Resource-Constrained On-Device Learning by Dynamic Averaging

Lukas Heppe[1], Michael Kamp[2], Linara Adilova[3,4], Danny Heinrich[1], Nico Piatkowski[3], and Katharina Morik[1]

[1] TU Dortmund `<name>.<surname>@tu-dortmund.de`
[2] Monash University `michael.kamp@monash.edu`
[3] Fraunhofer IAIS `<name>.<surname>@iais.fraunhofer.de`
[4] Fraunhofer Center for Machine Learning

Abstract. The communication between data-generating devices is partially responsible for a growing portion of the world's power consumption. Thus reducing communication is vital, both, from an economical and an ecological perspective. For machine learning, on-device learning avoids sending raw data, which can reduce communication substantially. Furthermore, not centralizing the data protects privacy-sensitive data. However, most learning algorithms require hardware with high computation power and thus high energy consumption. In contrast, ultra-low-power processors, like FPGAs or micro-controllers, allow for energy-efficient learning of local models. Combined with communication-efficient distributed learning strategies, this reduces the overall energy consumption and enables applications that were yet impossible due to limited energy on local devices. The major challenge is then, that the low-power processors typically only have integer processing capabilities. This paper investigates an approach to communication-efficient on-device learning of integer exponential families that can be executed on low-power processors, is privacy-preserving, and effectively minimizes communication. The empirical evaluation shows that the approach can reach a model quality comparable to a centrally learned regular model with an order of magnitude less communication. Comparing the overall energy consumption, this reduces the required energy for solving the machine learning task by a significant amount.

1 Introduction

Today more and more data is generated by physically distributed sources, e.g., smartphones, sensors, and IoT devices. Performing machine learning on this data not only poses severe challenges on the bandwidth, on storing and processing it, but also requires enormous amounts of energy: currently, the world communicates around $20 \cdot 10^{10}$ GB per month [11] with a power consumption of around 0.3kWh/GB [18], resulting in a total energy consumption of 6TWh per month. In comparison, the largest nuclear plant in the US, the R.E. Gina reactor in Arizona, generates around 0.39TWh per month [1], so more than 15 such reactors are needed just to power the communication. With an estimated number

of $5 \cdot 10^{10}$ connected devices by the end of 2020 [14], the amount of communication will grow substantially, with some of them (e.g., autonomous vehicles or airplanes) communicating up to 5GB per second [19]. Thus, machine learning on this data could become responsible for a large portion of the world's power consumption.

In order to reduce the communication load, models can be trained locally and only model parameters are centralized periodically [13] or dynamically [7, 10]. However, this approach requires sufficient computation power at the local data sources - this is usually available on smartphones, but not necessarily on sensors or IoT devices. Most sensors or IoT devices could be fit with efficient, low-power processors which typically only have integer processing capabilities - no floating point unit. Recently, it was shown that learning exponential families can be performed on such devices using only integer and bit operations [17]. Scaling this learning to the internet of things requires to implement communication-efficient distributed learning on these devices, too.

This work proposes a resource- and communication-efficient distributed learning approach for exponential families that uses only integer and bit operations. The key idea is to only communicate between local devices if their model is sufficiently different from the global mean, implying that it has learned truly novel information. Hence, we adapt dynamic averaging [8] to require only integer operations. In this approach, each device checks a sufficient local condition and only communicates if it is violated. In case of a violation, model parameters are centralized and averaged into a joint model, which is redistributed to the devices. Setting the maximum allowed divergence between a model and the global mean allows users to the trade-off between communication-efficiency and joint model quality.

We theoretically analyze this approach and provide guarantees on the loss and bounds on the amount of communication. We show empirically that using resource-constrained dynamic averaging on integer exponential families allows to reach a model performance close to full floating point models and reduces the required communication substantially.

Related work Several works have been published, minimizing the resource-consumptions in federated learning environments. However, most publications already consider smartphones as resource-constrained environments, while we go one step further and focus on ultra-low-power hardware without access to floating-point-units. A similar work has been done by Piatkowski [15], who also considered resource-constrained family models for a distributed learning task. Albeit, the reduced resource-consumption is based on parameter-sparsification and one-shot averaging. Besides, the model aggregates reside in regular model space and thus cannot be applied on ultra-low-power hardware. Alternative methods, proposed by Wang et. al., focus on selecting a trade-off between global aggregation and local updates, while we follow a distinct approach based on dynamic conditions. A survey on federated learning concepts and the challenges can be found in [12].

2 Resource-Constrained Exponential Family Models

Probabilistic graphical models form a subset of machine learning and combine graph with probability theory [20]. They are used to model complex probability distributions, which can be utilized to solve a variety of tasks. The following subsections will introduce the notation and background of graphical models in general as well as highlight the specifics for resource-constrained models.

2.1 Undirected graphical models

Let $G = (V, E)$ be an undirected graph with $|V| = n$ vertices which are connected by the edges $(s, t) \in E \subseteq V \times V$. A clique $C \subseteq V$ is formed by some fully connected subset of the vertices: $\forall\ u, v \in C, u \neq v : (u, v) \in E$. The maximal cliques of the graph are those, who are not contained in any larger clique. We denote the set of all maximal cliques by $\mathcal{C}$. In addition, let $\boldsymbol{X} = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_p)^\top$ denote some random vector where each variable $\boldsymbol{X}_i$ can take values of a discrete set $\mathcal{X}_i$. In turn the vector can take values of the cross product from each variable $\mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_p$. Besides, we allow the indexing of $\boldsymbol{X}$ and $\mathcal{X}$ by subsets like cliques, e.g., $\boldsymbol{X}_C$ and $\mathcal{X}_C$. Specific assignments $\boldsymbol{x}_C$ to those random variables are denoted by bold lowercase letters.

