased by $\sim 10\%$ by adopting all the factors (F1-3) compared with only considering the F1 factor. This trend holds for the rest of the query-centric DML mechanisms.

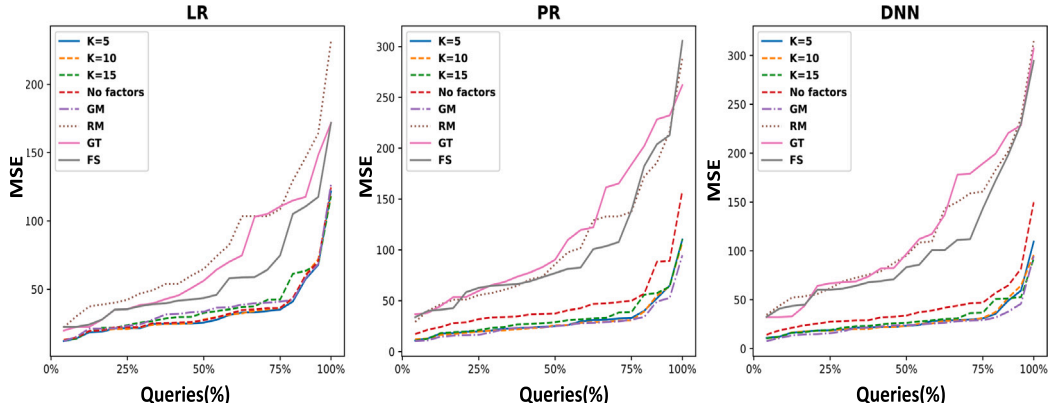
In terms of communication rounds among nodes during the DML process, our mechanism can reduce the communication by avoiding engaging unsuitable subsets of nodes in one round. This is because

only one round is dedicated to determining the node rankings based on supportive clusters using F1-3 factors, followed by determining the amount of data in each relevant cluster. Using the F1 and F2 factors, no communication round between the leader and the nodes is required. If the conditions in factors F1 and F2 are satisfied then the DML process commences. Subsequently, we assess the number of relevant samples in each separate cluster. Otherwise, a node will not participate in the training stage.

Given the fact that RFL are mostly superior than the other DML mechanisms schemes across static and dynamic environments, datasets, and predictive models, we further provide a detailed comparative assessment of our mechanism (both, adopting F1 and F1-3 factors) against the approaches under comparison by showing the *distribution of the MSE per query* in Figs. 11 and 12. Note that similar results are obtained for the rest of the DML mechanisms. We obtain insights about the percentage of the DPA queries served w.r.t. prediction error across each query workload, and how all the mechanisms achieve low MSE across all incoming queries. Specifically, Figs. 11 and 12 show that given a MSE value, which proportion of the issued queries (out of all queries in the workload) assumes error up to that MSE value. It is evidenced that our mechanism is efficient in terms of selecting the most appropriate nodes to train a DML model per query. As evidenced by the performance of FS, GT, and RM in both datasets for both environments, the performance of each query individually is negatively impacted when considering 100% data access during the DML process. That is, for more than 50% of the queries, the obtained MSE by FS, GT and RM is significantly higher than our mechanism.



(a) Distribution of MSE vs. proportion (%) of queries using RFL across different models in static data environment (DS2 dataset).



(b) Distribution of MSE vs. proportion (%) of queries using RFL across different models in dynamic data environment (DS2 dataset).

Fig. 12. Distribution of MSE against proportion (%) of DPA queries (from the query workloads) using RFL in static and dynamic data environments (DS2).

Moreover, one can observe the robustness of our mechanism where for up to 90% of the queries (see Fig. 12; both static and dynamic data environments), the obtained MSE remains almost constantly in low levels while being comparable with that of GM. This yields our mechanism applicable in the real of DPA.

