Adaptive Scheduling for Machine Learning Tasks over Networks

Konstantinos Gatsis

***Abstract*— A key functionality of emerging connected autonomous systems such as smart transportation systems, smart cities, and the industrial Internet-of-Things, is the ability to process and learn from data collected at different physical locations. This is increasingly attracting attention under the terms of distributed learning and federated learning. However, in this setup data transfer takes place over communication resources that are shared among many users and tasks or subject to capacity constraints. This paper examines algorithms for efficiently allocating resources to linear regression tasks by exploiting the informativeness of the data. The algorithms developed enable adaptive scheduling of learning tasks with reliable performance guarantees.**

I. Introduction

Conventional machine learning applications require data to be collected at a centralized location to be trained in a centralized manner, for example, running stochastic gradient descent from sampling the pool of data. However, the emergence of new cyber-physical architectures that are distributed requires rethinking this approach to enable learning across locations, for example when data are distributed or agents need to run tasks in different physical locations – see, for example, Fig. 1. Applications domains of interest include, for example the Industrial Internet-of-Things, Smart Cities, and Smart Transportation Systems.

One significant instance of this is the architecture of federated learning, a term used by Google [1], [2]. The primary role of this architecture is to enable multiple users to jointly solve a machine learning problem over a communication network from data collected from the users. A major challenge is that in many modern machine learning applications data can be very high-dimensional, making their communication costly and inefficient. To alleviate this communication bottleneck, various approaches are being explored towards more communication-efficient machine learning. One direction is based on communicating the machine learning model parameters as they are being trained, such as the weights of a Deep Neural Network, instead of the data itseldf, or communicate the gradients of the optimization objective with respect to the parameters. In cases of deep learning models where even these parameters can be high dimensional neural network weights, sparsification and quantization of the weights or the gradients is also introduced to limit the communication cost [1], [3]–[5]. Lazy updates are introduced in [6], combinations of non-periodic updates and quantization is explored in [7], and considerations of federated multi-task learning is introduced in [8].

The author is with the Department of Engineering Science, University of Oxford, Parks Road, Oxford, OX1 3PJ, UK. Email: konstantinos.gatsis@eng.ox.ac.uk.

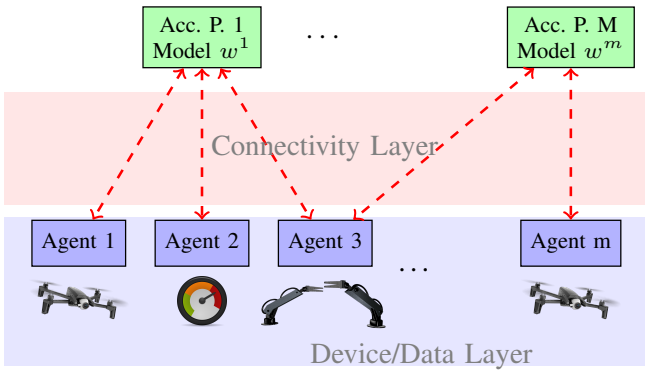

Fig. 1. Architecture for solving machine learning tasks over communication networks of agents. Agents are collecting data and are communicating with access points/servers. Communication efficiency is addressed in this work by assessing the informativeness of the data.

Furthermore, when distributed learning is taking place over a wireless network, there is an interest in allocating the available network resources efficiently among the users holding the data [9]. This has received increased interest in the wireless networking community leading to different directions including allocation of resources such as power [10], [11] or rates [12]–[14]. Often in such works the resource allocations are fixed and then machine learning performance properties such as convergence are analyzed. The problem of scheduling gradient updates over multiple access channels has also received initial consideration, for example comparing time-based approaches with approaches based on channel conditions [15], or including gradient information [16], [17], or combinations with analog communication [18], [19].

The purpose of this work is to introduce a new communication-efficient learning approach by posing a fundamental question: How informative are the collected data for the purpose of solving a machine learning problem? And how can this be used to communicate with efficiency over a network? Our approach hinges on the fact that when model parameters need to be updated from noisy data, then not all updates are equally informative. As a result, performing updates selectively may be beneficial, and we can evaluate the informativeness of the data by estimating the gain in machine learning performance. Building upon this intuition we propose scheduling algorithms that aims to update the machine learning task whose data carry the most informative information, i.e., bring the most gain. By prioritizing updates with more relevant information we aim to more efficiently use the communication resources. This is illustrated when communication capacity is limited, and it is undesirable to

have a node sending updates all the time, or when multiple machine learning tasks are solved over a network and not all tasks can be updated from distributed data at the same time.

The approach is developed for the problem of solving linear regression tasks in order to minimize the expected square prediction errors for the tasks. The problem is formulated in Section II and the scheduling problems to deal with communication capacity constraints are introduced. In Section III we present our scheduling algorithms, which attempt to prioritize updates whose data carries the most information, i.e., that would lead to the highest performance gain (minimize the prediction loss). The approach is theoretically analyzed and guarantees are provided about convergence and the tradeoff with required communication resources. Furthermore, as our method is ideally developed when the loss function is completely known, i.e., the data distribution is known, significant effort is placed on an approximate scheduling scheme that uses the currently available data to estimate how informative the current update will be.