An undirected graphical model represents the joint distribution of $\boldsymbol{X}$ by exploiting conditional independencies between the variables. We model those using a graph, if $(s, t) \notin E \implies \boldsymbol{X}_s \perp\!\!\!\perp \boldsymbol{X}_t \mid \boldsymbol{X} \setminus \{\boldsymbol{X}_s, \boldsymbol{X}_t\}$. Thus, the graph encodes the conditional independence structure of the distribution. If we introduce potential functions $\psi_C : \mathcal{X}_C \mapsto \mathbb{R}_+$ for each maximal clique of the graph, then according to the Hammersley-Clifford-theorem [5] the density factorizes as follows

$$\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C) \ , \tag{1}$$

where $Z = \sum_{\boldsymbol{x} \in \mathcal{X}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$ acts as normalizer in order to ensure that $\mathbb{P}$ is a valid probability distribution. By utilizing probabilistic inference algorithms like belief propagation, this distribution can be used for a variety of tasks like querying marginal- or conditional probabilities as well as computing (conditional) maximum-a-posteriori estimates, which, in turn, can be used for predictive tasks.

2.2 From regular to resource-constrained models

Let $C \in \mathcal{C}$ be a maximal clique of G, $\boldsymbol{\theta}_C \in \mathbb{R}^{|\mathcal{X}_C|}$ a parameter vector and $\phi : \mathcal{X}_C \mapsto \{0, 1\}^{|\mathcal{X}_C|}$ some feature function or sufficient statistic, which maps assignments of the variables to one-hot-encoded vectors. By defining $\psi_C(\boldsymbol{x}_C) = \exp \langle \boldsymbol{\theta}_C, \phi_C(\boldsymbol{x}_C) \rangle$, concatenating all $\boldsymbol{\theta}_C$ in $\boldsymbol{\theta}$ as well as all ϕ_C in ϕ, the joint distribution can be represented as the canonical exponential family [20]

$$\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{X} = \boldsymbol{x}) = \exp(\langle \boldsymbol{\theta}, \phi(\boldsymbol{x}) \rangle - A(\boldsymbol{\theta})) \ , \tag{2}$$

with $A(\boldsymbol{\theta}) = \log Z(\boldsymbol{\theta})$, which contains many well known distributions like the gamma, normal, and exponential distribution. Despite its compactness, in contrast to other models like neural networks with millions of parameters, the model cannot be evaluated on ultra low power devices since it requires the presence of floating point units.

Driven by the cheap availability of ultra low power hardware, the increased power consumption of machine learning models and the trend to push those towards the edge, Piatkowski [16] developed a variation of the regular exponential family, which is capable of running on devices, that only have access to integer arithmetic units. By replacing the base in equation 2 from e to 2 and restricting the parameters to be a subset of $\mathbb{N}_{\leq k}$ with fixed word size k, the resource constrained exponential family is defined as follows

$$P_{\boldsymbol{\theta}}(\mathbf{X} = \boldsymbol{x}) = 2^{\langle \boldsymbol{\theta}, \phi(\boldsymbol{x}) \rangle - A(\boldsymbol{\theta})}$$

with

$$A(\boldsymbol{\theta}) = \log_2 \sum_{\boldsymbol{x} \in \mathcal{X}} 2^{\langle \boldsymbol{\theta}, \phi(\boldsymbol{x}) \rangle - A(\boldsymbol{\theta})} \quad .$$

The availability of specialized integer versions of the inference- and optimization-algorithms does allow not only the application, but also the learning of these models directly on local devices. Moreover, Piatkowski has proven [17], that despite the limitations the models still provide theoretical guarantees on the quality.

Learning The parameters $\boldsymbol{\theta}$ of the distribution $\mathbb{P}_{\boldsymbol{\theta}}$ are estimated using a (regularized) maximum likelihood estimation. Suppose we are given a dataset $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ with n samples, the negative average log-likelihood is defined as follows

$$\ell(\boldsymbol{\theta}; \mathcal{D}) = \log_2 A(\boldsymbol{\theta}) - \left\langle \boldsymbol{\theta}, \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{x} \in \mathcal{D}} \phi(\boldsymbol{x}) \right\rangle$$

Setting $\hat{\boldsymbol{\mu}} = \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{x} \in \mathcal{D}} \phi(\boldsymbol{x})$ the partial derivative of ℓ is as follows

$$\frac{\partial \ell(\boldsymbol{\theta}; \mathcal{D})}{\partial \boldsymbol{\theta}_i} = \mathbb{E}_P[\phi(\boldsymbol{x})_i] - \hat{\boldsymbol{\mu}}_i \quad ,$$

which is just the difference between the empirical and the model's distribution. The model's distribution is computed using the BitLength-Propagation-algorithm [17], which returns the probabilities as quotients a/b to avoid floating point numbers. Likewise for $\hat{\boldsymbol{\mu}}$, we store the raw counts as well as the cardinality of the dataset as integers. Using a proximal block coordinate descent method, the parameters for each clique are updated by either increasing or decreasing the current value by one. In-depth details on those specialized integer algorithms can be found in [17]. If the samples $\boldsymbol{x}$ arrive as stream, e.g., sensor readings, the ϕ's and example's counters will be accumulated iteratively and used for a gradient step. This is showcased in algorithm 1.

