

Optimal trade-off between accuracy and network cost of distributed learning in Mobile Edge Computing: An analytical approach

Lorenzo Valerio

Institute for Informatics and Telematics
National Research Council
Pisa, Italy

Email: lorenzo.valerio@iit.cnr.it

Andrea Passarella

Institute for Informatics and Telematics
National Research Council
Pisa, Italy

Email: andrea.passarella@iit.cnr.it

Marco Conti

Institute for Informatics and Telematics
National Research Council
Pisa, Italy

Email: marco.conti@iit.cnr.it

Abstract—The most widely adopted approach for knowledge extraction from raw data generated at the edges of the Internet (e.g., by IoT or personal mobile devices) is through global cloud platforms, where data is collected from devices, and analysed. However, with the increasing number of devices spread in the physical environment, this approach rises several concerns. The data gravity concept, one of the basis of Fog and Mobile Edge Computing, points towards a decentralisation of computation for data analysis, whereby the latter is performed closer to where data is generated, for both scalability and privacy reasons. Hence, data produced by devices might be processed according to one of the following approaches: (i) directly on devices that collected it (ii) in the cloud, or (iii) through fog/mobile edge computing techniques, i.e., at intermediate nodes in the network, running distributed analytics after collecting subsets of the data. Clearly, (i) and (ii) are the two extreme cases of (iii). It is worth noting that the same analytics task executed at different collection points in the network, comes at different costs in terms of traffic generated over the network. Precisely, these costs refer to the traffic generated to move data towards the collection point selected (e.g. the Edge or the Cloud) and the one induced by the distributed analytics process. Until now, deciding if to use intermediate collection points, and which one they should be in order to both obtain a target accuracy and minimise the network traffic, is an open question. In this paper, we propose an analytical framework able to cope with this problem. Precisely, we consider learning tasks, and define a model linking the accuracy of the learning task performed with a certain set of collection points, with the corresponding network traffic. The model can be used to identify, given the specification of the learning problem (e.g. binary classification, regression, etc.), and its target accuracy, what is the optimal level for collecting data in order to minimise the total network cost. We validate our model through simulations in order to show that setting, in simulation, the level of intermediate collection indicated by our model, leads to the minimum cost for the target accuracy.

I. INTRODUCTION

The ubiquitous and increasingly pervasive presence of smart devices, such as smart-phones, tables, IoT devices of any size and purpose, is considered a fundamental enabler for the successful development of the services that will characterise the future of the smart environments, such as smart cities,

smart factories, etc. In fact, this huge quantity of devices connected at the edge of the network is expected to produce massive amounts of data that, once properly processed, opens uncountable and highly valuable possibilities in terms of novel applications and services. For example, the data and the knowledge extracted from it are indeed the “glue” bonding together the physical city (i.e. the physical infrastructures of the city, such as lighting, heating, traffic control, water distribution, smart grids, and, extensively, people) and the cyber world of the smart city Internet services. The continual interaction between the cyber and the physical worlds mediated by data, enables not only to sense the status of the city, but also to act upon it, exploiting on the knowledge extracted from data. This is at the basis of the Cyber-Physical convergence concept [1] for smart cities.

According to current IoT/M2M paradigms [2], the typical work-flow adopted to process and extract knowledge from data generated by these devices is based on central cloud systems. This is the case, among others, of the ETSI M2M architecture [3]. However, there are concerns whether this approach will be sustainable in the long run. For example the projections of growth of the number of deployed IoT devices are exponential over the next years [4]. Together with data generated by personal users’ devices, which are also likely to be part of IoT applications, this is likely to make the amount of data generated at the edge of the network huge, making it impractical or simply impossible to transfer them to a remote cloud platform, e.g. it is foreseen by Cisco that the global mobile data traffic will increase sevenfold between 2016 and 2021. Mobile data traffic will grow at a compound annual growth rate (CAGR) of 47 percent from 2016 to 2021, reaching 49.0 exabytes per month by 2021 [5].

In addition, data might also have privacy and confidentiality constraints, which might make it impossible to transfer them to third parties such as global cloud platform operators. For example, in the manufacturing domain, data analytics is one of the cornerstones of the Industry 4.0 or Industrial Internet [6] concepts. However, most of the time industries will not be willing to move their data to some external cloud provider in-

frastructure, due to confidentiality reasons. On the other hand, they will not have the competences and resources to build and manage a private cloud platform. These trends and needs push towards a decentralisation of cloud platforms towards the edge of the network, where the emerging paradigms of Edge computing, such as Fog [7] and Mobile Edge Computing [8] paradigms, Cloudlets[9], can address these problems.

