Train Global, Test Local: Privacy-preserving Learning of Cost-effectiveness in Decentralized Systems

Jovan Nikolić, Marcel Schöengens and Evangelos Pournaras

Abstract The mandate of citizens for more socially responsible information systems that respect privacy and autonomy calls for a computational and storage decentralization. Crowd-sourced sensor networks monitor energy consumption and traffic jams. Distributed ledgers systems provide unprecedented opportunities to perform secure peer-to-peer transactions using blockchain. However, decentralized systems often show performance bottlenecks that undermine their broader adoption: propagating information in a network is costly and time-consuming. Optimization of cost-effectiveness with supervised machine learning is challenging. Training usually requires privacy-sensitive local data, for instance, adjusting the communication rate based on citizens' mobility. This paper studies the following research question: How feasible is to train with privacy-preserving aggregate data and test on local data to improve cost-effectiveness of a decentralized system? Centralized machine learning optimization strategies are applied to DIAS, the *Dynamic Intelligent Aggregation Service* and they are compared to decentralized self-adaptive strategies that use local data instead. Experimental evaluation with a testing set of 2184 decentralized networks of 3000 nodes aggregating real-world Smart Grid data confirms the feasibility of a linear regression strategy to improve both estimation accuracy and communication cost, while the other optimization strategies show trade-offs.

1 Introduction

The optimization cost-effectiveness in decentralized networked systems is challenging, for instance, sensor networks making collective measurements for energy [3] or traffic monitoring [6]. Communication cost as well as convergence time required for data to propagate in a decentralized network are performance bottlenecks for their broader adoption. Instead, in centrally managed systems such as cloud com-

Jovan Nikolić and Evangelos Pournaras
Professorship of Computational Social Science, ETH Zurich, Zurich, Switzerland e-mail: {jnikolic,epournaras}@ethz.ch

Marcel Schöengens
Swiss National Supercomputing Centre, Zurich, Switzerland e-mail: schoengens@cscs.ch

puting infrastructures [22] in which performance data from each system component can be locally available or remotely accessed, a broad spectrum of optimization and machine learning techniques can be used to tune performance [10]. Hibernation of idle system components to save energy and dynamic resource allocation based on varying computational demand are some examples. Nonetheless, the data based on which optimization of cost-effectiveness is performed can be personal and privacy-sensitive. Citizens may not be willing to share their data. As a result, the data-intensive operations of traditional optimization and machine learning techniques oppose the design of decentralized systems and in particular they are not easily applicable in a privacy-sensitive application context. The resolution of this discrepancy is the subject and focus of this paper.

This paper addresses the following research questions: (i) *How to optimize cost-effectiveness in decentralized systems using supervised machine learning that exclusively uses aggregate data for training and local data for testing?* (ii) *How to aggregate training data in a privacy-preserving way?* To address these questions, a centralized machine learning approach is introduced that relies on baseline and runtime performance data aggregated over the nodes in a privacy-preserving way using differential privacy [7] or homomorphic encryption [14]. The learning capacity of linear regression and neural network classifiers is studied given the information loss by the performed aggregation. This approach is compared to decentralized self-adaptive strategies that rely instead on local data. The optimization approaches are implemented in DIAS, the *Dynamic Intelligent Aggregation Service* [18] that computes aggregation functions in a fully decentralized fashion under continuously changing input data. The accuracy of the estimations and the communication cost are optimized by the strategies. Training data are generated from simulation test runs of 2184 decentralized networks with 3000 nodes aggregating real-word Smart Grid data. The linear regression classifier is the strategy that improves both accuracy and communication cost. The neural network classifier shows a trade-off of lower communication cost for a lower accuracy, while the self-adaptive strategies show the opposite trade-off: higher accuracy at a cost of higher communication load.

The contributions of this paper are summarized as follows: (i) A new privacy-preserving framework to apply a broad spectrum of existing machine learning techniques for the optimization of cost-effectiveness in decentralized systems. (ii) The qualitative and quantitative comparison of centralized machine learning optimization strategies using aggregate data with decentralized self-adaptive strategies using local data. (iii) The enhancement of DIAS with the machine learning and self-adaptive strategies that improve accuracy and communication cost.

This paper is organized as follows: Section 2 illustrates the challenge of cost-effectiveness optimization in decentralized systems. Section 3 introduces machine learning and self-adaptive optimization strategies. Section 4 shows their applicability on a decentralized sensing scenario. Section 5 outlines the experimental settings and the results. Finally, Section 6 concludes this paper and outlines future work.

2 Cost-effectiveness Optimization in Decentralized Systems

A *decentralized system* is defined as a set of N (remote) autonomous nodes such as personal computers, smart phones, wearables or other Internet of Things devices that interact in a peer-to-peer fashion to achieve a collective goal. For instance, such devices can exchange numerical values, e.g. sensor data, to make collective measurements [18] known as in-network aggregation [5]: each device locally computes aggregation functions, for instance, the total load of the power grid [3] or the average vehicle traffic in a city [6]. Each node disseminates its values to other nodes whenever the values change or nodes join and leave the network. Without loss of generality and for a concrete illustration of the research challenge, *decentralized sensing* is the scenario studied in this paper.