Algorithm 1 Integer Learning Algorithm

Input: Graph $G = (V, E)$, bytes $k \in \mathbb{N}$
Initialization:
  local sufficient statistics $\hat{\boldsymbol{\mu}}_1^1, \ldots, \hat{\boldsymbol{\mu}}_1^m \leftarrow \mathbf{0}$
  local example counter $c_1^1, \ldots, c_1^m \;\;\leftarrow 0$
Round t at learner i:
  observe $S_t^i \subset \mathcal{X}$
  for $\boldsymbol{x} \in S_t^i$ **do**
    $\hat{\boldsymbol{\mu}}_t^i \leftarrow \hat{\boldsymbol{\mu}}_t^i + \phi(\boldsymbol{x})$
  $c_t^i \leftarrow c_t^i + \left| S_t^i \right|$
  $\boldsymbol{\mu}_t^i \leftarrow \texttt{BitLengthBP()} ;$ `// Compute models distribution`
  $\nabla \leftarrow \texttt{Grad}(\boldsymbol{\mu}_t^i, \boldsymbol{\mu}_t^i, c_t^i) ;$ `// Compute grad using Int-Prox`
  $\boldsymbol{\theta}_t^i \leftarrow \texttt{ApplyGrad}(\boldsymbol{\theta}_{t-1}^i, \nabla)$
  return $\boldsymbol{\theta}_t^i$

---

3 Distributed Learning of Integer Exponential Families

The integer exponential families described above can be trained on a data-generating device using only integer computations. The goal of distributed learning is to jointly train a model across multiple devices. That is, we assume a set of $m \in \mathbb{N}$ local devices, denoted **learners**, learning a joint task defined by a target distribution $\mathcal{D} : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}_+$. The learners obtain local samples over time. For simplicity, we assume rounds $t = 1, 2, \ldots$ where in each round, each learner $l \in [m]$ obtains a local dataset $S_t^l \subset \mathcal{X} \times \mathcal{Y}$ drawn iid. from $\mathcal{D}$.

The most straight-forward approach to solve this task is to compute local data summaries $\widehat{\boldsymbol{\mu}}_t^l$ on all observed data $\bigcup_{i=1}^t S_i^l$ as

$$\widehat{\boldsymbol{\mu}}_t^l = \frac{1}{t} \sum_{i=1}^t \frac{1}{\left|S_i^l\right|} \sum_{\boldsymbol{x} \in S_i^l} \phi(\boldsymbol{x}) \;\;.$$

These data summaries can then be centralized and the global data summary $\widehat{\boldsymbol{\mu}}$ is computed as the weighted average, where n_l is the number of samples the learner l received and $n = \sum_{i=1}^l n_l$ is total amount of samples accros all learners

$$\widehat{\boldsymbol{\mu}}_t = \sum_{l=1}^m \frac{n_l}{n} \widehat{\boldsymbol{\mu}}_t^l \;\;. \tag{3}$$

With this data summary, the respective $\boldsymbol{\theta}_t$ can be computed centrally and thus we call this the **centralized** approach. Since $\widehat{\boldsymbol{\mu}}_t$ is the exact data summary of the union of local dataset, this approach results in the same model as learning on the union of all local datasets, directly. However, it has two major disadvantages: it does not make use of the local computing power at the data-generating devices and it requires centralizing potentially sensitive data.

To overcome these disadvantages, we propose to train models locally to obtain both $\boldsymbol{\theta}^l$ and $\boldsymbol{\mu}^l$ for each learner $l \in [m]$. We synchronize these local models

Table 1. Summary of transmitted data

Protocol	Centralized	Naïve	Privacy
Send	$\boldsymbol{\mu}_i^t$	$\left\lfloor \boldsymbol{\mu}_i^t + \boldsymbol{\theta}_i^t \right\rfloor$	$\boldsymbol{\theta}_i^t$
Receive	$\boldsymbol{\theta}_{Global}^t$	$\left\lfloor \widehat{\boldsymbol{\mu}_i^t} + \widehat{\boldsymbol{\theta}_i^t} \right\rfloor$	$\widehat{\boldsymbol{\theta}}_i^t$

by averaging the parameters. The average of a set of integer vectors, however, is not necessarily an integer. Instead, the floored average can be computed using only integer operations [5]. Of course, averaging both, $\boldsymbol{\theta}$ and $\boldsymbol{\mu}$, does not solve the privacy issue, since $\boldsymbol{\mu}$ is shared just like in the centralized approach. Thus, we refer to this as **naïve averaging** and use it only as a baseline. Instead, we propose to only average the model parameters $\boldsymbol{\theta}$ and maintain local data summaries $\boldsymbol{\mu}$. We call this approach **privacy-preserving resource constrained averaging**.

This averaging can be performed periodically, i.e., after observing $b \in \mathbb{N}$ batches, hence we call this **periodic averaging**. The frequency of averaging allows to balance communication and model quality: communication effort can be saved by averaging less frequently at the expense of model quality.