Our technical methodology borrows ideas from the problem of scheduling multiple control tasks over a shared resource, where approaches such as opportunistic scheduling over wireless links [20], [21] and age-of-information or value-of-information [22]–[24] are proposed. Our methodology is also related to approaches in event-triggered learning that try to update only if necessary [25], [26]. Moreover, the problem of solving multiple tasks concurrently is studied, unlike previous approaches that are focused on solving a single machine learning task. This creates both a higher communication load, but also an opportunity to differentiate among tasks based on their performance gain, as described above.

In numerical evaluations in Section IV we validate the performance of our algorithms compared to other scheduling benchmarks and evaluate the performance gains at different regimes, when the covariance of the data changes, or when the number of available data is small or large. Our evaluations show significant performance improvements compared to other approaches in the literature that treat the magnitude of the gradients as a measure of the informativeness of the current update. We conclude with a discussion on future work.

II. Problem Setup

The architecture examined in this paper, shown in Fig. 1, involves multiple access-points/servers interested in building data-driven models by solving machine learning tasks on data that are collected by multiple agents. In general, both multiple machine learning tasks may need to be solved, and multiple agents are involved. We consider machine learning tasks indexed by $j = 1, \dots, m$. For each task we are interested in finding a vector of weights (parameters) w^j of appropriate dimensions. To assess the performance of each task we also have a performance metric (cost) to be minimized denoted by $J^j(w^j)$ for each task.

The aim will be to achieve this with communication efficiency. This is for example the case when an agent should not communicate all the time over a communication network to update the vector of parameters at the access point/server. Or when due to capacity constraints not all agents can communicate at the same time. This leads to problems of scheduling the data updates. Below we give a more precise description of the learning tasks in our framework, which correspond to linear regression tasks, and the scheduling problems considered.

A. Model for each learning task

We consider each machine learning task to be a linear regression problem [27, Ch. 9]. We are interested in finding a vector of weights w that explains the relationship between random variables $(x, y) \in \mathbb{R}^n \times \mathbb{R}$, i.e., $y \approx x^T w$. The random variables (x, y) follow in general a joint distribution denoted by μ. The desired choice for the weights is the one that minimizes the expected square prediction error, i.e.,

$$\min_w J(w) = \frac{1}{2}\mathbb{E}_{(x,y)\sim\mu}(y - x^T w)^2 \tag{1}$$

where the expectation is with respect to the data distribution μ – in the sequel we drop this notation when it is implied that expectation is with respect to this distribution.

The optimal solution w^* is given as the solution to the linear equations

$$\mathbb{E}xx^T w^* - \mathbb{E}xy = 0. \tag{2}$$

Towards finding an optimal set of weights, we would like to employ a gradient descent algorithm. Starting from some initial set of weights w_0 we would like to update the weights according to

$$w_{k+1} = w_k - \varepsilon \nabla J(w_k) \tag{3}$$

where $\nabla J(w_k) = \mathbb{E}xx^T w_k - \mathbb{E}xy$, and $\varepsilon > 0$ is a small positive stepsize. As will be illustrated later, choosing $\varepsilon < 2/\lambda_{\max}(\mathbb{E}xx^T)$ guarantees convergence.

The distribution of the data is not a priori known, and hence as is common in machine leaning, e.g., in empirical risk minimization [27, Ch. 2], we will attempt to minimize the empirical cost computed as an average over collected data. Specifically we assume that at each iteration k there are N new data points of the form

$$(x_i, y_i) \in \mathbb{R}^n \times \mathbb{R}, \qquad i = 1, \dots, N. \tag{4}$$

We assume each data pair is independent and identically distributed according to a distribution μ.[1] Then we form the empirical cost

$$\hat{J}(w) = \frac{1}{2}\frac{1}{N}\sum_{i=1}^{N}(y_i - x_i^T w)^2 \tag{5}$$

It may also be desirable to add regularization terms in the objective to avoid overfitting, but in this paper we restrict

[1]This setup arises either when an agent in Figure 1 collects N new independent samples at each iteration, or when it just maintains a large pool of samples and selects randomly N from them at each iteration as frequently done in stochastic gradient descent practice.

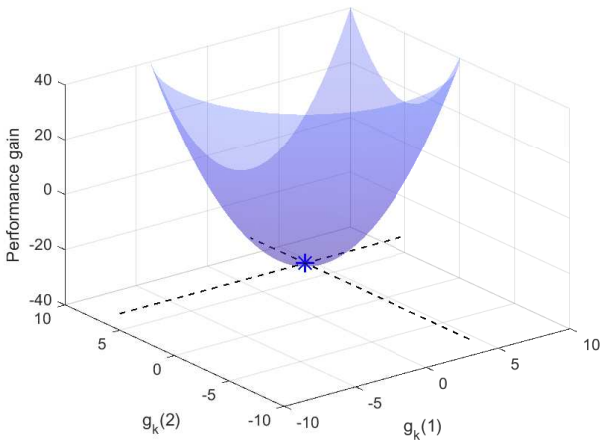

Fig. 2. Example of the performance gain as a function of the stochastic gradient update in two dimensions ($n = 2$). Depending on the informativeness of the data, the gain varies and is exploited in the proposed scheduling mechanism in Section III.

attention to the objective above. With this approximation, we follow a stochastic gradient vector

$$w_{k+1} = w_k - \varepsilon g_k \tag{6}$$

computed over the data as

$$g_k = \nabla \hat{J}(w_k) = \frac{1}{N}\sum_{i=1}^{N}\left(x_i x_i^T w_k - x_i y_i\right) \tag{7}$$

After this update the prediction error becomes

$$J(w_{k+1}) = \frac{1}{2}\mathbb{E}(y - x^T w_{k+1})^2 \tag{8}$$

where the expectation is with respect to the distribution μ.