The cost-effectiveness of decentralized systems is the focus of this paper. *Cost* is the amount of resources required to perform system operations. For instance, computational cost is the processing power consumed by nodes and communication cost is the number/size of exchanged messages between them. *Effectiveness* reflects on the quality of service and shows how well a decentralized system performs under a certain cost paid. For instance, the aggregation accuracy is indicator of effectiveness as it shows how 'close' the estimation of the aggregation functions is to the actual true values. Given that improvements of accuracy are a result of updates received by other nodes, i.e. input sensor data from joining nodes or updated data from connected ones, a higher communication cost can improve system effectiveness.

This paper addresses the following challenge: *self-management of trade-offs in the cost-effectiveness of decentralized systems*. A number of local parameters often regulate cost-effectiveness, for instance, the period of push-pull gossip requests [11] or Time-To-Live (TTL) in flooding [2]. In practice [20], the selection of these parameters is non-automated, system-wide (global) and made offline by system administrators/operators to control effectiveness given the available resources [21] in the deployed network infrastructure, i.e. network bandwidth or energy capacity.

Cloud computing virtualization [22] separates distributed processing and storage from centralized resource allocation. When universal access over distributed data is granted to a centralized authority, or when this authority collects these data, cost-effectiveness can be optimized online with existing (supervised) machine learning techniques [10]. However, this approach violates decentralization and raises privacy concerns over personal data that citizens may not be willing to share. This paper studies two types of privacy-preserving optimization strategies for the cost-effectiveness of decentralized systems: (i) *machine learning* and (ii) *self-adaptive* strategies. These two approaches are positioned in the design space of Table 1.

Unsupervised decentralized learning is highly complex and usually requires endogenous system redesign [1, 15]. Decentralized systems impose partial data storage and exchange. Sharing regular updates of a full feature vector over the network is inefficient and often infeasible [13] considering biases by outliers and initialization. Instead, the introduced supervised machine learning strategies (Figure 1) perform training using aggregate data and testing with local data at each node. Two aggregate data types are required for training: (i) *baseline performance* and (ii) *cost-*

Table 1: The design space of cost-effectiveness optimization in decentralized systems: Centralized optimization with local data and decentralized optimization with aggregate data are excluded (×). The former is privacy-intrusive, while the latter requires complex and costly mechanisms integration. Instead, centralized optimization with aggregate data using machine learning and decentralized optimization with local data using self-adaptive strategies are studied (★).

Optimization	**Local Data**	**Aggregate Data**
Centralized	Privacy-intrusive (×)	*Machine learning* (★)
Decentralized	*Self-adaptive strategies* (★)	Complex & costly (×)

effectiveness data. The former are used to compare system performance with an optimal performance. For instance, the actual aggregates of the sensor data can be used as a baseline against the aggregate estimations during system runtime. This comparison provides the effectiveness data. Both baseline and cost-effectiveness data are aggregated by the *optimizer* in a privacy-preserving way using differential privacy [7] or homomorphic encryption [14]. In differential privacy, nodes mask their data by adding a special noise, e.g. Laplace [7], that has the following property: when the masked data are summed up by the optimizer, the added noises cancel out and the aggregate data are revealed without revealing the individual data of the nodes. In homomorphic encryption this process is performed using cryptographic keys and is therefore more secure than differential privacy, though it usually requires key management by trusted third parties and more expensive computations [14].

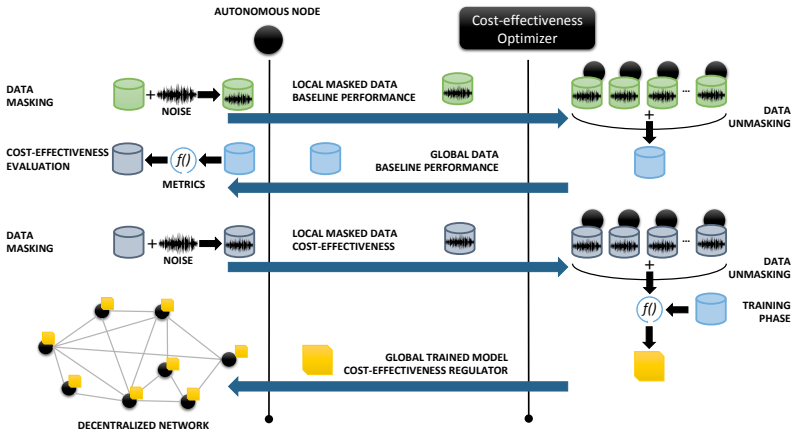


Fig. 1: The machine learning optimization approach with aggregate data.