Communication can be further reduced by deciding in a data-driven way when averaging has the largest impact. **Dynamic averaging** [7, 8] checks local conditions to determine when to communicate. The algorithm is presented in Algo. 2. It shows the local computation at each round using the (integer) or real-value learning and the local test of the conditions. That is, with a common reference point $\boldsymbol{r} \in \mathbb{Z}^d$, each local learner checks its local condition

$$\|\boldsymbol{\theta}^i - \boldsymbol{r}\|_2^2 = \sum_{j=1}^{d} \left(\boldsymbol{\theta}_j^i - \boldsymbol{r}\right)^2 \leq \Delta \ ,$$

where $\Delta \in \mathbb{Z}$ is a predefined threshold. If all model parameters $\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^m \in \mathbb{Z}^d$ and the reference point $\boldsymbol{r} \in \mathbb{Z}^d$ are d-dimensional integer vectors, and the divergence threshold $\Delta \in \mathbb{Z}$ is also an integer, then the local conditions can be checked using only integer operations.

The algorithm also shows the coordinator processing the submitted models. The augmentation requests additional parameter vectors in case of the dynamic averaging. The number of models received is doubled until the conditions are fulfilled. Hence, it may happen, that finally all learners are used for the central processing which gives a new global model to all local learners.

As mentioned before, using integer exponential families introduces an error into the models. Similarly, using rounded averaging introduces an error. This error can be bounded. Training regular exponential families is a convex learning problem. Indeed, it is trivial to show that the error of using $\widehat{\boldsymbol{\theta}}$ instead of

---

[5] Indeed, the average of two integers in binary representation can be computed using only the logical "and" & and "or" + operations, as well as the bit-shift operator ">>" as $\left\lfloor \frac{a+b}{2} \right\rfloor = (a \ \& \ b) + ((a \ XOR \ b) >> 1)$.

the standard average $\overline{\boldsymbol{\theta}}$, i.e., $\|\widehat{\boldsymbol{\theta}} - \overline{\boldsymbol{\theta}}\|_2$ is bounded by $\sqrt{d}$, where d denotes the number of parameters. Let $\epsilon \in \mathbb{R}_+$ denote a bound on the error of using an integer exponential family instead of a real-valued one. Furthermore, define the cumulative loss on $m \in \mathbb{N}$ learners until time $t \in \mathbb{N}$ as

$$L(t, m) = \sum_{i=1}^{t} \sum_{l=1}^{m} \sum_{(\boldsymbol{x}, y) \in S_t^l} \ell(\boldsymbol{x}, y) ,$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is a loss function. Then, it follows directly from Cor. 3.33 in Kamp [6] that when using stochastic gradient descent to train the local models, the cumulative loss of using resource-constrained dynamic averaging over normal periodic averaging is bounded.

Corollary 1 *Assume $m \in \mathbb{N}$ learners jointly training an integer exponential family with stochastic gradient descent with learning rate $\eta \in \mathbb{R}_+$. Furthermore, assume there exists $\rho \in \mathbb{R}_+$ such that $\|x\|_2 \leq \rho$ for all $x \in \mathcal{X}$. Let L_{Per}, L_{Dyn} denote the cumulative loss when local models are maintained by resource-constrained periodic, resp. dynamic averaging. Then it holds that*

$$L_{Dyn}(t, m) - L_{Per}(t, m) \leq \frac{t}{b\eta^2 \frac{\rho}{\rho^2 + 1}} (\Delta + 2d + \epsilon) .$$

Using the observation that for stochastic gradient descent, periodic averaging on $m \in \mathbb{N}$ learners with batch size $b = 1$ is equal to centralized mini-batch SGD with mini-batch size m and learning rate η/m [cf. 7, Prop. 3], as well as using the standard learning rate of $\eta = \sqrt{t}$ and $\Delta = \sqrt{t}$, it follows that the regret of using resource-constrained dynamic averaging over centralized training is a constant in d and ϵ. In the following section, we empirically compare dynamic averaging to periodic averaging, as well as the centralized approach, both in terms of model quality and communication demand.

4 Experiments

In our experiments, we want to empirically investigate the centralized, the public, and the private scheme of communication and distributed learning. We compare the periodic and the dynamic update in the distributed learning setting for both, the regular and the resource-constrained averaging operator. Before we introduce the specific research questions and results, we describe the experimental setup. To emphasize reproducible research, we selected two open-source-frameworks for our implementation and experiments. To simulate a distributed learning environment, we utilized the `Distributed-Learning-Platform`[6]. The model implementation was based on the `Randomfields`-library [7]. Also, the source code for the new aggregation-operator and experiments can be found online[8].

Algorithm 2 Resource-Constrained Dynamic Averaging Protocol

Input: learning algorithm $\mathcal{A}$, divergence threshold $\Delta \in \mathbb{N}$, parameter $b \in \mathbb{N}$, m learners

Initialization:

    local models $\theta_1^1, \ldots, \theta_1^m \leftarrow$ one random θ

    reference vector $r \leftarrow \theta$

    violation counter $v \leftarrow 0$

Round t at learner i:

    observe $S_t^i \subset \mathcal{X} \times \mathcal{Y}$

    update θ_{t-1}^i using the learning algorithm $\mathcal{A}$

    if $t \mod b = 0$ **and** $\|\theta_t^i - r\|_2^2 > \Delta$ **then**

        send θ_t^i to coordinator (violation)