We note that since the N data points are random, so is the constructed gradient direction g_k, the updated vector w_{k+1}, as well as the performance metric $J(w_{k+1})$. To evaluate how good is this updated prediction error, we would like to measure

$$\mathbb{E}[J(w_{k+1})|w_k] = \mathbb{E}_{data\sim\mu^N}[J(w_{k+1})|w_k] \tag{9}$$

It is important to note here that the expectation is over the N i.i.d. data that are collected at iteration k and used to construct the stochastic gradient g_k. In the paper, whenever an expectation over iterates w_k is taken, this is an expectation over the data collected until time k.

B. Scheduling problem for single task

Given the above modeling for a machine learning task that needs to be solved, the scheduling problem is as follows: given the stochastic gradient g_k available from the data at each iteration, select whether to transmit this gradient update over the communication network to a receiving server. The server maintains a current vector of weights w_k which will be updated depending on the case to

$$w_{k+1} = \begin{cases} w_k - \varepsilon g_k & \text{if scheduled} \\ w_k & \text{if not scheduled} \end{cases} \tag{10}$$

We also denote with $\alpha_k \in \{1, 0\}$ the decision to schedule/transmit or not. This decision is made at the node. At the next iteration $k+1$ a new set of data is collected as in (4), a new stochastic gradient direction g_{k+1} is computed, and the process repeats. The aim will be to avoid sending updates all the time to limit the communication burden.

C. Scheduling problem for multiple tasks

Given the above modeling for each of the machine learning tasks that need to be solved, the scheduling problem for multiple tasks is as follows. Given the available data across all tasks $j = 1, \ldots, m$, select one task j^* whose weights will be updated at the server changing values from $w_k^{j^*}$ to $w_{k+1}^{j^*}$, while all other weights remain the same at w_k^j for all $j \neq j^*$. More succinctly, for any given scheduling rule, we denote the vector after the scheduling decision as

$$w_{k+1}^j = \begin{cases} w_k^j - \varepsilon g_k^j & \text{if } j \text{ scheduled} \\ w_k^j & \text{if } j \text{ not scheduled} \end{cases} \tag{11}$$

This means that in an architecture as shown in Fig 1, due to communication capacity constraints, at most one communication link is active at any time step, and hence at most one of the tasks can be updated over the network. The setup can be easily generalized to the case where p out of m tasks can be scheduled, for some $p < m$.

III. Proposed scheduling approach

A. Scheduling a single task

The approach is based on the notion of performance gain which can be thought as a measure of how informative are the data collected at each time step with respect to the machine learning problem. The gain can be calculated as the difference

$$J(w_k - \varepsilon g_k) - J(w_k), \tag{12}$$

measuring how much will the objective change after the update. Owing to the fact that the objective function is a quadratic function of the weights w, this gain can be thought as a quadratic function of the stochastic gradient g_k. The best gain (minimum) is a negative value and is achieved at some vector g_k that updates the weights to the optimal solution of the problem. On the other hand, as g_k grows larger in any direction, the quadratic form takes a larger positive value, meaning that actually no gain is achieved. The update direction in turn depends on the random data samples by (7). As a result the gain can be large or small depending on the *informativeness of the data* and how well they point towards to the optimal solution. The gain also depends on the current vector of weights w_k. An illustration is shown in Fig. 2.

The proposed approach then is to send a gradient update if the gain is large enough. Mathematically we write

$$\alpha_k = \begin{cases} 1 & \text{if } J(w_k - \varepsilon g_k) - J(w_k) \leq -\lambda \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

for some scalar parameter $\lambda > 0$.

Intuitively this approach saves up communication resources, because the larger the parameter value λ is, the more infrequent the updates will be. But then the question is what can be said about the progress of learning. We have then the following result.

Theorem 1 (Convergence)**.** *Consider the optimization problem defined in (1). Consider the update rule in (10). Suppose g_k are independent random variables with mean equal to $\nabla J(w_k)$ at each iteration k and covariance G. Consider the update rule in (13). Then for any iteration N we have that*

$$\mathbb{E}J(w_N) \leq \rho^N J(w_0) + (1-\rho^N)\left[J(w^*) + \frac{\lambda + \varepsilon^2 Tr(\Sigma_x G)}{1-\rho}\right] \tag{14}$$

where the expectation is with respect to the data collected until iteration N, and the parameters are $\Sigma_x = \mathbb{E}xx^T/2$ and $\rho = \max_i(1-\varepsilon\lambda_i(\mathbb{E}xx^T))^2$.