6.4. Limitations & directions of enhancement

Node selection in DML environments has several perspectives related to DPA. Our paradigm predicts the most suitable nodes holding the most relevant data that will be used for training ML models in a distributed manner. Our mechanisms excel where there is significant heterogeneity and diversity between the query access patterns and the available data in each node. Hence, by selecting a sub-set of nodes for each incoming DPA query into the system is of paramount importance in terms of our objectives: model performance and data access load. Nonetheless, our node selection process needs to be expanded to accommodate different perspectives. Specifically, node selection can take into consideration the accessibility of nodes due to the network status and connectivity. One limitation is that DPA tasks are based on the assumption that selected nodes are also accessible and available during model training. This implies that these nodes should devote their resources and computational load when requested to be engaged in ML models distributed learning. However, ‘stragglers’ (nodes with limited computational capacity) and heavy loaded nodes might hinder

the entire process. Proactive mechanisms for refining the outcome of node selection e.g., Kolomvatsos et al. (2022) or providing guarantee about the capacity of the selected nodes e.g., ALFahad et al. (2024) is a limitation in our paradigm.

In addition, our paradigm does not focus on the system resilience in DML environments. Even if the selected nodes have been proved to accommodate the learning process for a specific DPA query, failure rates, nodes attacks, and unexpected events yielding some of the selected nodes incapable to support the learning process would jeopardize the DPA tasks (Wang et al., 2023). Factors including device heterogeneity, heterogeneity in network connectivity (e.g., mobile and stationary nodes), and incentives for nodes devoting their resources for training define perspectives and directions where our current paradigm does not include in the node selection process in this work.

Finally, in terms of the nature of data, our mechanism may be less effective with other forms of selection queries like object recognition over moving images, and multi-modal learning of ML models over simultaneously different types of data like text, audio, or images. Moreover, when the amount of data is similar across many nodes, then this would require our mechanism to be enhanced with methods to eliminate potential nodes hosting data with similar distributions to avoid redundancy in the learning process. The above-mentioned perspectives for enhancement of our paradigm are included in our future agenda for developing a holistic paradigm in the emerging area of distributed analytics.

7. Conclusions

We introduce an innovative node and relevant data selection paradigm to support DPA in DML environments. Our objective is to predict the most suitable nodes with the most relevant data to be engaged in DML process of ML models in light of achieving high predictive performance while avoiding redundant access over irrelevant data. Our principle is based on filtering out nodes whose data do not match the query access patterns. We provide comprehensive experimental evaluation and comparative assessment with other methods found in the literature over query workloads and datasets to assess the robustness, scalability and performance of our mechanisms. Our paradigm showcased to outperform baselines and relevant mechanisms, thus, demonstrating its ability to predict the most suitable sets of nodes tailored to DPA queries. Moreover, our DML mechanisms are proved efficient in reducing the training communication rounds and redundant access over irrelevant data ensuring streamlined interactions among selected nodes. We finally elaborated on the directions of enhancement of our paradigm towards a holistic eco-system in the realm of DPA.

CRedit authorship contribution statement

Tahani Aladwani: Writing – review & editing, Writing – original draft, Validation, Software, Resources, Methodology, Investigation, Data curation, Conceptualization. **Christos Anagnostopoulos:** Writing – review & editing, Writing – original draft, Supervision, Resources, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Kostas Kolomvatsos:** Writing – review & editing, Supervision, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work is partially funded by the EU Horizon Grant ‘Integration and Harmonization of Logistics Operations’ TRACE (#101104278).