The sequence of operations during the training phase is as follows: Nodes mask their local sensor data used as baseline performance and send them to the optimizer. The optimizer unmasks the data at an aggregate level and sends back the

aggregate sensor data used as the global baseline performance. The nodes compare the estimates of the aggregates performed in a fully decentralized way (see Section 4) with the actual baseline aggregates sent by the optimizer using an error metric that measures the aggregation accuracy (effectiveness). These data together with the measured communication cost are masked before sent to the optimizer. The optimizer unmasks the cost-effectiveness data by aggregating them and feeds them in the learning model for training. The universally generated trained model is sent back to the nodes based on which they locally regulate the communication cost at low levels while maximizing effectiveness, i.e. accuracy.

On the other hand, nodes with self-adaptive strategies perform local adjustments over parameters that control cost-effectiveness using stimuli from other nodes they interact. No training is required. Communication cost is regulated by measuring the portion of the network from which updates are received or by monitoring the stability of the aggregates as an indicator of convergence, i.e. maximal effectiveness.

3 Machine Learning vs. Self-adaptive Optimization Strategies

Assume a system parameter that controls the resource utilization at each node. Resources have a cost paid to improve system effectiveness. For instance, a higher communication rate increases the speed of aggregation accuracy. Each node is given the autonomy to regulate such system parameters to *consume* or *save* resources. Resource utilization is controlled within the *flexibility range* $[B_L, B_U]$, where B_L is a lower and B_U an upper bound. Assume as well that the nodes of the decentralized system trigger and process events at discrete time steps referred to as *epochs*, e.g. sending, and processing received messages. Nodes choose at each epoch their resource utilization within the flexibility range $[B_L, B_U]$ as the means to control cost-effectiveness. Maintaining a maximum accuracy in decentralized sensing requires utilization of the maximum number B_U of exchanged messages per epoch. However, input data may cancel out each other in aggregation, e.g. $5-2-3=0$ and therefore the update of summation does not require these data records, whose exchange adds up communication and computational overhead. Saving resources from the nodes in which these records are originated is an optimization that can improve cost-effectiveness. The rest of this section illustrates the resource utilization of the machine learning and self-adaptive strategies in the flexibility range $[B_L, B_U]$.

3.1 Machine learning optimization strategies

Each node is equipped with a classifier trained to distinguish between the two classes CONSUME and SAVE that indicate low and high cost-effectiveness respectively. During testing, CONSUME sets resource utilization to B_U, whereas SAVE to B_L. Training data are collected via (i) *simulations* or (ii) *pilot test runs*. In decentralized sensing, they concern the number of exchanged messages and the estimated aggregates among others. Formally, let $x_i \in \mathbb{R}^d$ be a d-dimensional feature vec-

tor with $i = 1, ..., n$, where n is the number of features. Labeling of the training data is performed by evaluating whether cost-effectiveness is above or below a fixed threshold, assuming it is a discrete variable. Formally, let $y_i \in \{0, 1\}$, $i = 1, ..., n$ be a label corresponding to the feature vector x_i, where 1 corresponds to CONSUME and 0 to SAVE. The data is then divided into training and validation set.

Two classifiers are studied: (i) *logistic regression* and (ii) *neural network*. The former is chosen for its non-linearity and learning efficiency since it directly learns posterior probabilities $P(y_i = 1|x_i)$. Given the optimization scope, ridge regularized logistic regression [12] is used that minimizes the following cost function:

$$\min_{\beta,C,b} \frac{1}{2}\beta^T\beta + C\sum_{i=1}^{n} \log(\exp(-y_i(x_i^T\beta + b)) + 1), \tag{1}$$

where β is a d-dimensional weight vector, b is the bias and C is the regularization strength parameter. Ridge regularization favors low values in β for irrelevant features and is chosen for its stability of the solutions and computational efficiency over L1 regularization, which favors a sparse weight vector β. Learning is performed via stochastic gradient descent. Let the learned weight vector β^* minimize Equation 1 and $\tilde{x} \in \mathbb{R}^d$ be a new feature vector from the validation or test set. The probability of $\tilde{x}$ belonging to the class CONSUME for $\tilde{y} = 1$ is given as follows:

$$P(\tilde{y} = 1|\tilde{x}) = \frac{1}{1 + \exp(-(\tilde{x}^T\beta^* + b^*))}. \tag{2}$$

If $P(\tilde{y} = 1|\tilde{x}) < 0.5$, a node saves resources by consuming B_L. Otherwise, it expands the resource consumption to B_U. The logistic regression classifier is generalized to a *Multi-Layer Perceptron* (MLP) neural network, by adding h hidden layers, each with l_q, $q = 1, ...h$ hidden nodes. The activation function in the hidden layer is the ReLU function, chosen for its simplicity, ease of computation and reduced likelihood of vanishing gradients [9]. Learning is performed via backpropagation.

Both classifiers are validated on the validation set and then tested during the decentralized system runtime. The trained classifier is integrated into each node as a black box. It receives as input the locally estimated aggregates and provides as output the resource utilization that is the communication rate at each epoch.