Proof. First we note that due to the choice in (13) the following inequality holds for all times (technically it holds almost surely as all the variables involved are random variables)

$$\alpha_k J(w_k - \varepsilon g_k) + (1-\alpha_k)J(w_k) \leq \lambda + J(w_k - \varepsilon g_k). \tag{15}$$

This can be easily verified by examining the two cases $\alpha_k = 0$ or 1. Then note that by the dynamics in (10) we can rewrite the left hand side as

$$J(w_{k+1}) \leq \lambda + J(w_k - \varepsilon g_k). \tag{16}$$

Taking expectation over the stochastic gradient g_k, conditioned on the current iterate w_k, we get that

$$\mathbb{E}[J(w_{k+1}) \,|\, w_k] \leq \lambda + \mathbb{E}[J(w_k - \varepsilon g_k) \,|\, w_k] \tag{17}$$

Then given the fact that the function $J(w)$ is quadratic and can be rewritten as $J(w) = J(w^*) + (w-w^*)^T\Sigma_x(w-w^*)$, and the properties of the stochastic gradient, we have that

$$\begin{aligned}\mathbb{E}[J(w_k - \varepsilon g_k) \,|\, w_k] &= J(w^*) + \\ (w_k - \varepsilon\mathbb{E}g_k - w^*)^T&\Sigma_x(w_k - \varepsilon\mathbb{E}g_k - w^*) + \mathrm{Tr}(\Sigma_x G)\end{aligned} \tag{18}$$

Here we can substitute $\mathbb{E}g_k = \nabla J(w_k) = 2\Sigma_x(w_k - w^*)$, and the fact that $(I-\varepsilon 2\Sigma_x)'\Sigma_x(I-\varepsilon 2\Sigma_x) \preceq \rho\Sigma_x$ to conclude that

$$\begin{aligned}\mathbb{E}[J(w_k - \varepsilon g_k) \,|\, w_k] &\leq J(w^*) + \\ &\rho(w_k - w^*)^T\Sigma_x(w_k - w^*) + \mathrm{Tr}(\Sigma_x G)\end{aligned} \tag{19}$$

Substituting this in (17) and rearranging terms we find that

$$\begin{aligned}\mathbb{E}[J(w_{k+1}) \,|\, w_k] &\leq \lambda + \rho J(w_k) \\ &+ \varepsilon^2\mathrm{Tr}(\Sigma_x G) + (1-\rho)J(w^*)\end{aligned} \tag{20}$$

Taking expectation on both sides with respect to the variable w_k, and iterating over time $k = 1, \ldots, N$, we get the desired result (14). □

The result verifies that the update rule converges (in a stochastic sense) because $\rho < 1$ as can be confirmed by the appropriate choice of the stepsize $0 < \varepsilon < 2/\lambda_{\max}(\mathbb{E}xx^T)$. Essentially the result follows because the function $J(w)$ can be thought as a Lyapunov function for the stochastic dynamics of the update in (10) and showing the convergence result. A direct consequence of the above result is

$$\limsup_{N\to\infty} \mathbb{E}J(w_N) \leq J(w^*) + \frac{\lambda + \varepsilon^2\mathrm{Tr}(\Sigma_x G)}{1-\rho} \tag{21}$$

This means that eventually we get close to the optimal set of weights w^* subject to some overshoots. There is the effect of the stochastic gradient and its covariance G, which can be made small in practice by choosing the step size ε to be small – or by choosing a diminishing stepsize which will be analyzed in future work. Moreover, there is a penalty proportional to the parameter λ, introduced to save up on communication cost.

Remark 1. In Theorem 1 we assumed for simplicity that the stochastic gradients have bounded covariances that are constant over time. In reality for the problem above the covariance of the stochastic gradient in (7) will depend on the current iterate w_k, but our choice can be justified in two ways. We can either consider these covariances to be uniformly bounded over time by some constant G. Or alternative if we consider the case close enough to the equilibrium $w_k \approx w^*$, then it follows that the covariances are indeed constant over time. A more detailed investigation will be explored in a follow up work.

Furthermore, we can establish the following guarantee about the total communication rate of the proposed approach.

Theorem 2 (Communication guarantee)**.** *Consider the same setup as in Theorem 1. The total expected communication rate satisfies*

$$\limsup_{N\to\infty} \sum_{k=0}^{N} \mathbb{E}\alpha_k \leq \frac{J(w_0) - J(w^*)}{\lambda}. \tag{22}$$

Here the expectation is with respect to the data collected as iterations $N \to \infty$.

Proof. Due to the choice in (13) the following inequality holds for all times (technically it holds almost surely as all the variables involved are random variables)

$$\lambda\alpha_k + J(w_{k+1}) \leq J(w_k). \tag{23}$$

This can be easily verified by examining the two cases $\alpha_k = 0$ or 1. Taking expectation, iterating over time $k = 0, \ldots, N$, and summing up, we conclude that

$$\lambda\sum_{k=0}^{N} \mathbb{E}\alpha_k + \mathbb{E}J(w_{N+1}) \leq J(w_0). \tag{24}$$

This can be rearranged as

$$\sum_{k=0}^{N} \mathbb{E}\alpha_k \leq \frac{J(w_0) - \mathbb{E}J(w_{N+1})}{\lambda} \tag{25}$$

Moreover, since any value of the variable w_{N+1} is in general suboptimal, we have that $\mathbb{E}J(w_{N+1}) \geq J(w^*)$. From which we get the desired result (22). □

This verifies that increasing λ will decrease the resulting communication rounds in an inversely proportional manner.