In the Mobile Edge/Fog Computing approach we consider in this paper, data is not necessarily sent to the cloud, but can be analysed (i) either at the same nodes that generate them, or (ii) at a set of intermediate nodes, each collecting a subset of the data. In both cases, we assume that a distributed data analytics algorithm is used, and therefore that nodes where data is analysed exchange information with each other to refine the analysis of the local data based on knowledge extracted by the other nodes. Let's consider again the smart manufacturing case we introduced before. By using our approach, nodes operating in the factory would collect data from the physical environment. Then, appropriate collection points would be selected, among the nodes owned by the factory itself (as a special case, each node might act as collection point solely for its own data). These nodes will collect each a subset of the data, and run a distributed data analytics process to extract knowledge. Ideally, by collaborating with each other, the accuracy of the distributed data analytics process would be equivalent to the accuracy of a central entity running in the cloud that would execute the same data analytics on the entire set of generated data. Collaboration among nodes in the manufacturing example would be clearly guaranteed by the fact that nodes are owned by the factory itself, and are thus configured and controlled by a unique entity. Similar approaches may also be used also in more general application scenarios, such as crowd sensing in smart cities [10], [11], by adopting appropriate incentive schemes.

In this paper, we develop an analytical model to identify the optimal number of collection points, i.e., the number of collection points that guarantee to achieve a target accuracy of the analytics process, at the minimum possible network cost. As we discuss in the paper, any configuration of the analytics process, from keeping the data in all the nodes that generate them, up to collecting them in a unique point in the cloud, can achieve any target accuracy. However, this will come at different costs in terms of network traffic.

In the cloud scenario, for example, the network traffic generated would be the traffic required to send all the generated data to the cloud. Conversely, in a scenario where the processing is performed directly on the devices that generate the data, the only traffic would be the one produced by the analytics process, i.e. by the fact that the devices have to communicate with each other in order to collaboratively perform the analytics task. Finally, in the Fog/Edge computing scenario the amount of network traffic generated is the sum of the traffic spent to move data from the devices to the Collection Points (CPs) and the traffic generated by CPs to communicate with each other in order to accomplish the analytics task. The total network traffic will largely depend on the number of

used Collection Points. Specifically, the more the CPs, the less the traffic required to move raw data. However, as the amount of data available at each CP will be lower, the traffic generated during the analytics process would be in general higher, intuitively because each local model computed by CPs would be less precise (as it would be computed over fewer data points) and thus a higher number of interactions between CPs would be needed to achieve the target accuracy. Therefore, for a given target accuracy, there is an optimal operating point, i.e. an optimal number of CPs, which minimises the cost in terms of network traffic.

In this paper we propose an analytical framework able to identify this optimal point. To this end we focus on general learning problems (e.g., binary classification or regression) and define a network traffic model that takes into account (i) the specific behaviour of the learning problem ii) the target accuracy we want to obtain after the learning process and iii) the number of nodes that generate raw data, and the amount of data generated by each node. Then, we show how to use the model to identify the optimal number of CPs. Simulation results show that the operating point indicated by our model is consistently the one that achieves the minimal network traffic for the target learning accuracy.

The rest of the paper is organised as follows. Section II presents related work. In Section III we describe the problem we consider and we define the network traffic model. In Section IV we describe the analytics approach we used to instantiate our model. In Section V we present the analytical solution of our model and its validation is presented in Section VI. Finally, Section VII concludes the paper.

II. RELATED WORK

In this paper we consider two different aspects that we combine together. On the one hand we want to collect data from the devices that generated it to an optimal number of Collection Points. On the other hand we want to execute a distributed analytics task on the collected data in order to obtain a model from data as much accurate as possible.

Let us consider the aspect of data collection. The problem of collecting data produced by devices, though from different perspectives and with different purposes with respect to this paper, has been widely investigated in the field of Wireless Sensor Networks (WSN). In that area, data aggregation (or data gathering) is mainly approached from the energy saving point of view. In fact in WSN the main issue of sensors is to save battery power, and as it is well known, the radio transmission is one of the most energy consuming activity done by a sensor. Therefore, much effort has been devoted to the design of energy efficient data aggregation protocols. Several in-network aggregation techniques for WSN are discussed in [12], which typically operate at the network level, needing to deal with the resource constraints of sensor nodes. A review of the existing literature more focused on energy efficiency is presented in [13], and on security in [14]. A more comprehensive review on data aggregation can be found in [15]. However, differently from the above literature, in this paper

with “data collection” we mean that devices that generated data simply send it to a certain number of collection points at the edge of the network, transparently from communication protocol or other details regarding the type of physical wireless network used. In other words, our work is placed at a higher level of abstraction where the communication channel and its quality are taken for granted.

To the best of our knowledge there are only few works that cope with the exact problem considered in this paper. Precisely, authors of [11] take into account the problem of accomplishing a learning task suitable for IoT scenarios. They propose a framework based on Hypothesis Transfer Learning with the aim at accomplishing a learning task without sending all the data to the cloud. Their main purpose is to both i) complete the learning task with a performance comparable to a centralised cloud solution, and ii) reduce the amount of traffic connected to the learning process. The closest work to the one proposed in this paper, at least in terms of scope, is presented in [10] where authors, for the first time, studied if there exists a trade off between the accuracy and the network performance connected to the execution of a distributed analytics task in IoT scenarios. Although the methodology followed in [10] is very empirical, the paper shows that there could exist such type of trade-off and that, once found, it is possible to save network resources. In this paper we follow a different approach. We defined an analytical cost model in which we are able to relate both the accuracy of a distributed analytics task and the network traffic that is triggered by it. Differently from the literature, with our model we are able to i) predict in advance the network traffic generated by the analytics process and ii) control its accuracy. Moreover, with our model we are able to find the optimal trade off between accuracy and network traffic.