3.2 Self-adaptive strategies

The self-adaptive strategies are data-independent heuristics that predict cost-effectiveness by monitoring local system parameters, for instance, counting the number of nodes with which sensor data are exchanged. This measurement encodes both the (i) communication cost and (ii) aggregation accuracy. As the counter increases, a larger portion of the network receives the latest sensor data and estimations are more accurate. Therefore this counter can regulate resource allocation: a node joining the network or having new sensor data to share begins with a communication rate of B_U to minimize the time operating with low accuracy. As the number of data exchanges

with different nodes increases, the communication rate decreases to save resources and eventually becomes B_L when data are disseminated to all nodes of the network.

Formally, let $r_j^{(t)} \in [0,1]$ define the relative remaining communication cost that is required for a node j at epoch t to exchange the sensor data with all nodes in the network. Next assume a function $f(r_j^{(t)})$ that calculates the allocated resources such that $B_L \leq f(r_j^{(t)}) \leq B_U$. This paper studies the following functions:

Table 2: Functions used by the self-adaptive strategies.

Function Name	Function	α
Linear	$B_L + \alpha \cdot r \cdot B_U$	$\frac{B_U - B_L}{B_U}$
Exponential	$B_L - 1 + e^{\alpha \cdot r \cdot B_U}$	$\frac{\ln(B_U - B_L + 1)}{B_U}$
Square Root	$B_L + \sqrt{\alpha \cdot r \cdot B_U}$	$\frac{(B_U - B_L)^2}{B_U}$
Logistic Regression	$B_U - \frac{B_U - B_L}{1 + 0.001 \cdot e^{14 \cdot r}}$	Parameters smoothly covering $[B_L, B_U]$

These self-adaptive strategies do not require data labeling and offline training. On the contrary, they assume apriori an exact relation between the relative remaining communication cost and the allocated resources.

4 Applicability on Decentralized Sensing

The optimization strategies are applied on DIAS[1], the *Dynamic Intelligent Aggregation Service*. DIAS is a generic and highly dynamic service for fully decentralized and real-time computations of aggregation functions: Each DIAS node can share (data provider) and aggregate (data consumer) sensor data. DIAS can adapt and self-correct computations to improve accuracy under changing sensor data [18] or nodes joining and leaving the network [17, 16]. A large family of aggregation functions, e.g. summation, average, maximum, minimum, count, standard deviation, etc., can be computed without any change in the core distributed algorithm.

DIAS nodes discover each other and disseminate their sensor data updates via the gossip-based peer sampling service [11]. Each node periodically updates its partial *view* that is a list of limited size c with randomly populated descriptors of other nodes containing the IP address, the port number and other application-level information. The HEALING and SWAPPING gossip parameters determine the features of the discovered nodes: the latest nodes joined the network vs. random ones.

Real-time aggregation of highly violatile data is feasible in DIAS using the model of *possible states*. Let $P_j = (p_{j,u})_{u=1}^{k}$ be a sequence of k possible states generated by a node j using, for instance, historical sensor data [19]. For instance, smart meter power data can be abstracted by numerical representations corresponding to the low, medium and high consumption level. Moreover, let $s_j^t \in P_j$ be the local *selected state* of node j at epoch t that is the input in the aggregation functions of other

[1] Available af `http://dias-net.org` (last access: May 2018)

remote nodes. Sensor data exchanges are performed within an *aggregation session* between an *aggregator* and a *disseminator*, locally or in two remote nodes. A node has a disseminator, an aggregator or both. The latter scenario is the most demanding one in terms of computational and communication resources. The disseminator initiates the aggregation session and sends its selected state s_j to the aggregator. The aggregator classifies it as a new input value, i.e. fist performed aggregation session, or as a replacing input value, i.e. an earlier aggregation session has been performed with another outdated selected state. The aggregator completes the aggregation session by sending an acknowledgment and the outcome back to the disseminator.

Aggregation accuracy improves by counting a new selected state or updating an outdated one. Any other scenario, i.e. duplicate selected states, is excluded using a distributed memory system based on probabilistic data structures, the *bloom filters* [4]. The memory system consists of two nested bloom filter layers as shown in Figure 2: (i) the *interactions layer* verifies whether an aggregation session between an aggregator and a disseminator has been earlier performed while (ii) the *data layer* validates which latest selected state has been aggregated. Disseminators use the memory system to determine whether an aggregation session should be performed with an aggregator sampled from the gossiping service. The exact algorithms are out of the scope of this paper and can be found in earlier work [18, 17, 16].