B. Practical scheduling scheme

Despite the above guarantee, implementing the proposed scheduling scheme in (13) would be practically challenging because it requires information that is not known. Specifically it would require knowledge of the data distribution in order to compute the actual performance gain. Since the true distribution is unknown, one approach is to *estimate the performance gain from the data*. In particular, since the objective function is quadratic, we can write the performance gain as

$$J(w_k - \varepsilon g_k) - J(w_k) = -\varepsilon g_k^T \nabla J(w_k) + \frac{1}{2}\varepsilon^2 g_k^T \nabla^2 J(w_0) g_k \quad (26)$$

Then we can approximate the quantities

$$\nabla J(w_k) \approx \frac{1}{N}\sum_{i=1}^{N}\left(x_i x_i^T w_k - x_i y_i\right) = g_k \quad (27)$$

$$\nabla^2 J(w_k) \approx \frac{1}{N}\sum_{i=1}^{N} x_i x_i^T \quad (28)$$

where we note that the stochastic gradient direction g_k appears again. Hence, using the expression for the information gain in (26), we can approximate the gain as

$$J(w_k - \varepsilon g_k) - J(w_k) \approx -\varepsilon g_k^T \left[I - \varepsilon\frac{1}{2}\frac{1}{N}\sum_{i=1}^{N} x_i x_i^T\right] g_k \quad (29)$$

It is crucial to emphasize that *this is no longer a simple quadratic function* of the data but a more complicated function - we note that the data appear both in the stochastic gradients g_k by (7) as well as the matrix in the middle. An example is presented next. This approximate value of the gain may take again positive or negative values but it induces an approximation error/bias.

As a result, we can implement the scheduling decision in (13) with the approximation in (29). In this case we no longer have the performance guarantee in Theorem 1. In numerical evaluations however we see that despite the bias this mechanism performs very well.

Example 1. Consider as an illustration the case where a task has only one $N = 1$ sample (x_1, y_1) and the initial set of weights is zero $w_0 = 0$. Then the stochastic gradient step by (7) is $g_k = -x_1 y_1$ and the estimated gain by (29) is

$$J(w_1) - J(0) \approx -\varepsilon y_1^2 \|x_1\|^2 (1 - \frac{\varepsilon}{2}\|x_1\|^2) \quad (30)$$

This estimated gain is negative for small values of the norm $\|x_1\|$ and attains its minimum value at the point $\|x_1\| = 1/\sqrt{\varepsilon}$. The estimated gain increases and becomes positive as $\|x_1\|$ increases. This means that our scheduling mechanism in (13) would avoid scheduling the updates in that case. The expression becomes more complex for larger number of samples.

C. Scheduling multiple tasks

We have m tasks, with objectives $J^1(w^1), \ldots, J^m(w^m)$ and we are interested in selecting to update one of the vectors w^j. We propose a greedy scheduling algorithm: *select the update that will bring the best gain*. Using the notation in the above section we select

$$j^* = \underset{j=1,\ldots,m}{\operatorname{argmin}} J^j(w_k^j - \varepsilon^j g_k^j) - J^j(w_k^j) \quad (31)$$

where the gain can be approximated from the data for each task as explained earlier in (29). This can exploit the fact that not all of the updates are equally informative due to the randomness of the data collected for each task.

The theoretical guarantees for this scheduling approach will be further developed in future work. The practical benefits will be shown numerically in the next section.

Remark 2 (Practical Implementation)**.** The scheduling decision in (31) is made in a centralized manner. This can be made for example at the server end of the architecture in Fig. 1, deciding which node to upload information, or at a node if it needs to send updates to many different servers. The proposed approach requires having access to all the available data when making the scheduling decision, and compare the informativeness of the data per task in (31). This is feasible in the second scenario at the node, but is more challenging to implement in practice in the first scenario at the server end. One potential approach, also proposed in [16], [17], is to perform this in two stages: first have each node estimate the performance gains locally from the data, and only send these scalar gains at the server (not the data or the gradients). Then the server can compare the gains among them and decide which gradient vector to receive. Other approaches, such as estimating the performance gain at the server end, will be examined in future work.

Remark 3 (Computational cost)**.** We note here that the computational cost for making the scheduling decision in (31) is small. First, for each task, the vector g_k given by (7) can be computed by first computing the residual terms $x_i^T w_k - y_i$ and then multiplying them with the vectors x_i and summing up for $i = 1, \ldots, N$. Then, for each task, the term by (29) can be computed as

$$\varepsilon\frac{1}{2}\frac{1}{N}\sum_{i=1}^{N}(g_k^T x_i)^2 - g_k^T g_k \quad (32)$$

Overall across tasks these require $O(Nnm)$ operations. Finally finding the minimum across m in (33) is not costly for a small number of tasks m.