III. PROBLEM DEFINITION AND COST MODEL

In this paper we consider a scenario in which there are m_0 devices (e.g. high-end sensors of workers’ devices in a smart factory, hand held devices in a smart city, etc.) collecting and storing on their local storage, some kind of data. In this paper we also assume that all the devices collect the same amount of data n_0 , therefore the total amount of data collected by all the devices is $N = m_0 n_0$. Although this is a simplifying assumption taken for pure modelling reasons, we think it does not affect the generality of the line of reasoning we keep all along this paper. In fact our model is still a reasonable approximation if n_0 is the average number of data points held by each device, since the network traffic generated to transfer data to CPs would be exactly the same. Moreover, we consider all the devices in the system are computationally able to process the amount of data they hold. The data analytics process can be executed: i) in the cloud, ii) on the m_0 devices or iii) on a subset $m_1 \leq m_0$ of collection points that evenly collect an equal number of data. Specifically, each CP collects $n_1 = \frac{n_0 m_0}{m_1}$ data points. In the following, we will treat all three cases with the same model, as case i) corresponds to $m_1 = 1$, while case ii) corresponds to $m_1 = m_0$.

From the network point of view, the amount of network traffic generated by each one of the above mentioned cases can be modelled as a sum of two terms: on the one hand there is the network traffic generated to transfer data from nodes generating the data to the CPs. On the other hand there is the network traffic related to the specific distributed algorithm used by CPs in the data analytics process.

More formally, we model the network cost connected to a distributed learning task as follows:

$$C = C_A + C_D \quad (1)$$

$$= C_A + (m_0 - m_1)n_0 \quad (2)$$

where C_A represents the network traffic generated by the distributed data analytics process performed by the CPs and C_D is the network traffic generated to send data to the CPs. Formally, C_D is defined as the number of data points that must be transferred from $m_0 - m_1$ devices to m_1 CPs. As far as C_A is concerned, we point out that its exact definition is dependent on the distributed data analytics task. More details about C_A will be discussed in Section IV.

As one may notice, the network cost C computed according to the model defined in Equation (1) is strongly dependent on where the data analytics task is executed. For example, if we decide to execute the analytics directly on all the m_0 devices that generated the data then $m_1 = m_0$ and $C = C_A$ because the only network traffic spent is the one triggered by communication needed by the distributed analytics process. Conversely, if $m_1 = 1$ then data is processed on a single device and we have $C = C_D$ because the only traffic generated is to move data from the generators to a single CP. In the paper we assume that this configuration coincides with a centralised solution implemented in the cloud. For each possible configuration of m_1 ranging from 1 to m_0 , the value of C_A changes accordingly. As it will become clear in Section IV, this is due to the fact that the traffic generated by C_A depends on the amount of data present at CPs.

Without loss of generality, to have a single control parameter in the model, in the following we define γ as

$$\gamma = \frac{m_0}{m_1}$$

representing the level at which we group data at collection points (we call it *grouping parameter*). Clearly, it holds that $\gamma \in [1, m_0]$, and Equation (1) becomes:

$$C(\gamma) = C_A(\gamma) + C_D(\gamma) \quad (3)$$

$$= C_A(\gamma) + (m_0 - \frac{m_0}{\gamma})n_0 \quad (4)$$

The goal of this paper is to find the optimal value of γ for a given accuracy, i.e. the value of γ that minimises the resulting network traffic. Mathematically, we want to solve the following problem.

$$\begin{aligned} \min_{\gamma} \quad & C_A(\gamma) + C_D(\gamma) \\ \text{s.t.} \quad & \end{aligned} \quad (5)$$

$$1 \leq \gamma \leq m_0$$

Before presenting all the mathematical details of the solution of our model, in the next section we describe the class of data analytics tasks (i.e., learning) that we consider in the rest of the paper.

IV. DISTRIBUTED LEARNING ALGORITHM

In this section we briefly present the details about the “Distributed Stochastic Variance Reduced Gradient” (DSVRG) [16] method we consider as a specific data analytics task. As explained in the following, this has been selected as it is a reference method for learning algorithms, and because the analytical form to describe the cost in terms of generated network traffic can be derived from the literature. Any data analytics method satisfying this property can be analysed as we do for DSVRG in this paper. DSVRG is a distributed stochastic optimization algorithm based on the very well known Stochastic Gradient Descent Learning Algorithm. For the sake of clarity, before presenting the details regarding DSVRG, we introduce the general learning problem that can be solved by this algorithm.