At coordinator on violation:

    let $\mathcal{B}$ be the set of learners with violation

    $v \leftarrow v + |\mathcal{B}|$

    if $v = m$ **then** $\mathcal{B} \leftarrow [m]$, $v \leftarrow 0$

    while $\mathcal{B} \neq [m]$ **and** $\left\| \left\lfloor \frac{1}{\mathcal{B}} \sum_{i \in \mathcal{B}} \theta_t^i \right\rfloor - r \right\|^2 > \Delta$ **do**

        augment $\mathcal{B}$ by augmentation strategy

        receive models from learners added to $\mathcal{B}$

    send model $\widehat{\theta} = \left\lfloor \frac{1}{\mathcal{B}} \sum_{i \in \mathcal{B}} \theta_t^i \right\rfloor$ to learners in $\mathcal{B}$

    if $\mathcal{B} = [m]$ also set new reference vector $r \leftarrow \widehat{\theta}$

4.1 Model Quality and Communication

During the simulation of the distributed learning environment, we limited our-selves to $m = 16$ learners, however, in future work we want to investigate, how this approach scales with an increasing number of learners. For each integer learner, we limited the number of bits for each parameter to $k = 3$, which re-sults in 2^3 possible choices. Lower choices of k resulted in significantly worse performance in comparison to regular exponential family models. Increasing the number of bits did not result in a sufficient increase in performance and had the disadvantage of higher memory- and thus communication complexity. For the evaluation, we chose three different datasets – `DOTA2`, `COVERTYPE` and `SUSY` – of the UCI-Repository [4]. The datasets possess different properties. While the `DOTA2`-dataset features mostly discrete columns, has many features but a low amount of samples, the `SUSY` dataset consists of only a few real-valued columns, while consisting of many samples. The `COVERTYPE`-dataset sits in between those two with a mix of discrete and numerical features and is a medium dataset size. Details on those datasets can be found in table 2.

    All datasets have undergone the same preprocessing: numerical columns have been discretized into ten bins based on their quantiles. Furthermore, a random subset of 10.000 examples was selected as a holdout set, which in turn was used to estimate the model's structure via the Chow-Liu-algorithm [3]. This step has to be done in order to ensure that all models share the same structure. Other-wise, the aggregation would not be possible. Note that this a serious limitation and future work could also investigate if aggregation can be applied to nodes

Table 2. Dataset and model properties

Dataset	Samples	Features	Classes	Discrete	Numerical	Model Dimension
SUSY	5.000.000	19	2	1	18	1620
COVERTYPE	581.012	55	7	45	10	1596
DOTA2	102.944	117	2	117	0	2790

with distinct graph structures. The remaining data was partitioned horizontally alongside the nodes.

During the running-phase, at each time step t all learners received a batch of $bs = 10$ new samples. As soon as the batch arrived, learners are asked to predict the labels for each of the samples. Afterwards, the learner uses the new samples in order to update its local data summary $\widehat{\mu}$. Thereafter, except for the centralized approach, where a global model is fitted on the accumulated data summaries, learners run an optimization-algorithm with the current data summaries and weights for a specified optimization-budget o in terms of iterations. The budget allows for a further trade-off between model-quality and battery life of some device. Finally, each learner checks if synchronization should be performed. In case of the periodic protocol, the model parameters and/or data summaries are transmitted to the central coordinator after b batches have been processed. In contrast, the dynamic protocol checks if its local conditions hold and only communicates if some condition is violated.

Specifically we want to answer the following **questions**:

– How does the periodic protocol compare against the dynamic one?
– How do resource-constrained models compare to regular ones in terms of predictive quality?

We compare the methods against **two baselines**:

– **Global Learning** We want to compare ourselves against the traditional machine learning setting, where all the data is centralized first. The performance of the global model, a regular exponential family, is evaluated using a 5-fold cross-validation procedure. In the plots this is denoted by `Global`.
– **Local learning** In this setting each device fits a model based on its local data. No communication takes place, thus this setting acts as a guard, showing that communication helps to improve model quality. In the plots this is denoted by `NoSync`.