D. Other approaches in the literature: Gradient-based scheduling

We note that a different scheduling approach would be to treat the tasks with the largest updates as the most important. In that case, a simple approach would be to schedule the task with the largest norm of the (stochastic) gradient, i.e.,

$$j^* = \underset{j=1,\ldots,m}{\operatorname{argmax}} \|g_k^j\|^2 \quad (33)$$

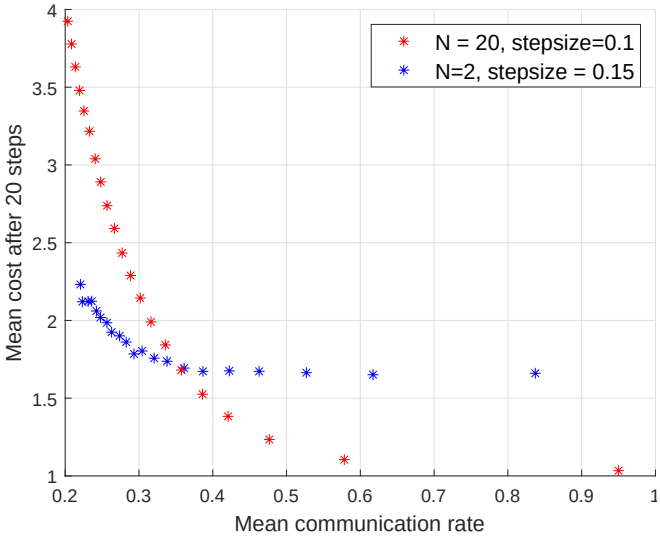

Fig. 3. Evaluation of the tradeoff between communication rate and machine learning performance of the proposed scheduling approach in (13). The curves are obtained by varying the values of the parameter λ for different settings of the optimization problem.

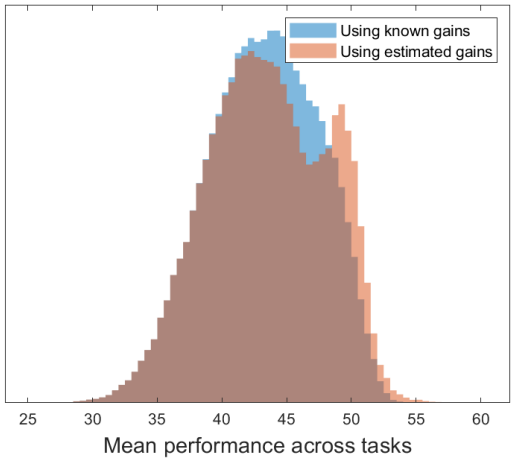

Fig. 4. Comparison between our scheduling approach in (31) knowing the distributions to compute the gains by (26) versus estimating the gain by (29). We observe that the lack of model knowledge is counteracted from available data.

In numerical comparisons we show that this scheme often leads to worse performance. From our expression on (29) we see that for small stepsizes ε the magnitude of the gradient may serve as a proxy for the performance gain. This may also be the case when the Hessian of the problem is closer to an identity matrix.

We note that the idea of scheduling based on gradient magnitudes has been proposed in very recent works in federated learning over wireless channels [16], [17]. Similarly, in the context of sparsification and quantization for high-dimensional gradient updates, it has been proposed that elements with larger values should be given priority [3], [4]. Our findings hence point to a *novel and more communication-efficient approach for gradient updates*.

IV. Numerical results

In this section we analyze and compare numerically the proposed approach. In this section we make an additional assumption about the data samples. Specifically we assume that the points x_i are i.i.d. Gaussian random variables, while the points y_i are given as $y_i = x_i^T w^* + \eta_i$ where w^* is the true parameter and η_i are i.i.d. Gaussian measurement noises.

A. Tradeoff between communication rate and learning performance

We consider the scheduling algorithm in (13) with the performance gains estimated as in (29). We consider a problem with dimensions $n = 2$, with covariances $\mathbb{E}xx^T = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$ (which affects the Hessian of the problem), the initial weights are $w_0 = 0$, and the true weights equal to $w^* = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$. First for stepsize $\varepsilon = 0.1$ and $N = 20$ data points available at each iteration (cf.(4)), we simulate algorithm (13) for varying values of the parameter λ. In Fig. 3 we plot the observed mean learning performance after 20 iterations ($J(w_{20})$) versus the average communication rate ($1/20 \sum_{k=0}^{19} \alpha_k$). We observe that the proposed scheduling approach indeed allows us to tradeoff communication rate with machine learning performance.

Then for stepsize $\varepsilon = 0.15$ and $N = 2$ data points available at each iteration, we simulate the algorithm again for varying value of the parameter λ and plot the results in Fig. 3. This case corresponds to both a very noisy gradient setting due to the very small number of data at each iteration and also a larger stepsize. We observe now qualitatively a different tradeoff curve. In this case we can remarkably lower the communication rate without a significant loss in learning performance. This indicates that our proposed scheduling approach may achieve better communication efficiency when data are more noisy.