A. General Learning Problem

The general distributed learning problem that can be solved by DSVRG belongs to the class of the so called “Empirical Risk Minimization (ERM)” problems. Precisely, suppose that there exists a set of i.i.d. samples $\{s_1, s_2, \dots, s_N\}$ belonging to an unknown distribution $\mathcal{D}$. Since in this paper the learning setting considered is supervised learning, each data point s is a pair (x, y) where $x \in \mathbb{R}^d$ is a feature vector and $y \in \mathbb{R}$ is either a continuous value (for regression problems) or a discrete value (for classification problems). The ERM problem is defined as follows:

$$\min_{w \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^N f(w, s_i) \quad (6)$$

In Equation (6) $f(w, s_i)$ is a loss function, i.e. the error of the model obtained through learning in estimating the real values of y for the data points s . w is a vector with the parameters of the learning algorithm. Therefore, solving Equation (6) means finding the model parameters that minimise the error with respect to the set of samples s_i . Examples of loss functions that can be used in this very general setup include, among others: the square loss for linear regression

$$f(w, s) = (x^T w - y)^2 \quad (7)$$

and the logistic loss for binary classification

$$f(w, s) = \log(1 + \exp(-y(x^T w))) \quad (8)$$

In order to improve the generalisation abilities (i.e. the ability to correctly predict unknown values of y for data points s_i other than those used in the training phase) of the model learnt by the learning algorithm that solves (6), a common practice is to add to the objective function an additional term, called regularisation term. The physical effect of the regularisation term is to obtain a model which is more robust to the over fitting problem. The most commonly used form is

the Tikhonov regularisation [17], which transforms Equation (6) as follows.

$$\min_{w \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^N f(w, s_i) + \frac{\lambda}{2} \|w\|^2 \quad (9)$$

where λ is the “regularisation parameter” that controls how much the solution is influenced by the regularisation term.

In the following section we present the details of DSVRG, the stochastic optimization algorithm we use to learn the model’s parameters w from data.

B. DSVRG

In order to find the solution of the optimization problem defined in (9), in this paper we use DSVRG. DSVRG is a distributed optimization algorithm that assumes the presence of several computing devices and a central unit whose task is to coordinate the communication between the devices and monitor the overall learning process. A typical assumption is that the central unit (also called center) has complete knowledge of the system, i.e. information about devices, about the data they hold, etc.

DSVRG, is the distributed implementation of the Stochastic Variance Reduced Gradient (SVRG) algorithm [18]. SVRG algorithms represent an improvement of the well known Stochastic Gradient Descent method (SGD) [19], which is an iterative optimization algorithm used to find a solution to problem (9). At each iteration t , a data point s_t is randomly picked up from the dataset and it is used to update the “current estimate” of the solution as follows:

$$w_t = w_{t-1} - \eta_t \nabla f(w_{t-1}, s_t)$$

where $\nabla f(w_{t-1}, s_t)$ is the value of the gradient of f computed with respect to w and η_t is the learning rate. In this way, at each data point the solution is updated by a quantity that is proportional to the gradient of the prediction error committed by the solution at iteration $t - 1$. After a certain number T of iterations, SGD comes up with a solution. SGD has proven to be a very good and cheap learning algorithm, i.e. at each iteration it relies on a single derivative of the loss function that is not computationally expensing to compute. However, its convergence to the optimal solution could be very slow. This is due to fact that the randomness of the data selection process may introduce a high variance that slows down the convergence towards the solution. SVRG (and its distributed version) solve the variance issue of SGD. In the following we describe the DSVRG algorithm and we refer to Algorithm 1. The description of DSVRG can be split in two logically separate procedures, one executed by the “center” and one executed by each device. Note that in practice, the “center” can be any of the devices involved in the learning process, thus there is no need to have an additional device to complete successfully the learning task. However, just for the sake of presentation we describe DSVRG considering the center and the devices as separate entities.

At the beginning, the center broadcasts an initial guess of the solution w_0 to all the devices. Each of them computes

Algorithm 1 DSVRG

```
1: procedure DSVRG( $w^0, \{x_i\}_{i=1,\dots,N}, m, \eta, T, R, N$ )  $\triangleright$ 
2:    $k \leftarrow 1$ 
3:   for  $r = 0, 1, 2, \dots, R-1$  do
4:     Center sends  $w^r$  to each device
5:     for device  $j = 0, 1, 2, \dots, m$  do  $\triangleright$  This is done in
       parallel by all devices
6:        $h_j^r \leftarrow \sum_{s \in S_j} \nabla f(w^r, s)$ 
7:       Send  $h_j^r$  to the center
8:     end for
9:     Center computes  $h^r \leftarrow \frac{1}{N} \sum_{j=1}^m h_j^r$ 
10:    Send  $h^r$  to device  $k$ 
11:     $w^{r+1} \leftarrow \text{SS-SVRG}(w^r, h^r, k, \eta, T)$ 
12:     $k \leftarrow k + 1$ 
13:  end for
14: end procedure
```