We have plotted the resource constrained performance of dynamic versus periodic updates for the three datasets and aggregation-mechanisms in Fig. 4.1, 4.2 and 4.3. The y-axis shows the error and the x-axis shows the communication consumption in bytes. The baseline of no synchronization is shown by a dashed-dotted line. We further include the performance of the global baseline as dotted line. The periodic and dynamic approaches are displayed with different markers, while different periods and deltas are displayed with varying color choices. Some
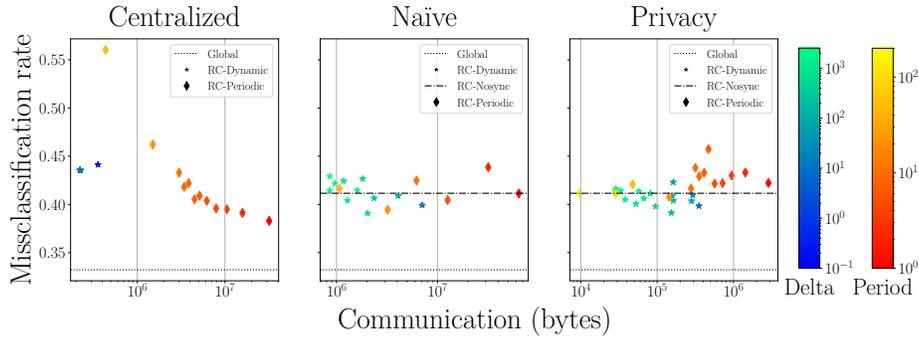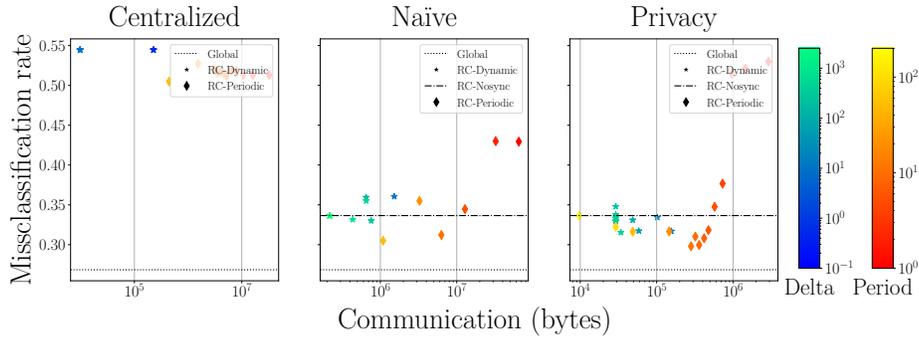
**Fig. 4.1.** Missclassification rate vs Communication (`Covertype`)
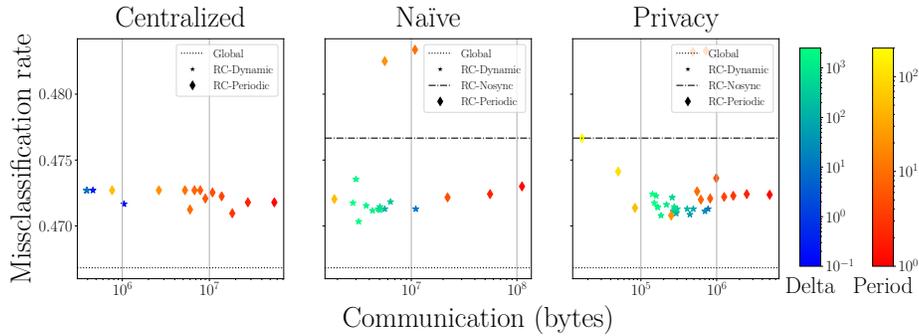
Fig. 4.2. Missclassification rate vs Communication (`DOTA2`)

periodically updated models are worse, but most models are superior to the base line. As expected, for `DOTA2` and `COVERTYPE`, we see that the dynamic update requires less communication in both, the private and the public setting (little stars in the lower left corner). In the numerical dataset `SUSY`, the dynamic update uses less communication resources, but has less accuracy than the periodic one in the public setting. We also see that the quality of the periodically updated models varies much more than the one of the dynamically updated ones. Although for the dataset with only numerical attributes, `SUSY`, the results are not clearly favoring the dynamic update, overall, the answer to the first question leans towards the dynamic protocol. Furthermore, we notice the privacy-preserving aggregation retains the same predictive quality as the naïve approach while reducing the required communication substantially. Thus we focus on the privacy-preserving protocol for the next question.

Fig. 4.4 shows the privacy-preserving scheme for the three datasets comparing the resource-constrained and the normal graphical models. Again, the y-axis shows the error and the x-axis shows the communication consumption. The dotted baseline shows the performance of the global model, while the dotted

Fig. 4.3. Missclassification rate vs Communication (`SUSY`)
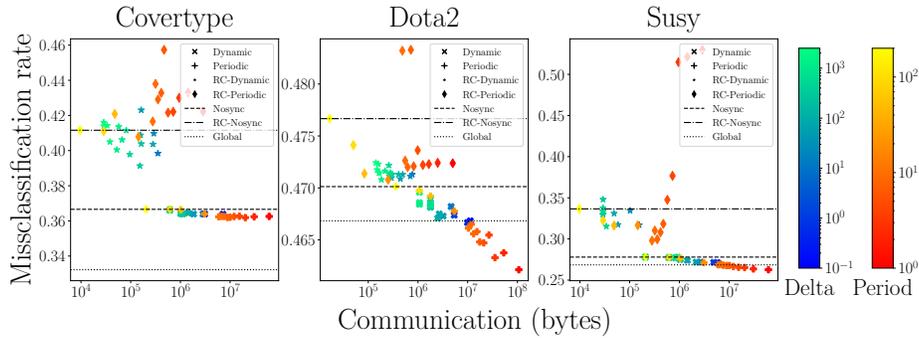
Fig. 4.4. Privacy-Preserving-Resource-Constrained-Averaging

baseline stands for not synchronizing normal models and the dashed-dotted line for not synchronizing resource-constrained models. We have varied the frequency of updates for the normal models as well as for the resource constrained ones using the periodic or the dynamic scheme. The parameters for the frequency of update show not surprisingly that more frequent synchronization leads to better model quality which introduces more communication costs as can be seen below the benchmark spreading towards the right. Many of the resource constrained private models outperform the RC baseline. The variance is due to the frequency of updates. The most accurate model is the not resource constrained one in the lower right corner. It uses by far the most communication. We have varied the parameters and see that actually most models of the resource-constrained dynamic updates (stars) outperform the resource-constrained baseline and approach the regular one. However, we must admit that there are models of the resource-constrained periodically and dynamically updated schemes that are worse than the baseline. These have a low frequency of update in common. In the plots, we easily recognize the parameter choice, which balances quality (low) and communication (left). Though, in all settings, we can save communication cost of 1-3

magnitudes while dropping a few percents of classification performance. This is a natural trade-off we encounter in resource-constrained machine learning methods.