B. Bias of data-based estimated performance gains

Next we consider the problem of scheduling multiple m tasks, using the rule in (31). We would like to investigate how much bias is introduced by our practical scheme that is based on estimating the performance gain for each task update based on the currently available data. Hence we compare (31) when using the performance gains computed by (26) that requires knowledge of the problem distributions, with the fully data-based scheme in (29).

In particular we consider scheduling $m = 2$ linear regression tasks with randomly chosen parameters. While all problem parameters are fixed, we consider a single iteration of the problem, and we draw many data sample batches of size N and simulate the two scheduling approaches. In Fig. 4 we plot the histogram of the mean performance across tasks, computed as $\frac{J^1(w_{k+1}^1)+J^2(w_{k+1}^2)}{2}$ for the two schemes. In our numerical evaluations, we surprisingly do not observe a significant loss due to the estimation procedure. This was

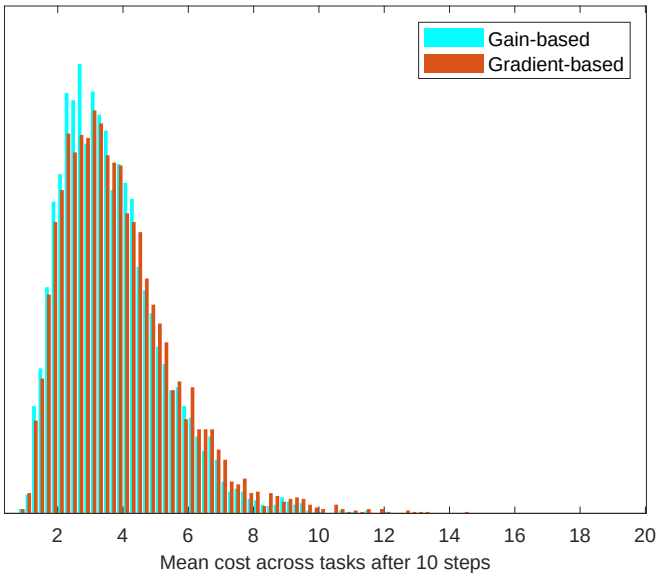

Fig. 5. Comparison between our scheduling approach based on estimating the gain in (29) versus the approach in (33) based on the magnitude of the gradient.

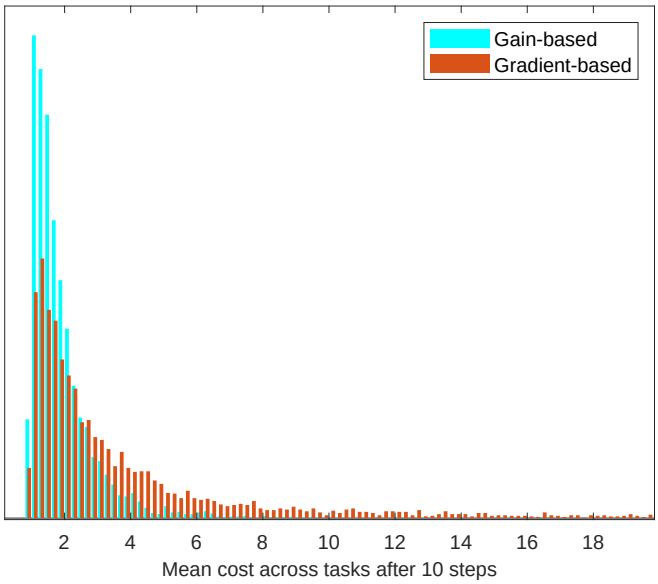

Fig. 6. Comparison between our scheduling approach based on estimating the gain in (29) versus the approach in (33) based on the magnitude of the gradient. The gradient-based approach leads to considerably worse performance when the stepsize is large.

consistently observed across different instances, reinforcing the usefulness of our scheme. In particular we observe that the difference between the two schemes is larger when the coordinates of x are more dependent. Our intuition is that in that case the covariances $\mathbb{E}xx^T$ are more different than identity matrices and it is harder to estimate the performance gains using data. The spike that appears in Fig. 4 is because of the multi-modal nature of the problem; sometimes the data-based approach has errors which lead to schedule tasks that lead to larger costs.

C. Comparison with gradient-based scheduling

We finally compare our scheduling scheme (31) based on estimating the performance gain across tasks in (29) with the simple scheduling strategy based on the magnitude of the gradients for each task (33). To illustrate how these approaches can lead to very different results we consider a specific setup of scheduling $m = 2$ identical tasks of dimensions $n = 2$, with covariances $\mathbb{E}xx^T = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$ (which affects the Hessian of the problem), the initial weights are $w_0 = 0$, and the true weights equal to $w^* = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$. We assume $N = 5$ data points are available at each iteration per task. First for stepsize $\varepsilon = 0.1$ the comparisons are shown in Fig. 5 again as histograms of the mean cost across the two tasks, where we observe very close performance among both approaches - even though the proposed gain-based one is better. Both approaches of course have the same communication burden - only one of the two tasks is selected to communicate/update at each step. Then for a slightly larger stepsize $\varepsilon = 0.2$ the comparisons are shown in Fig. 6. We observe that our approach performs significantly better than the gradient-based one. The improvements get even more significant as the setpsize increases. Our conclusion is that *the magnitude of the gradient is not a reliable measure for the informativeness of the data*. Our approach which is based on the more complex estimate of performance gain provides a more reliable and communication-efficient approach.