Algorithm 2 Single-Stage SVRG: SS-SVRG

```
1: procedure SS-SSVRG( $\tilde{w}, h, k, \eta, T$ )  $\triangleright$ 
2:    $w_0 = \tilde{w}$  and  $\bar{w}_0 = \mathbf{0}$ 
3:   for  $t = 0, \dots, T-1$  do
4:      $w_{t+1} = w_t - \eta(\nabla f(w_t, s_t) - \nabla f(\tilde{w}, s_t) + h)$   $\triangleright$ 
       Device  $k$  samples an instance  $s_t$  from its local data and
       computes the update
5:      $\bar{w}_{t+1} = \frac{w_{t+1} + t\bar{w}_t}{t+1}$ 
6:   end for
7:   Return  $\bar{w}_{t+1}$  to center
8: end procedure
```

the gradient in batch over all data points contained in their local datasets and sends it back to the center which collects them and computes the average batch gradient h (lines 4-8). At this point the center selects a single device k and sends to it the batch gradient h (lines 9-11). Device k , once received this information, executes the SS-SVRG procedure listed in Algorithm 2. At each iteration t device k samples a data point s_t and updates the current estimate of the model w (line 4). Note that, differently from the update equation of SGD, the SVRG method computes the update of the model including information regarding i) the local gradient ($\nabla f(w_t, s_t)$), ii) the gradient of the global current solution ($\nabla f(\tilde{w}, s_t)$) and iii) the average batch gradient (h). This has the effect of reducing the variance of the updates. Once computed the update, device k updates the running average of the global current solution $\bar{w}$ (line 5). Finally, when the maximum number of iterations T is reached (typically the size of the local dataset), device k returns to the center the global current solution $\bar{w}$, the control comes back to the center procedure that repeats the process (lines 7-8). This procedure is repeated for a number R of rounds.

We selected this algorithm not only because of its well assessed good performance in practice [20] but also because it is accompanied by theoretical guarantees. In fact this algorithm is provided with a bound regarding the amount of communication

generated during the distributed learning phase. Precisely it is shown that DSVRG is able to converge to an ε -accurate solution in at most R rounds. We provide a precise definition of ε accuracy in the following of the section. To discuss R , suffices to say that ε is a measure of the accuracy obtained by the learning algorithm after the training phase (i.e., after Algorithm 1 and 2). The relationship between the ε -accuracy and the number of communication rounds between the center and the devices is expressed by

$$R = (1 + \frac{\kappa}{n}) \log_2(\frac{1}{\varepsilon}) \quad (10)$$

where n is the size of the local dataset held by each device and κ is the condition number of the problem. The condition number is defined as the ratio between L and μ , respectively the first and the last eigenvalue of S_l . The information brought by κ regards how ill-posed is the optimization problem at hand. Computing the exact value of κ is computationally expensive, but in the machine learning community it is typically considered to be proportional to $O(\sqrt{N})$. The ε -accuracy is defined as

$$|f(\bar{w}, s) - f(w^*, s)| \leq \varepsilon$$

and means that given a point s the solution learnt by the learning algorithm is far from to the optimal and unknown solution w^* at most ε . Practically, if ε is set to 10^{-4} , it means that the predictions made by the learnt model and those made by the optimal one are precise at the fourth decimal place. It is important to note that the value of ε must not be confused with the value of the generalisation performance (i.e. the loss ℓ) of the learning algorithm. Clearly, these quantities are strictly related to each other because a more accurate search for the optimal solution of the learning algorithm translates into a better prediction performance of the algorithm. However, quantifying in advance the exact relation between the two is very difficult (if not impossible) and strongly problem dependent. This means that if we set the value of ε , we cannot know in advance, i.e. without running the learning algorithm, what value of prediction loss will produce for a certain learning problem. Nevertheless, as we will show it can be used to control the behaviour of the algorithm.

Finally, for this algorithm, the amount of communication generated in a single round, computed as the number of gradient vectors of dimension d (i.e. the same dimensionality of a data point s) that must be sent over the network is defined as:

$$C_R = 2(m-1) \quad (11)$$

where m is the number of devices involved in the learning process. Note that this equation considers that one device acts as center and that the number of vectors exchanged between devices is 2, i.e. the current estimate $\bar{w}$ and the batch gradients h . Summarising, given a specific learning problem (e.g. binary classification) and a dataset distributed over different devices, we can calculate in advance the total amount of network traffic that the algorithm generates to reach an ε -accurate solution:

$$C_A = C_R * R = 2(m-1)(1 + \frac{\kappa}{n}) \log_2 \frac{1}{\varepsilon} \quad (12)$$

As it will be clear in the next sections this theoretical guarantee about the relationship between the accuracy and the network traffic generated by the algorithm plays a fundamental role in our model. In fact it is exploited to evaluate the overall network cost of the entire learning process for each possible number of collection points, in order to find the optimal one.

V. COST MODEL ANALYSIS

In this section we present the analytical solution of the optimization problem defined in Section III for the case of the learning algorithm presented in Section IV. Firstly, let us rewrite the cost model expanding $C_A(\gamma)$:

$$\begin{aligned} C(\gamma) &= C_A(\gamma) + C_D(\gamma) \\ &= 2\left(\frac{m_0}{\gamma} - 1\right)\left(1 + \frac{\kappa}{\gamma n_0}\right) \log_2 \frac{1}{\varepsilon} + (m_0 - \frac{m_0}{\gamma})n_0 \end{aligned} \quad (13)$$

As we can notice from (13) both C_A and C_D are affected by the grouping parameter γ . Now we analyse the analytical behaviour of the function $C(\gamma)$ (i.e. when it grows or decays, if it admits maxima or minima) in order to understand if there exists an optimal value of γ in the interval $[1, m_0]$. This would allow us to identify the optimal grouping level, i.e. the optimal number of Collection Points. To this end we compute the derivative of $C(\gamma)$ with respect to γ

$$C'(\gamma) = \frac{m_0 n_0 \gamma (n_0 - 2 \log_2(1/\varepsilon)) + 2\kappa \log_2(1/\varepsilon)(\gamma - 2m_0)}{n_0 \gamma^3} \quad (14)$$

and we calculate for what value of γ the derivative is null

$$\hat{\gamma} = \frac{4 \log_2(1/\varepsilon) \kappa m_0}{2\kappa \log_2(1/\varepsilon) - 2 \log_2(1/\varepsilon) m_0 n_0 + m_0 n_0^2} \quad (15)$$

From equation 15 we get that $C'(\gamma) = 0 \iff \gamma = \hat{\gamma}$. Thus let us study the sign of $\hat{\gamma}$.