4.2 Energy Savings

In this section we provide a rough estimate on the amount of energy that could be saved using resource-constraint distributed learning instead of centralizing data. For that, we compare the energy required to centralize all data and train a model to the energy required for locally training models and averaging them. In this simplified scenario, we do not assume that the centrally computed model needs to be transferred regularly to the local learners.

To compute the energy required for communication, we assume the data is transmitted over 3G, requiring around 2.9 kWh/GB [18], i.e., $\sigma = 0.0029$ Wh/GB. Furthermore, we assume the central computation is performed on a $p_c = 100$ Watt processor and one of the parallel low-energy processors consume $p_p = 1$ Watt. Since these low-energy processors (e.g., FPGAs) are specialized hardware, the execution time for aggregating data or training a models is usually shorter than on CPUs [2]. However, for simplicity we assume similar runtimes.

Let $c_c \in \mathbb{R}$ denote the amount of communication in GB required by the central approach and $c_p \in \mathbb{R}$ the amount for the parallel one. Let $a, t \in \mathbb{R}$ denote the time required for aggregating $N \in \mathbb{N}$ data points, respectively training a model. With $m \in \mathbb{N}$ local learners, the energy consumed by the central approach then is

$$e_c = \underbrace{(mNa + t)p_c}_{\text{central training}} + c_c\sigma$$

and the energy for the parallel approach is

$$e_p = m\underbrace{(Na + t)p_p}_{\text{local training}} + \underbrace{map_p}_{\text{model aggr.}} + c_p\sigma$$

As reference, we use the empirical results on the SUSY dataset, where the centralized approach achieves an accuracy of 0.73 using $6200640B$ of communication and dynamic averaging achieves a comparable accuracy of 0.69 with $34020B$ of communication. With $a = 10^{-12}$ (roughly the cost of an integer operation on a 1GHz processor) and $t = 10^{-10}$ (roughly 100 integer operations on a 1GHz processor), $m = 16$ and $N = 100$ the rough estimates are $e_c = 0.22\mu Wh$ and $e_p = 0.0033\mu Wh$, i.e., the centralized approach requires more than 67 times more energy than the parallel one. Note that this estimate is conservative since it does not take into account the energy required in the centralized approach for running an operating system or powering additional components of a computer.

To analyze the potential scaling behavior we make one more simplifying assumptions: We assume that the ratio of central to parallel communication remains constant with the number of learners. This assumption is not very realistic, but the actual scaling behavior depends on the underlying learning problem. Increasing the number of learners typically leads to an even more favorable ratio
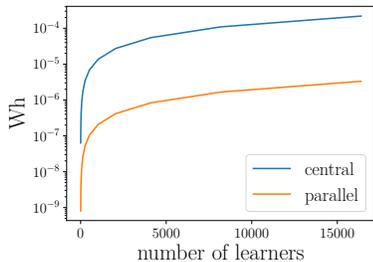
Fig. 4.5. Scaling behavior of the estimated energy consumption with the number of learners.
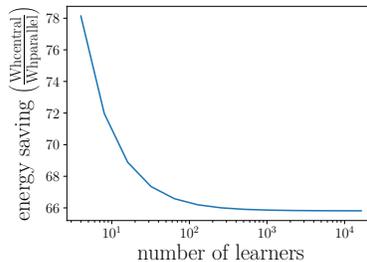
Fig. 4.6. Scaling behavior of the energy saving's ratio with the number of learners.

for simple learning problems, whereas for hard learning problems it can be less favorable [6]. Under this assumption, we show the estimated energy consumption in Figure 4.5. It shows that the energy consumption of the parallel approach remains substantially lower for larger amounts of learners. We show the relative reduction in energy consumption (i.e., e_c/e_p) in Figure 4.6. It shows that the relative reduction decreases with the number of learners, but remains above 65 even for large numbers of learners.

5   Discussion

As the empirical evaluation has shown, using only integer operations allows to successfully train models distributedly. However, when using the integer average the relationship between communication and model quality is not as clear as for normal averaging, where more communication reliably leads to higher model quality. The results for resource-constrained averaging indicate that the errors through rounding the average lead to less predictable behavior. The results on SUSY furthermore indicate that too much communication can even be harmful. A possible explanation is that when averaging very often, small changes of local models will get leveled out by the rounding. Thus, the effect of local training is nullified, delaying the overall training process.

This rounding effect might vanish when using larger numbers of learners (in the experiments, we only used $m = 16$ learners). At the same time, this can influence the communication reduction. When computing the energy savings, we assumed the reduction remains constant with larger numbers of learners. It is conceivable for dynamic averaging that larger numbers of learners lead to an even greater reduction in communication when compared to the centralized approach. However, the effects of rounding might similarly lead to more local violations and ultimately a higher amount of communication. To answer this question and determine whether the approach is useful in practice, it is necessary to further study the effects of rounding the average, both empirically and theoretically.