References

- Abbott, Dean, 2014. *Applied Predictive Analytics: Principles and Techniques for the Professional Data Analyst*, first ed. Wiley Publishing.
- Aladwani, Tahani, Anagnostopoulos, Christos, Kolomvatsos, Kostas, Alghamdi, Ibrahim, Deligianni, Fani, 2023. Query-driven edge node selection in distributed learning environments. In: 39th IEEE International Conference on Data Engineering, ICDE 2023 - Workshops, Anaheim, CA, USA, April 3-7, 2023. IEEE, pp. 146–153.
- Alawani, Tahani, Parambath, Shammem, Anagnostopoulos, Christos, Deligianni, Fani, 2024. The price of labelling: A two-phase federated self-learning approach. In: *Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. In: ECML-PKDD’ 24, Springer.
- AlFahad, Saleh, Wang, Qiyuan, Anagnostopoulos, Christos, Kolomvatsos, Kostas, 2024. Task offloading in mobile edge computing using cost-based discounted optimal stopping. *Open Comput. Sci.* 14 (1), 20230115.
- Alghamdi, Ibrahim, Anagnostopoulos, Christos, Pezaros, Dimitrios P., 2021a. Data quality-aware task offloading in mobile edge computing: An optimal stopping theory approach. *Future Gener. Comput. Syst.* 117, 462–479.
- Alghamdi, Ibrahim, Anagnostopoulos, Christos, Pezaros, Dimitrios P., 2021b. Optimized contextual data offloading in mobile edge computing. In: 2021 IFIP/IEEE International Symposium on Integrated Network Management. IM, pp. 473–479.
- Anagnostopoulos, Christos, 2020. Edge-centric inferential modeling & analytics. *J. Netw. Comput. Appl.* 164, 102696.
- Anagnostopoulos, Christos, Kolomvatsos, Kostas, 2019. An intelligent, time-optimized monitoring scheme for edge nodes. *J. Netw. Comput. Appl.* 148, 102458.
- Anagnostopoulos, Christos, Triantafyllou, Peter, 2017a. Efficient scalable accurate regression queries in in-DBMS analytics. In: 2017 IEEE 33rd International Conference on Data Engineering. ICDE, pp. 559–570.
- Anagnostopoulos, Christos, Triantafyllou, Peter, 2017b. Query-driven learning for predictive analytics of data subspace cardinality. *ACM Trans. Knowl. Discov. Data* 11 (4).
- Angiulli, Fabrizio, Basta, Salvatore, Pizzuti, Claudio, 2006. Distance-based detection and prediction of outliers. *IEEE Trans. Knowl. Data Eng.* 18 (2), 145–160.
- Arjovsky, Martin, Bottou, Léon, Gulrajani, Ishaan, Lopez-Paz, David, 2020. Invariant risk minimization.
- Aubin, Benjamin, Slowik, Agnieszka, Arjovsky, Martin, Bottou, Leon, Lopez-Paz, David, 2021. Linear unit-tests for invariance discovery.
- Auer, Peter, Cesa-Bianchi, Nicolo, Fischer, Paul, 2002. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.* 47, 235–256.
- Bagdasaryan, Eugene, Veit, Andreas, Hua, Yiqing, Estrin, Deborah, Shmatikov, Vitaly, 2020. How to backdoor federated learning. In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 2938–2948.
- Bellavista, Paolo, Corradi, Antonio, Foschini, Luca, Scotece, Domenico, 2019. Differentiated service/data migration for edge services leveraging container characteristics. *IEEE Access* 7, 139746–139758.
- Boobalan, Parimala, Ramu, Swarna Priya, Pham, Quoc-Viet, Dev, Kapil, Pandya, Sharnil, Maddikunta, Praveen Kumar Reddy, Gadekallu, Thippa Reddy, Huynh-The, Thien, 2022. Fusion of federated learning and industrial Internet of Things: A survey. *Comput. Netw.* 212, 109048.
- Boulougari, Georgios, Kolomvatsos, Kostas, 2022. A QoS-aware, proactive tasks offloading model for pervasive applications. In: 2022 9th International Conference on Future Internet of Things and Cloud. FiCloud, pp. 24–31.