Case: $\hat{\gamma} > 0$: From the numerator of equation 14 and under the condition $\kappa < m_0 n_0$ (which is the setting considered in this paper) we can observe that $\hat{\gamma} > 0$ only when

$$n_0 - 2 \log_2 \frac{1}{\varepsilon} > 0 \quad (16)$$

Under this condition we derive the behaviour of the function $C(\gamma)$.

- When $\gamma > \hat{\gamma}$ we have that $C'(\gamma) > 0$. This is due to the fact that the first term of the numerator $m_0 n_0 \gamma (n_0 - 2 \log_2(1/\varepsilon))$ is positive while the second term $2\kappa \log_2(1/\varepsilon)(\gamma - 2m_0)$ is negative. Thus the former dominates the latter and this determines the positive sign of the derivative.
- Conversely, when $0 < \gamma < \hat{\gamma}$ we have that $C'(\gamma) < 0$. In fact as γ approaches 0 from $\hat{\gamma}$ the sign of the numerator of $C'(\gamma)$ is dominated by $2\kappa \log_2(1/\varepsilon)(\gamma - 2m_0)$ which is negative because $\gamma < 2m_0$ by definition. Thus the sign of the derivative is negative.

Summarising, we can conclude that $\hat{\gamma}$ is a minimum for the function $C(\gamma)$. If we consider the range of admissible values for γ , the physical interpretation of what presented in this case is that when $\hat{\gamma} = 1$, $C(\gamma)$ is increasing and the best option is not to move data from the devices that generated it.

Conversely, when $\hat{\gamma} = m_0$, $C(\gamma)$ is decreasing thus the best option is send data in the cloud.

1) *Case: $\hat{\gamma} < 0$:* Under the same assumption $\kappa < m_0 n_0$, we have that $\hat{\gamma} < 0$ when

$$n_0 - 2 \log_2 \frac{1}{\varepsilon} < 0 \quad (17)$$

Since by definition $\gamma \in [1, m_0]$ the only interesting case for our problem is $\gamma > 0 > \hat{\gamma}$. Following the same line of reasoning used in the previous case we obtain that the numerator of $C'(\gamma)$ is negative, therefore the sign of the entire derivative is negative. This means that $C(\gamma)$ is decreasing for increasing values of γ , thus the only option that minimises the network traffic is to send all the data in the cloud.

A. Relationship between γ and ε

From this analysis, it clearly emerges an important relation between ε and γ . This is due to the fact that the parameter ε affects the number of communication rounds performed by the distributed learning algorithm, where the more rounds are performed the more accurate the model learnt is. Moreover, it is clear that a given ε accuracy corresponds to an optimal aggregation level. From a system design point of view, one interesting problem would be to identify, given the specification of a scenario, the range of ε accuracies one can obtain (and the corresponding γ s) for which the optimal number of CPs is higher than 1, i.e., sending all data to the cloud is not the optimal configuration. In other words, we can look for the values of ε for which the corresponding optimum γ is in the interval $[1, m_0]$. Formally, the lowest accuracy obtainable for a problem is:

$$\begin{aligned} \hat{\gamma} &\geq 1 \\ \log_2 \frac{1}{\varepsilon} &\geq \frac{mn^2}{2\kappa(2m_0 - 1) + 2m_0 n_0} \\ \varepsilon &\leq 2^{-\frac{m_0 n_0^2}{2\kappa(m_0 - 1) + m_0 n_0}} \end{aligned} \quad (18)$$

while the highest accuracy obtainable is:

$$\begin{aligned} \hat{\gamma} &\geq m_0 \\ \log_2 \frac{1}{\varepsilon} &\geq \frac{m_0 n_0^2}{2\kappa(m_0 - 1) + m_0 n_0} \\ \varepsilon &\leq 2^{-\frac{m_0 n_0^2}{2\kappa + 2m_0 n_0}} \end{aligned} \quad (19)$$

The physical meaning of these relations is that, given the specification of a problem we can compute the lower and the highest ε accuracy we can obtain at the extreme limit of the range $[1, m_0]$. The direct consequence of this is that, for example, if we need an accuracy better than $2^{-\frac{m_0 n_0^2}{2\kappa + 2m_0 n_0}}$ the only option is to collect data in the cloud.

VI. MODEL VALIDATION

In this section we present the validation of our model. The claim we want to prove through simulations is that our model is able to identify the optimal aggregation level that allows a distributed learning algorithm to reach an accuracy of ε

while minimising the network traffic generated by the learning process.