V. Concluding remarks

In this paper we examine the problem of solving multiple machine learning tasks concurrently over a network. We consider the problem of selecting which updates to be scheduled to meet capacity constraints. To exploit the informativeness of the data we examine the notion of performance gain and we illustrate numerically how this can be approximated from the data without further model knowledge. The approach is contrasted to other related works in the area of communication-efficient learning.

A limitation of our approach is that scheduling must be performed centrally with all data points available. In practical scenarios, such as in federated learning, the data are distributed across different agents and hence alternative approaches will be explored in future work. Ongoing work explores the use of the approach in more complex networks of learning agents, as well as other machine learning tasks beyond linear regression.

References

[1] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[2] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan, *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[3] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," *arXiv preprint arXiv:1704.05021*, 2017.

[4] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Sparse binary compression: Towards distributed deep learning with minimal communication," in *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2019.

[5] C.-Y. Lin, V. Kostina, and B. Hassibi, "Achieving the fundamental convergence-communication tradeoff with differentially quantized gradient descent," *arXiv preprint arXiv:2002.02508*, 2020.

[6] T. Chen, G. Giannakis, T. Sun, and W. Yin, "Lag: Lazily aggregated gradient for communication-efficient distributed learning," in *Advances in Neural Information Processing Systems*, pp. 5050–5060, 2018.

[7] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," *arXiv preprint arXiv:1909.13014*, 2019.

[8] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, pp. 4424–4434, 2017.

[9] D. Gündüz, P. de Kerret, N. D. Sidiropoulos, D. Gesbert, C. R. Murthy, and M. van der Schaar, "Machine learning in the air," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2184–2199, 2019.

[10] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *arXiv preprint arXiv:1909.07972*, 2019.

[11] C. Dinh, N. H. Tran, M. N. Nguyen, C. S. Hong, W. Bao, A. Zomaya, and V. Gramoli, "Federated learning over wireless networks: Convergence analysis and resource allocation," *arXiv preprint arXiv:1910.13067*, 2019.

[12] J. Ren, G. Yu, and G. Ding, "Accelerating dnn training in wireless federated edge learning system," *arXiv preprint arXiv:1905.09712*, 2019.

[13] J.-H. Ahn, O. Simeone, and J. Kang, "Cooperative learning via federated distillation over fading channels," *arXiv preprint arXiv:2002.01337*, 2020.

[14] W.-T. Chang and R. Tandon, "Communication efficient federated learning over multiple access channels," *arXiv preprint arXiv:2001.08737*, 2020.

[15] H. H. Yang, Z. Liu, T. Q. Quek, and H. V. Poor, "Scheduling policies for federated learning in wireless networks," *IEEE Transactions on Communications*, 2019.

[16] M. M. Amiri, D. Gunduz, S. R. Kulkarni, and H. V. Poor, "Update aware device scheduling for federated learning at the wireless edge," *arXiv preprint arXiv:2001.10402*, 2020.

[17] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Convergence time optimization for federated learning over wireless networks," *arXiv preprint arXiv:2001.07845*, 2020.

[18] M. M. Amiri and D. Gündüz, "Federated learning over wireless fading channels," *IEEE Transactions on Wireless Communications*, 2020.

[19] T. Sery and K. Cohen, "On analog gradient descent learning over multiple access fading channels," *arXiv preprint arXiv:1908.07463*, 2019.

[20] M. Eisen, M. M. Rashid, K. Gatsis, D. Cavalcanti, N. Himayat, and A. Ribeiro, "Control aware radio resource allocation in low latency wireless control systems," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7878–7890, 2019.

[21] K. Gatsis, M. Pajic, A. Ribeiro, and G. J. Pappas, "Opportunistic control over shared wireless channels," *IEEE Transactions on Automatic Control*, vol. 60, pp. 3140–3155, December 2015.

[22] M. H. Mamduhi, A. Molin, D. Tolić, and S. Hirche, "Error-dependent data scheduling in resource-aware multi-loop networked control systems," *Automatica*, vol. 81, pp. 209–216, 2017.

[23] O. Ayan, M. Vilgelm, M. Klügel, S. Hirche, and W. Kellerer, "Age-of-information vs. value-of-information scheduling for cellular networked control systems," in *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, pp. 109–117, 2019.

[24] T. Soleymani, S. Hirche, and J. S. Baras, "Optimal self-driven sampling for estimation based on value of information," in *2016 13th International Workshop on Discrete Event Systems (WODES)*, pp. 183–188, IEEE, 2016.

[25] F. Solowjow, D. Baumann, J. Garcke, and S. Trimpe, "Event-triggered learning for resource-efficient networked control," in *2018 Annual American Control Conference (ACC)*, pp. 6506–6512, IEEE, 2018.

[26] Z. Zhao, S. Cerf, B. Robu, and N. Marchand, "Event-based control for online training of neural networks," *arXiv preprint arXiv:2003.09503*, 2020.

[27] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.