- Casado, Fernando E., Lema, Dylan, Criado, Marcos F., Iglesias, Roberto, Regueiro, Carlos V., Barro, Senén, 2022. Concept drift detection and adaptation for federated and continual learning. *Multimedia Tools Appl.* 1–23.
- Chen, Shengchao, Long, Guodong, Shen, Tao, Jiang, Jing, 2023. Prompt federated learning for weather forecasting: Toward foundation models on meteorological data. *arXiv preprint arXiv:2301.09152*.
- Cheng, Xu, Li, Chendan, Liu, Xiufeng, 2022. A review of federated learning in energy systems. In: 2022 IEEE/IAS industrial and commercial power system Asia. I&CPS Asia, IEEE, pp. 2089–2095.
- Deng, Yongheng, Lyu, Feng, Ren, Ju, Wu, Huaqing, Zhou, Yuezhi, Zhang, Yaoyue, Shen, Xuemin, 2021. Auction: Automated and quality-aware client selection framework for efficient federated learning. *IEEE Trans. Parallel Distrib. Syst.* 33 (8), 1996–2009.
- Feng, Siwei, Yu, Han, 2020. Multi-participant multi-class vertical federated learning. *arXiv preprint arXiv:2001.11154*.
- Ghoorchian, Saeed, Maghsudi, Setareh, 2020. Multi-armed bandit for energy-efficient and delay-sensitive edge computing in dynamic networks with uncertainty. *IEEE Trans. Cogn. Commun. Netw.* 7 (1), 279–293.
- Goetz, Jack, Malik, Kshitiz, Bui, Duc, Moon, Seungwhan, Liu, Honglei, Kumar, Anuj, 2019. Active federated learning. *arXiv preprint arXiv:1909.12641*.
- Grinberg, Nastasiya F., Orhobor, Oghenejokpeme I., King, Ross D., 2020. An evaluation of machine-learning for predicting phenotype: Studies in yeast, rice, and wheat. *Mach. Learn.* 109 (2), 251–277.
- Gu, Binbin, Kargar, Saeed, Nawab, Faisal, 2022. Efficient dynamic clustering: Capturing patterns from historical cluster evolution. In: *International Conference on Extending Database Technology*.
- Hajizadeh, Rassoul, Aghagolzadeh, Ali, Ezoji, Mehdi, 2022. Mutual neighborhood and modified majority voting based KNN classifier for multi-categories classification. *Pattern Anal. Appl.* 25 (4), 773–793.
- Hammoud, Ahmad, Mourad, Azzam, Otrok, Hadi, Dziong, Zbigniew, 2022. Data-driven federated autonomous driving. In: *International Conference on Mobile Web and Intelligent Information Systems*. Springer, pp. 79–90.
- He, Wen, He, Dazhi, Huang, Yihang, Zhang, Yizhe, Xu, Yin, Yun-feng, Guan, Zhang, Wenjun, 2020. Bandit learning-based service placement and resource allocation for mobile edge computing. In: 2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications. IEEE, pp. 1–6.
- Hong, Mannsoo, Kang, Seok-Kyu, Lee, Jee-Hyong, 2022. Weighted averaging federated learning based on example forgetting events in label imbalanced non-IID. *Appl. Sci.* 12 (12), 5806.
- Huang, Tiansheng, Lin, Weiwei, Shen, Li, Li, Keqin, Zomaya, Albert Y., 2022. Stochastic client selection for federated learning with volatile clients. *IEEE Internet Things J.*
- Idreos, Stratos, Papaemmanouil, Olga, Chaudhuri, Surajit, 2015. Overview of data exploration techniques. In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’15, Association for Computing Machinery, New York, NY, USA, pp. 277–281.
- Jiang, Yanna, Ma, Baihe, Wang, Xu, Yu, Guangsheng, Yu, Ping, Wang, Zhe, Ni, Wei, Liu, Ren Ping, 2024. Blockchain federated learning for Internet of Things: A comprehensive survey. *ACM Comput. Surv.* 56 (10), 1–37.