To this end we solve a binary classification problem on two datasets, one synthetic and one coming from the real world. The loss function used in our simulations is the logistic function defined in equation (8). The synthetic dataset contains 10000 points divided in two classes. Each class of points is distributed according to a Normal distribution $\in \mathbb{R}^{50}$ (i.e. we used the same number of features of the real world dataset). The two Normal distribution have different means and standard deviation equal to 1. We used this dataset in order to have a controlled environment for our simulations, i.e. in this way we can control the characteristics of the dataset such as the number of clusters of points for each class (here is 1), the reciprocal placement of the two cluster of points, etc. The second dataset we considered in this paper is connected to the forest cover type prediction problem (CovType)¹. It is a publicly available dataset that contains 581012 observations. Each observation is a vector of 54 features containing both cartographic and soil information, corresponding to a $30 \times 30m$ area of forest. The learning task is to use this information to predict what is the main kind of tree covering the area. The dataset contains 7 classes. Since in this paper we test our model with a binary classification problem we selected two out of seven classes, i.e. classes 5 and 6. Thus the dataset size in our simulations is 26000 points. This dataset might be seen as an instance of our motivating scenario described in Section I. In case of an operator that needs to perform an environmental monitoring task through a number of mobile devices communicating with each other, our model could be used to identify the optimal number of data collector among the nodes participating to the monitoring campaign.

In order to validate our model we used the following procedure. Each dataset is split into training and test sets, with a proportion (80%, 20%). We set a number m_0 of devices, each one holding n_0 observations of the training set. We selected a target accuracy ε and imposed, as it is common practice in the machine learning field that κ is equal to $\sqrt{N}$, where N is the size of the entire dataset. Then we run the distributed learning algorithm for different values of γ in the range $[1, m_0]$. For each value of γ we measured the prediction loss computed on the test set and the amount of network traffic triggered by each level of aggregation γ . Note that the prediction loss is defined as the percentage of misclassified points by the trained model. We repeated this procedure 10 times and averaged the results for the loss values. Confidence intervals have been calculated but not shown because they are too tight. From the simulative results we expect that for a fixed epsilon there is only one value of γ corresponding to the cheapest learning process in terms of network traffic. We consider the model validated if the optimal value of γ obtained from our model corresponds, in simulation, to the value of γ resulting in the minimum network cost.

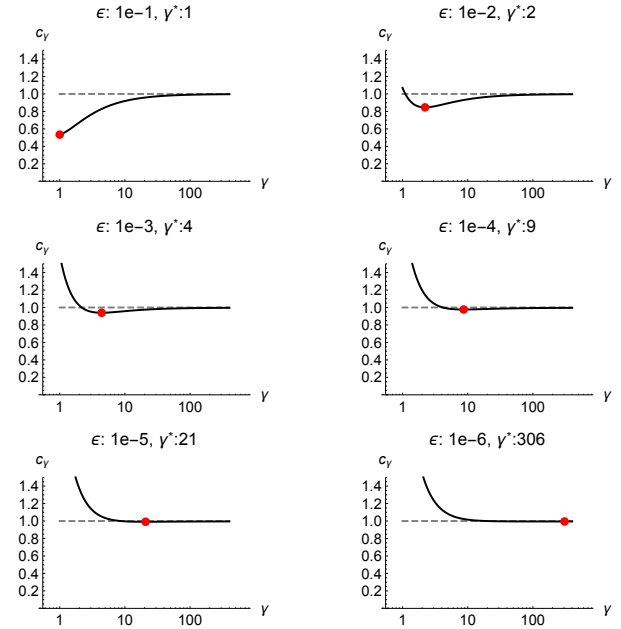


Fig. 1. Synthetic dataset. The six plot represent the predication of network traffic for different values of ε . In each plot, the black curve shows the percentage of network cost predicted for different values of γ . The red dot is the optimal γ predicted by the model. Dashed grey curve is the network traffic the would be generated if we sent all data to the cloud.

A. Synthetic dataset

Let us now present the validation of our model using the Synthetic dataset. In this simulated scenario we consider $m_0 = 200$ devices that generate $n_0 = 40$ labelled observations, thus the entire dataset size is $N = 8000$. In this scenario, according to its definition, $\kappa = 89.4427$. Given this initial configuration we compute the range of ε for which the optimal γ falls in the interval $[1, m_0]$: $\varepsilon \in [1.11165 \times 10^{-6}, 0.0789803]$. Moreover, in this specific case, in order to reach an accuracy better than $\varepsilon < 10^{-6}$ data must be sent in the cloud. Fig. 1 shows, for each ε in this range, the network cost $C(\gamma)$ coming from the execution of the learning algorithm for values of γ in the range $[1, m_0]$. In the plots, for each target ε the red dot shows the optimal operating point according to our model. As we can see for each ε there is a different optimal γ and when the target accuracy is equal to the lower limit of the range of ε , the best options becomes the cloud. Conversely, when we can tolerate worse accuracies, i.e. $\varepsilon = 10^{-1}$ we can perform the learning task directly on the devices that generate the data.