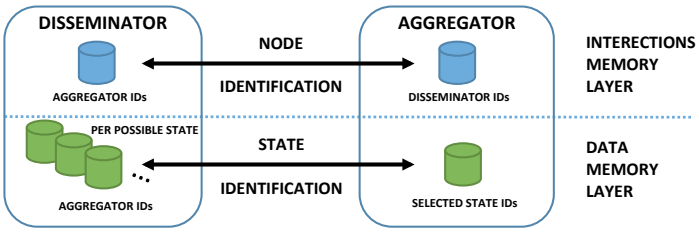


Fig. 2: The DIAS distributed memory systems with two nested layers of bloom filters: (i) *Interaction layer* - tracks the involved disseminators and aggregators of the aggregation sessions. (ii) *Data layer* - tracks the exchange of selected states.

Cost is measured by the total number of exchanged messages per epoch originated by the aggregation sessions. The effectiveness of DIAS at each epoch t is measured by the average estimation error $\tilde{S}_j^{(t)}$ of summation over N nodes, assuming each node has an aggregator:

$$\epsilon^{(t)} = \frac{1}{N} \sum_{j=1}^{N} \frac{|\tilde{S}_j^{(t)} - \sum_{v=1}^{N} s_v^{(t)}|}{\sum_{v=1}^{N} s_v^{(t)}}, \tag{3}$$

where $\tilde{S}_j^{(t)}$ is the estimated sum by each node j and $\sum_{v=1}^{N} s_v^{(t)}$ is the actual sum of all selected states. The aggregation function of summation is chosen as it is the most sensitive one to changes in cost-effectiveness, i.e. possible states. Recall from Figure 1 that $\sum_{v=1}^{N} s_v^{(t)}$ is the baseline performance calculated via differential privacy schemes [7] to prevent leaking of the individual selected states to a third party.

The number of initiated aggregation sessions $b_j^{(t)}$ for each node j and epoch t, i.e. communication rate, is the system parameter (control variable) that regulates cost-effectiveness and the one controlled by the optimization strategies. Instead of fixing $b_j^{(t)}$ for all nodes, the strategies vary $b_j^{(t)}$ in the flexibility range $[B_L, B_U]$, where $B_L = 0$ determines no performed aggregation sessions (saving resources), whereas $B_U = c$ improves accuracy by maximizing[2] the number of aggregation sessions.

4.0.1 Optimization stratagies in decentralized sensing

Machine learning strategies: Training is performed by varying in multiple test runs (i) the number of aggregation sessions and (ii) the possible states as well as the selected state of the nodes (different datasets). This process generates a 5-dimensional vector with the following metrics: (i) test run identifier, (ii) epoch number, (iii) number of aggregation sessions, (iv) average estimation of the sum over all nodes and (v) slope of this sum based on a time window of 5 epochs. The average estimation of the sum at epoch t is calculated as $\frac{1}{N}\sum_{j=1}^{N}\tilde{S}_j^{(t)}$, where $\tilde{S}_j^{(t)}$ is the estimation of the sum by node j at epoch t. The slope of the sum quantifies the dynamics of the selected states, i.e. changes performed during a test run:

$$\delta^{(t)} = \frac{Var[\frac{1}{N}\sum_{j=1}^{N}\tilde{S}_j^{(\tau)}|\tau = t, t-1, ..., t-4]}{Var[\tau|\tau = t, t-1, ..., t-4]}. \tag{4}$$

The optimizer labels each feature vector using the average estimation error (Equation 3). If the error is below a fixed threshold, it is labeled as SAVE, otherwise, CONSUME. These labeled data are then used to train the classifier.

During the testing phase, each node j at epoch $t-1$ uses the classifier to determine the number of aggregation sessions $b_j^{(t)}$ at epoch t. No interactions are required between nodes and optimizer. Autonomy and decentralization are preserved. The test feature vector generated online during system operation contains the local sum estimation $\tilde{S}_j^{(t)}$ and the slope of this estimate instead of the average sum estimation and its slope over all nodes. The number of aggregation sessions are set to $b_j^{(t+1)} = B_U$ if the classifier determines CONSUME or $b_j^{(t+1)} = B_L$ if SAVE.

Self-adaptive strategies: At each epoch $t-1$, a disseminator j counts the aggregators $Q_j^{(t-1)}$ that have aggregated its current selected state $s_j^{(t-1)}$. If the selected state changes, the counter sets back to zero. The count aggregation function of DIAS provides an estimate of the total number of aggregators $\tilde{N}_j^{(t-1)}$ at each epoch $t-1$ that is N if all nodes have an aggregator. The relative remaining aggregation sessions $r_j^{(t-1)}$ initiated by a disseminator j at epoch t are calculated as follows:

$$b_j^{(t)} = f(r_j^{(t-1)}) = 1 - \frac{Q_j^{(t-1)}}{\tilde{N}_j^{(t-1)}}, \tag{5}$$

[2] $B_U = c$ assumes that gossiping updates the view at least once per epoch.

where $r_j^{(t-1)} \in [0,1]$ is fed in one of the functions of Table 2. The higher the rate of changes in the selected state, the higher the $r_j^{(t-1)}$.