- Karanika, Anna, Oikonomou, Panagiotis, Kolomvatsos, Kostas, Loukopoulos, Thanasis, 2020. A demand-driven, proactive tasks management model at the edge. In: 2020 IEEE International Conference on Fuzzy Systems. FUZZ-IEEE, pp. 1–8.
- Kleinberg, Robert, Slivkins, Aleksandr, Upfal, Eli, 2019. Bandits and experts in metric spaces. *J. ACM* 66 (4).
- Kolomvatsos, Kostas, Anagnostopoulos, Christos, 2020. A probabilistic model for assigning queries at the edge. *Computing* 102 (4), 865–892.
- Kolomvatsos, Kostas, Anagnostopoulos, Christos, 2022. A proactive statistical model supporting services and tasks management in pervasive applications. *IEEE Trans. Netw. Serv. Manag.* 19 (3), 3020–3031.
- Kolomvatsos, Kostas, Anagnostopoulos, Christos, Koziri, Maria, Loukopoulos, Thanasis, 2022. Proactive & time-optimized data synopsis management at the edge. *IEEE Trans. Knowl. Data Eng.* 34 (7), 3478–3490.
- Lai, Tze Leung, Robbins, Herbert, 1985. Asymptotically efficient adaptive allocation rules. *Adv. in Appl. Math.* 6 (1), 4–22.
- Lee, Woonghee, 2022. Reward-based participant selection for improving federated reinforcement learning. *ICT Express*.
- Lee, Jaewook, Ko, Haneul, Seo, Sangwon, Pack, Sangheon, 2022. Data distribution-aware online client selection algorithm for federated learning in heterogeneous networks. *IEEE Trans. Veh. Technol.* 72 (1), 1127–1136.
- Li, Lihong, Chu, Wei, Langford, John, Schapire, Robert E., 2010. A contextual-bandit approach to personalized news article recommendation. In: Proceedings of the 19th International Conference on World Wide Web. WWW '10, Association for Computing Machinery, New York, NY, USA, pp. 661–670.
- Li, Tan, Song, Linqi, 2022. Privacy-preserving communication-efficient federated multi-armed bandits. *IEEE J. Sel. Areas Commun.* 40 (3), 773–787.
- Lin, Weiwei, Xu, Yin Hai, Liu, Bo, Li, Dongdong, Huang, Tiansheng, Shi, Fang, 2022. Contribution-based federated learning client selection. *Int. J. Intell. Syst.*
- Long, Qianyu, Anagnostopoulos, Christos, Kolomvatsos, Kostas, 2024. Enhancing knowledge reusability: A distributed multitask machine learning approach. *IEEE Trans. Emerg. Top. Comput.* 1–14.
- Long, Qianyu, Anagnostopoulos, Christos, Parambath, Shameem Puthiya, Bi, Daning, 2023. FedDIP: Federated learning with extreme dynamic pruning and incremental regularization. In: 2023 IEEE International Conference on Data Mining. ICDM, pp. 1187–1192.
- Long, Qianyu, Kolomvatsos, Kostas, Anagnostopoulos, Christos, 2022. Knowledge reuse in edge computing environments. *J. Netw. Comput. Appl.* 206, 103466.
- McMahan, H. Brendan, Moore, Eider, Ramage, Daniel, y Arcas, Blaise Agüera, 2016. Federated learning of deep networks using model averaging. *CoRR* abs/1602.05629.
- Nagalapatti, Lokesh, Mittal, Ruhi Sharma, Narayanam, Ramasuri, 2022. Is your data relevant?: Dynamic selection of relevant data for federated learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, (no. 7), pp. 7859–7867.
- Nguyen, Thanh Toan, Quoc Viet Hung, Nguyen, Nguyen, Thanh Tam, Huynh, Thanh Trung, Nguyen, Thanh Thi, Weidlich, Matthias, Yin, Hongzhi, 2024. Manipulating recommender systems: A survey of poisoning attacks and countermeasures. *ACM Comput. Surv.*
- Panagidi, Kyriaki, Anagnostopoulos, Christos, Chalvatzaras, A., Hadjiefthymides, Stathes, 2020. To transmit or not to transmit: Controlling communications in the mobile IoT domain. *ACM Trans. Internet Techn.* 20 (3), 22:1–22:23.
- Peng, Sony, Yang, Yixuan, Mao, Makara, Park, Doo-Soon, 2022. Centralized machine learning versus federated averaging: A comparison using mnist dataset. *KSII Trans. Internet Inform. Syst. (TIIS)* 16 (2), 742–756.
- Phamtoan, Dinh, Vovan, Tai, 2021. Automatic fuzzy genetic algorithm in clustering for images based on the extracted intervals. *Multimedia Tools Appl.* 80 (28–29), 35193–35215.
- Puthiya Parambath, Shameem A., Anagnostopoulos, Christos, Murray-Smith, Roderick, 2024. Sequential query prediction based on multi-armed bandits with ensemble of transformer experts and immediate feedback. *Data Min. Knowl. Discov.*
- Puthiya Parambath, Shameem, Anagnostopoulos, Christos, Murray-Smith, Roderick, MacAvaney, Sean, Zervas, Evangelos, 2021. Max-utility based arm selection strategy for sequential query recommendations. In: Balasubramanian, Vineeth N., Tsang, Ivor (Eds.), Proceedings of the 13th Asian Conference on Machine Learning. In: Proceedings of Machine Learning Research, vol. 157, PMLR, pp. 564–579.
- Rai, Sumit, Kumari, Arti, Prasad, Dilip K., 2022. Client selection in federated learning under imperfections in environment. *AI* 3 (1), 124–145.
- Ramaswamy, Sridhar, Rastogi, Rajeev, Shim, Kyuseok, 2000. Efficient algorithms for mining outliers from large data sets. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. SIGMOD '00, Association for Computing Machinery, New York, NY, USA, pp. 427–438.
- Ren, Jianji, Wang, Haichao, Hou, Tingting, Zheng, Shuai, Tang, Chaosheng, 2020. Collaborative edge computing and caching with deep reinforcement learning decision agents. *IEEE Access* 8, 120604–120612.