6 Conclusion and Future work

In this paper, we proposed a new resource-constrained averaging operator, which can be evaluated on ultra-low-power hardware using only integer operations. Besides, we have shown, that the same applies to the evaluation of the local conditions, which consecutively allows performing every step in this distributed learning setting in a resource-constrained fashion. Furthermore, we have shown that the excess loss of using the dynamic averaging protocol over the periodic protocol is bounded. In our experiments we verified that using resource-constrained averaging of integer exponential family models, we reach a similar performance in terms of prediction quality compared to regular exponential family models with access to the non-restricted parameter space, while reducing network requirements substantially. Besides, we do not only save energy by reducing network communication, but also by employing these models on specific, cheap available hardware. However, there is still a decrease in predictive quality from using resource-constrained models, so this trade-off has to be taken into account: Are we willing to drop $x\%$ accuracy for the sake of energy or bandwidth savings?

Future work This work opens many new exciting research questions, e.g., how does the averaging perform if we vary the number of nodes in our learning environment. Does it increase the error or do more nodes provide more information for faster convergence? Furthermore, it would be interesting to investigate the effects of resource-constrained-averaging on the parameter vectors in a controlled environment using artificial datasets with known parameters. Besides, the dynamic averaging protocol might yield better results in terms of communication cost in the presence of more nodes, since the partial synchronization mechanism will be triggered more often. Another way to reduce communication could be the clique-wise transmission of data summaries and/or parameters. This technique could possibly help to overcome the limitations of the fixed graph structure by matching the cliques of the different learners and only averaging their parameters. Also, we saw that using incorrect choices for the synchronization period and/or delta, we receive sub-optimal solutions with low predictive quality. This raises the need for informed methods to choose those hyperparameters, incorporating the parameter space, the number of processed samples as well as the optimization budget. Furthermore, since we only considered plain averaging in this work, we could also try to adopt different aggregation mechanisms, e.g., performance-weighted averages or the Radon machine [9], to the resource-constrained setting. Further work could also examine the adaptivity to time-variant distributions as well as the performance / convergence on non i.i.d datasets.

Bibliography

[1] Administration, U.E.I.: How much electricity does a typical nuclear power plant generate? https://www.americangeosciences.org/critical-issues/faq/how-much-electricity-does-typical-nuclear-power-plant-generate (2018), accessed: 02.12.2019

[2] Asano, S., Maruyama, T., Yamaguchi, Y.: Performance comparison of fpga, gpu and cpu in image processing. In: 2009 FPL. pp. 126–131. IEEE (2009)

[3] Chow, C., Liu, C.: Approximating discrete probability distributions with dependence trees. IEEE Trans. Inf. Theor. 14(3), 462–467 (Sep 2006)

[4] Dua, D., Graff, C.: UCI machine learning repository (2017), http://archive.ics.uci.edu/ml

[5] Hammersley, J.M., Clifford, P.E.: Markov random fields on finite graphs and lattices. Unpublished manuscript (1971)

[6] Kamp, M.: Black-Box Parallelization for Machine Learning. Ph.D. thesis, Rheinische Friedrich-Wilhelms-Universität Bonn (2019)

[7] Kamp, M., Adilova, L., Sicking, J., Hüger, F., Schlicht, P., Wirtz, T., Wrobel, S.: Efficient decentralized deep learning by dynamic model averaging. In: ECML-PKDD. pp. 393–409. Springer (2018)

[8] Kamp, M., Boley, M., Keren, D., Schuster, A., Sharfman, I.: Communication-efficient distributed online prediction by dynamic model synchronization. In: ECML-PKDD. pp. 623–639. Springer (2014)

[9] Kamp, M., Boley, M., Missura, O., Gärtner, T.: Effective parallelisation for machine learning. In: Advances in Neural Information Processing Systems 30, pp. 6477–6488. Curran Associates, Inc. (2017)

[10] Kamp, M., Bothe, S., Boley, M., Mock, M.: Communication-efficient distributed online learning with kernels. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 805–819. Springer (2016)

[11] Kemp, S.: Digital 2019 global digital overview. https://datareportal.com/reports/digital-2019-global-digital-overview (2019), accessed: 02.12.2019

[12] Lim, W.Y.B., Luong, N.C., Hoang, D.T., Jiao, Y., Liang, Y., Yang, Q., Niyato, D., Miao, C.: Federated learning in mobile edge networks: A comprehensive survey. CoRR abs/1909.11875 (2019), http://arxiv.org/abs/1909.11875

[13] McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics. pp. 1273–1282 (2017)

[14] Mohan, N., Kangasharju, J.: Edge-fog cloud: A distributed cloud for internet of things computations. In: 2016 Cloudification of the Internet of Things (CIoT). pp. 1–6. IEEE (2016)

[15] Piatkowski, N.: Distributed generative modelling with sub-linear communication overhead. In: Kamp, M., Paurat, D., Krishnamurthy, Y. (eds.) Decentralized Machine Learning at the Edge. Springer (2019)

[16] Piatkowski, N., Lee, S., Morik, K.: Integer undirected graphical models for resource-constrained systems. Neurocomputing 173, 9–23 (2016)

[17] Piatkowski, N.P.: Exponential families on resource-constrained systems. Ph.D. thesis, TU Dortmund (2018)

[18] Pihkola, H., Hongisto, M., Apilo, O., Lasanen, M.: Evaluating the energy consumption of mobile data transfer—from technology development to consumer behaviour and life cycle thinking. Sustainability 10(7), 2494 (2018)

[19] Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: Vision and challenges. IEEE Internet of Things Journal 3(5), 637–646 (2016)

[20] Wainwright, M.J., Jordan, M.I.: Graphical models, exponential families, and variational inference. Found. Trends Mach. Learn. 1(1-2) (2008)