5 Experimental Evaluation

DIAS and the peer sampling service on which DIAS relies on are implemented[3] in Java using the Protopeer distributed prototyping toolkit [8]. Training data are collected by deploying simulation test runs in the CSCS supercomputing infrastructure[4], while DIAS is deployed in the Euler HPC cluster infrastructure[5] of ETH Zurich for execution of the testing phase and evaluation of the cost-effectiveness.

Real-world data from the *Electricity Customer Behavior Trial* project[6] are used for aggregation by DIAS. The data were collected in 2009 and 2010 (364 days) and concern the electricity consumption of 3000 residential consumers. They contain 30-minutes records of the power consumption. The possible states of each disseminator are the cluster centroids computed by *k*-means using Weka[7], with $k = 5$. The dataset is randomly divided into two sets: 80% for training and 20% for testing.

The following parameters are used for the simulation test runs: 3000 nodes, 800 epochs, view size $c = 50$, a HEALING of 1 and a SWAPPING of 24. Each experiment runs with a fixed maximum number of aggregation sessions among the following values in different experiments: $\{5, 15, 20, 30, 40, 50\}$. Therefore, a total number of 6 sessions $*$ 364 days $= 2184$ experiments are performed for training.

Each feature vector is labeled with 1 (CONSUME) if $\epsilon^{(t)} \geq 0.1$ and with 0 (SAVE) otherwise. The labeled feature vectors are fed into the two classifiers from *scikit-learn*[8]. The logistic regression uses the 'sag' solver and regularization strength of $C = 0.5$. The multilayer perceptron has $h = 2$ hidden layers with $l_1 = 6$ and $l_2 = 3$ hidden nodes. Learning rate is constant and equal to 0.001. Training is performed without regularization using the 'lbfgs' solver from the quasi-Newton family methods. Both classifiers are validated[9] on the training set. The trained classifiers are then integrated into each DIAS node by importing the weight vectors and biases.

During testing phase for Day 199, the flexibility range of $[B_L = 10, B_U = 40]$ is used by each node j to determine the number of aggregation sessions $b_j^{(t)}$ for

[3] `https://github.com/epournaras/DIAS` and `https://github.com/epournaras/PeerSamplingService` (last access: May 2018)

[4] `https://www.cscs.ch` (last access: May 2018)

[5] `https://scicomp.ethz.ch/wiki/Euler` (last access: May 2018)

[6] `http://www.ucd.ie/issda/data/commissionforenergyregulationcer/` (last accessed: May 2018)

[7] `https://www.cs.waikato.ac.nz/ml/weka/` (last access: May 2018)

[8] `http://scikit-learn.org/stable/` (last access: May 2018)

[9] Linear regression has an average *precision*, *recall*, *f1-score* of 0.8 and 0.96 for neural network. 273662 occurrences appear for SAVE and 123557 for CONSUME in linear regression. The respective occurences are 274108 and 123111 for neural network. Validation metrics documentation: `http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html` (last access: May 2018).

both machine learning and self-adaptive strategies. The range $[B_L = 20, B_U = 40]$ is also evaluated for the machine learning strategies. Figure 3a illustrates how $b_j^{(t)}$ is determined by the self-adaptive strategies and also shows the actual summation estimated by each node during the testing phase.

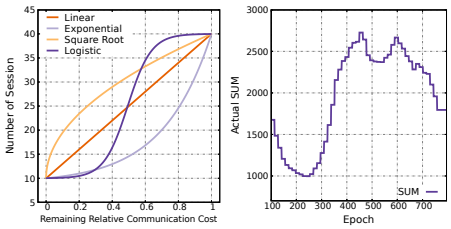


Fig. 3: Testing phase: Calculation of the aggregation sessions by the self-adaptive strategies and the actual summation estimated by each node.

The strategies are evaluated by comparing their cost-effectiveness against a fixed number of aggregation sessions for all nodes. This number is calculated by the average number of aggregation sessions observed in the compared optimization strategy. The former case is indexed with O and the latter with F. Comparison measurements are made in terms of *performance improvement* defined at each epoch t as follows:

$$g(v)^{(t)} = v_{\mathsf{F}}^{(t)} - v_{\mathsf{O}}^{(t)}, \tag{6}$$

where v is the communication cost m or the estimation error ϵ. Similarly, the *mean relative performance improvement* over the DIAS runtime is introduced as follows:

$$\overline{g}(v) = \frac{1}{T_{max} - T_{min} + 1} \sum_{t=T_{min}}^{T_{max}} \frac{g(v)^{(t)}}{v_{\mathsf{F}}^{(t)}} \cdot 100\%, \tag{7}$$

where v is the communication cost m or the estimation error ϵ and $T_{max} = 800$, $T_{min} = 101$, excluding the first 100 epochs used for system bootstrapping.