- Saha, Rituparna, Misra, Sudip, Chakraborty, Aishwariya, Chatterjee, Chandranath, Deb, Pallav Kumar, 2022. Data-centric client selection for federated learning over distributed edge networks. *IEEE Trans. Parallel Distrib. Syst.* 34 (2), 675–686.
- Savva, Fotis, Anagnostopoulos, Christos, Triantafillou, Peter, 2019. Aggregate query prediction under dynamic workloads. In: 2019 IEEE International Conference on Big Data (Big Data). pp. 671–676.
- Savva, Fotis, Anagnostopoulos, Christos, Triantafillou, Peter, 2020a. Adaptive learning of aggregate analytics under dynamic workloads. *Future Gener. Comput. Syst.* 109, 317–330.
- Savva, Fotis, Anagnostopoulos, Christos, Triantafillou, Peter, 2020b. SuRF: Identification of Interesting Data Regions with surrogate models. In: 2020 IEEE 36th International Conference on Data Engineering. ICDE, pp. 1321–1332.
- Savva, Fotis, Anagnostopoulos, Christos, Triantafillou, Peter, Kolomvatsos, Kostas, 2020c. Large-scale data exploration using explanatory regression functions. *ACM Trans. Knowl. Discov. Data* 14 (6).
- Shokri, Reza, Shmatikov, Vitaly, 2015. Privacy-preserving deep learning. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. pp. 1310–1321.
- Soula, Madalena, Karanika, Anna, Kolomvatsos, Kostas, Anagnostopoulos, Christos, Stamoulis, George, 2022. Intelligent tasks allocation at the edge based on machine learning and bio-inspired algorithms. *Evol. Syst.* 13 (2), 221–242.
- Torra, Vicenç, 2023. A systematic construction of non-I.I.D. data sets from a single data set: Non-identically distributed data. *Knowl. Inf. Syst.* 65 (3), 991–1003.
- Tran, Christopher, Zheleva, Elena, 2022. Improving data-driven heterogeneous treatment effect estimation under structure uncertainty. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 1787–1797.
- Tuor, Tiffany, Wang, Shiqiang, Ko, Bongjun, Liu, Changchang, Leung, Kin Kwong, 2020. Data selection for federated learning with relevant and irrelevant data at clients. *arXiv abs/2001.08300*.
- Verbraeken, Joost, Wolting, Matthijs, Katzy, Jonathan, Kloppenburg, Jeroen, Verbeelen, Tim, Rellermeyer, Jan S., 2020. A survey on distributed machine learning. *ACM Comput. Surv.* 53 (2).
- Wang, Qiyuan, Anagnostopoulos, Christos, Fornes, Jordi Mateo, Kolomvatsos, Kostas, Vrachimis, Andreas, 2023. Maintenance of model resilience in distributed edge learning environments. In: 2023 19th International Conference on Intelligent Environments. IE, pp. 1–8.
- Wang, Xiong, Ye, Jiancheng, Lui, John C.S., 2022. Decentralized task offloading in edge computing: A multi-user multi-armed bandit approach. In: IEEE INFOCOM 2022-IEEE Conference on Computer Communications. IEEE, pp. 1199–1208.
- Wang, Jianyu, Zhao, Ming, Yang, Li, 2020. Federated learning for industrial IoT: A case study on predictive maintenance. *IEEE Trans. Ind. Inform.* 16 (5), 3230–3240.
- Wu, Bochun, Chen, Tianyi, Wang, Xin, 2020. A combinatorial bandit approach to UAV-aided edge computing. In: 2020 54th Asilomar Conference on Signals, Systems, and Computers. IEEE, pp. 304–308.
- Wu, Yue, Zhang, Shuaicheng, Yu, Wencho, Liu, Yanchi, Gu, Quanquan, Zhou, Dawei, Chen, Haifeng, Cheng, Wei, 2023. Personalized federated learning under mixture of distributions. *arXiv preprint arXiv:2305.01068*.
- Yan, Yunlu, Zhu, Lei, 2023. A simple data augmentation for feature distribution skewed federated learning.
- Yang, Chien-Sheng, Pedarsani, Ramtin, Avestimehr, A. Salman, 2021. Edge computing in the dark: Leveraging contextual-combinatorial bandit and coded computing. *IEEE/ACM Trans. Netw.* 29 (3), 1022–1031.
- Yang, Jiawei, Rahardja, Susanto, Fränti, Pasi, 2019. Outlier detection: How to threshold outlier scores? In: Proceedings of the International Conference on Artificial Intelligence, Information Processing and Cloud Computing. AIIPCC '19, Association for Computing Machinery, New York, NY, USA.
- Ye, Dongdong, Yu, Rong, Pan, Miao, Han, Zhu, 2020. Federated learning in vehicular edge computing: A selective model aggregation approach. *IEEE Access* 8, 23920–23935.
- Yu, Shuai, Chen, Xu, Zhou, Zhi, Gong, Xiaowen, Wu, Di, 2020. When deep reinforcement learning meets federated learning: Intelligent multitimescale resource management for multiaccess edge computing in 5G ultradense network. *IEEE Internet Things J.* 8 (4), 2238–2251.
- Zeighami, Sepanta, Ahuja, Ritesh, Ghinita, Gabriel, Shahabi, Cyrus, 2022. A neural database for differentially private spatial range queries. *Proc. VLDB Endow.* 15 (5), 1066–1078.
- Zhou, Zhongchang, Sun, Fenggang, Chen, Xiangyu, Zhang, Dongxu, Han, Tianzhen, Lan, Peng, 2023. A decentralized federated learning based on node selection and knowledge distillation. *Mathematics* 11 (14), 3162.
- Zhu, Zirui, Sun, Lifeng, 2021. Federated trace: A node selection method for more efficient federated learning. In: 2021 IEEE International Conference on Image Processing. ICIP, IEEE, pp. 1234–1238.