5.1 Experimental results

Figure 4 illustrates the mean relative improvement of the optimization strategies. The following observations can be made: The logistic regression with the narrower flexibility range of $[B_L = 20, B_U = 40]$ is the only strategy that achieves to decrease both the estimation error by 7.15% and the communication cost by 4.96%. The other strategies show two opposing trade-offs. The neural networks increase the estimation error by 12.35% on average but decrease communication cost by 7.37% respectively. In contrast, the self-adaptive mechanisms decrease the estimation error by 8.18% on average but increase communication cost by 11.25% respectively.

Figure 5 illustrates how performance improvement varies over runtime. The following observations can be made for the estimation error in Figure 5a and 5b: The

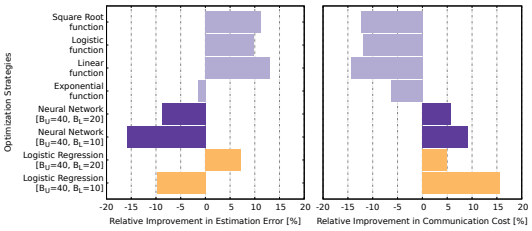
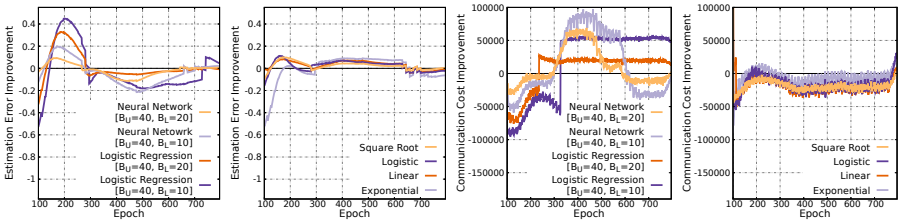


Fig. 4: Mean relative performance improvement of the optimization strategies.

machine learning strategies and especially the logistic regression maximize the decrease of the estimation error during epochs 131 to 281 during which a significant change in summation is observed (Figure 3). The self-adaptive strategies manage to maintain the improvement during the next epochs in contrast to the machine learning strategies that fall into a performance deterioration during epochs 281 to 800.



(a) Estimation error, machine learning strategies

(b) Estimation error, self-adaptive strategies

(c) Communication cost, machine learning strategies

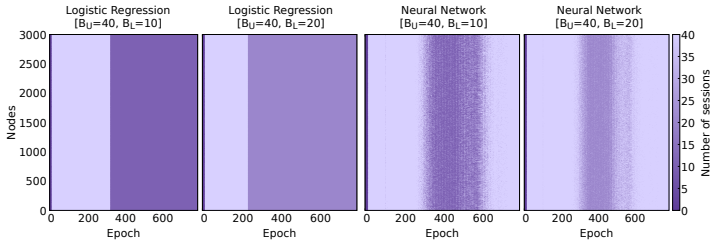
(d) Estimation error, self-adaptive strategies

Fig. 5: Performance improvement of the optimization straegies over system runtime.

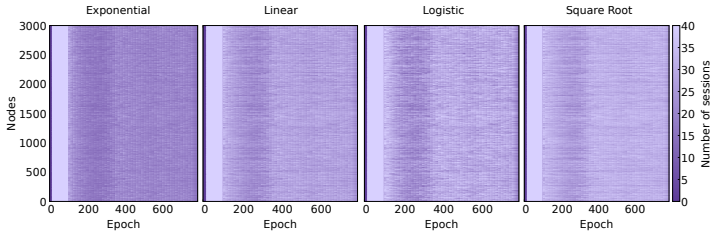
Figure 5c and 5d show the improvement of the optimization strategies in communication cost along with Figure 6 that shows the number of aggregation sessions for each node over runtime. The machine learning strategies and especially the logistic regression consume a high number of messages during the first epochs in which large changes in the summation are observed. In this way they manage to decrease the estimation error during this period. During epochs 327 to 800, all machine learning strategies reduce communication cost and as a consequence they do not capture the increase of summation. This justifies the negative improvement in the estimation error. The self-adaptive strategies decrease the estimation error by sacrificing communication cost, with the exponential function providing the best trade-off.

6 Conclusion and Future Work

This paper concludes that the optimization of cost-effectiveness in decentralized systems is feasible using machine learning without violating privacy. This is made possible by introducing a novel training scheme that relies entirely on aggregate



(a) The machine learning strategies



(b) The self-adaptive strategies

Fig. 6: The selections of the aggregations sessions by the optimization strategies.

data collected in a privacy-preserving way, while testing is locally performed in each node of the network using local data. A linear regression classifier manages to improve both accuracy and communication cost in an application scenario of decentralized sensing using real-world Smart Grid data, while the neural network classifier shows a trade-off: lower accuracy but higher savings in communication load. The opposite trade-off is observed in the self-adaptive strategies.

Future work includes the linking of the validation performance with the testing performance of the decentralized network as well as performance comparisons with with the other two design options of Table 1. The feasibility of deep learning and other machine learning techniques empowered by cooperative data forwarding strategies is subject of future work as well.