Tahani Aladwani is a Research Assistant and currently pursuing her PhD in the School of Computing Science, University of Glasgow. She obtained his BSc in 2015, Taif University, and her MSc in Computer Science and Engineering from Umm Al-Qura University in 2018. Her interests include Distributed Machine Learning, Artificial Intelligence, Distributed Data Management and Knowledge Engineering in Edge Computing.





Christos Anagnostopoulos is an Associate Professor in Data Engineering and Distributed Computing, School of Computing Science, University of Glasgow. His expertise is at the intersection of distributed computing, data-centric AI, and machine learning. He has received funding by the EU/H2020, EPSRC and the industry. His is an author of over 180 refereed journals/conferences. He serves as the General Chair of the IEEE ICDCS 2025 and is Editor-in-Chief of the Open Computer Science (De Gruyter). He leads the Knowledge and Data Engineering Systems. He was MSCA Fellow Supervisor, University of Glasgow. He holds a BSc, MSc, and PhD in Computing Science, University of Athens, fellow of HEA, professional member of ACM and IEEE.



Kostas M. Kolomvatsos is an Associate Professor in Department of Informatics & Telecommunications, University of Thessaly holding BSc from Athens University of Economics and Business, and MSc and PhD in Computer Science, University of Athens. He leads the Intelligent Pervasive Systems. He was a MSCA Fellow, University of Glasgow with interest in Intelligence and Pervasive Computing Systems. He is the author of over 110 papers.