Acknowledgements This work is supported by the European Community's H2020 Program under the scheme 'ICT-10-2015 RIA', grant agreement #688364 'ASSET: Instant Gratification for Collective Awareness & Sustainable Consumerism' (`http://www.asset-consumerism.eu`).

References

1. Azimi, R., Sajedi, H.: A decentralized gossip based approach for data clustering in peer-to-peer networks. Journal of Parallel and Distributed Computing (2018)
2. Barjini, H., Othman, M., Ibrahim, H., Udzir, N.I.: Shortcoming, problems and analytical comparison for flooding-based search techniques in unstructured p2p networks. Peer-to-Peer Networking and Applications **5**(1), 1–13 (2012)
3. Bhattarai, B., de Cerio Mendaza, I.D., Myers, K.S., Bak-Jensen, B., Paudyal, S.: Optimum aggregation and control of spatially distributed flexible resources in smart grid. IEEE Transactions on Smart Grid (2017)

4. Broder, A., Mitzenmacher, M.: Network applications of bloom filters: A survey. Internet mathematics **1**(4), 485–509 (2004)
5. Dagar, M., Mahajan, S.: Data aggregation in wireless sensor network: a survey. International Journal of Information and Computation Technology **3**(3), 167–174 (2013)
6. Dietzel, S., Petit, J., Kargl, F., Scheuermann, B.: In-network aggregation for vehicular ad hoc networks. IEEE communications surveys & tutorials **16**(4), 1909–1932 (2014)
7. Dwork, C., Roth, A., et al.: The algorithmic foundations of differential privacy. Foundations and Trends® in Theoretical Computer Science **9**(3–4), 211–407 (2014)
8. Galuba, W., Aberer, K., Despotovic, Z., Kellerer, W.: Protopeer: a p2p toolkit bridging the gap between simulation and live deployement. In: Proceedings of the 2nd International Conference on Simulation Tools and Techniques, p. 60. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) (2009)
9. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pp. 315–323 (2011)
10. Hameed, A., Khoshkbarforoushha, A., Ranjan, R., Jayaraman, P.P., Kolodziej, J., Balaji, P., Zeadally, S., Malluhi, Q.M., Tziritas, N., Vishnu, A., et al.: A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. Computing **98**(7), 751–774 (2016)
11. Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.M., Van Steen, M.: Gossip-based peer sampling. ACM Transactions on Computer Systems (TOCS) **25**(3), 8 (2007)
12. Lee, A., Silvapulle, M.: Ridge estimation in logistic regression. Communications in Statistics-Simulation and Computation **17**(4), 1231–1257 (1988)
13. Luo, L., Liu, M., Nelson, J., Ceze, L., Phanishayee, A., Krishnamurthy, A.: Motivating in-network aggregation for distributed deep neural network training. In: Workshop on Approximate Computing Across the Stack (2017)
14. Parmar, P.V., Padhar, S.B., Patel, S.N., Bhatt, N.I., Jhaveri, R.H.: Survey of various homomorphic encryption algorithms and schemes. International Journal of Computer Applications **91**(8) (2014)
15. Pilgerstorfer, P., Pournaras, E.: Self-adaptive learning in decentralized combinatorial optimization: a design paradigm for sharing economies. In: Proceedings of the 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, pp. 54–64. IEEE Press (2017)
16. Pournaras, E., Nikolić, J.: On-demand self-adaptive data analytics in large-scale decentralized networks. In: Network Computing and Applications (NCA), 2017 IEEE 16th International Symposium on, pp. 1–10. IEEE (2017)
17. Pournaras, E., Nikolic, J.: Self-corrective dynamic networks via decentralized reverse computations. In: Proceedings of the 14th International Conference on Autonomic Computing (ICAC 2017) (2017)
18. Pournaras, E., Nikolic, J., Omerzel, A., Helbing, D.: Engineering democratization in internet of things data analytics. In: Proceedings of the 31st IEEE International Conference on Advanced Information Networking and Applications-AINA-2017. IEEE (2017)
19. Pournaras, E., Vasirani, M., Kooij, R.E., Aberer, K.: Measuring and controlling unfairness in decentralized planning of energy demand. In: Energy Conference (ENERGYCON), 2014 IEEE International, pp. 1255–1262. IEEE (2014)
20. Ramassamy, C., Fouchal, H.: A decision-support tool for wireless sensor networks. In: Communications (ICC), 2014 IEEE International Conference on, pp. 7–11. IEEE (2014)
21. Wu, J., Jia, Q.S., Johansson, K.H., Shi, L.: Event-based sensor data scheduling: Trade-off between communication rate and estimation quality. IEEE Transactions on automatic control **58**(4), 1041–1046 (2013)
22. Xiao, Z., Song, W., Chen, Q.: Dynamic resource allocation using virtual machines for cloud computing environment. IEEE transactions on parallel and distributed systems **24**(6), 1107–1117 (2013)