Let us now validate the model. To this end we select one of the above cases, i.e. $\varepsilon = 10^{-3}$, and we perform the learning process for different values of $\gamma \in [2, 7]$. We selected this interval in order to show the behaviour of the system, before and after the optimal value identified by our model (i.e. $\gamma = 4$). As we can see from Table I, increasing the value of γ has the effect of collecting more data on few devices. This results in a decreasing number of communication rounds needed to reach the target accuracy. As a consequence, the cost of the algorithm decreases while the cost of moving data increases. However,

¹Dataset available at <https://archive.ics.uci.edu/ml/datasets/Covertype>

TABLE I
SYNTHETIC DATASET. EMPIRICAL BEHAVIOUR OF THE LEARNING ALGORITHM FOR $\epsilon = 10^{-3}$. THE BOLD FIGURES CORRESPOND TO THE OPTIMAL SOLUTION.

γ	rounds	CPs	C_A	C_D	C_γ
2	21	100	0.51975	0.5	1.01975
3	17	66	0.27625	0.67	0.94625
4	15	50	0.18375	0.75	0.93375
5	14	40	0.1365	0.8	0.9365
6	13	33	0.104	0.835	0.939
7	13	28	0.08775	0.86	0.94775

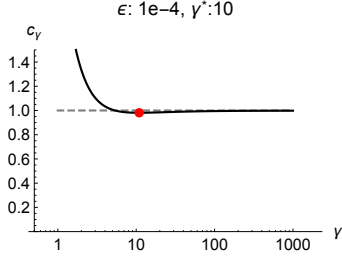


Fig. 2. CovType dataset. The plot show the prediction of network traffic for $\epsilon = 10^{-4}$. The black curve shows the percentage of network cost predicted for different values of γ . The red dot is the optimal γ predicted by the model. Dashed grey curve is the network traffic the would be generated if we sent all data to the cloud.

the solution of our model corresponds to the minimal network cost.

B. Covtype dataset

In order to further confirm the results obtained for the synthetic dataset, we repeated the validation procedure for the CovType dataset. In the initial configuration considered here there are $m_0 = 500$ devices that collected $n_0 \approx 43$ labelled observations each and in this case the total size of the training set is $N = 21488$. In this case the condition number is set to $\kappa = 146.588$ and the ϵ range for which the optimal γ falls in the interval $[1, m_0]$ is $(3.76093 \times 10^{-7}, 0.148695]$. Also in this case for $\epsilon \leq 3.76093 \times 10^{-7}$ the optimal collection point is the cloud. For this dataset we fixed the target accuracy to $\epsilon = 10^{-4}$. As we did before, we varied $\gamma \in [1, m_0]$ and analysed the trend of C_γ and finally we identified the optimal value of $\gamma = 10$, as shown in Fig. 2. Looking at Table II, also in this case the value of γ found by our model corresponds to the distributed learning execution of lowest network cost. Moreover, also in this case we see the very same behaviour in terms of number of rounds w.r.t. γ and the same interplay between C_A and C_D . Finally, let us consider the loss as a function of γ . As we can see from Fig. 3b the loss is not affected by γ as expected. Note that the small variations of the loss curve are due to statistical fluctuations related to the fact that we are dealing with a real world dataset. This concludes the model validation

C. Connection between γ , ϵ and the loss

In principle, the key target parameter for configuring the system should not be the ϵ -accuracy, but the prediction loss,

TABLE II
COVTYPE DATASET. EMPIRICAL BEHAVIOUR OF THE LEARNING ALGORITHM FOR $\epsilon = 10^{-4}$. THE BOLD FIGURES CORRESPOND TO THE OPTIMAL SOLUTION.

γ	rounds	CPs	C_A	C_D	C_γ
2	35	250	0.81115	0.5	1.31115
3	28	166	0.430007	0.668	1.09801
4	24	125	0.276992	0.75	1.02699
5	22	100	0.202718	0.8	1.00272
6	20	83	0.152643	0.834	0.986643
7	19	71	0.12379	0.858	0.98179
8	18	62	0.102197	0.876	0.978197
9	18	55	0.0904691	0.89	0.980469
10	17	50	0.0775316	0.9	0.977532
11	17	45	0.0696203	0.91	0.97962
12	17	41	0.0632911	0.918	0.981291
13	16	38	0.0551005	0.924	0.979101
14	16	35	0.0506329	0.93	0.980633
15	16	33	0.0476545	0.934	0.981655
16	16	31	0.0446761	0.938	0.982676
17	15	29	0.0390916	0.942	0.981092
18	15	27	0.0362993	0.946	0.982299
19	15	26	0.0349032	0.948	0.982903

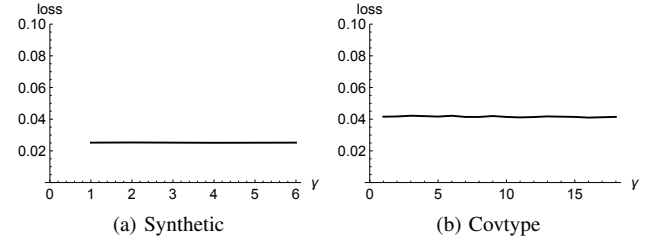


Fig. 3. Prediction loss computed on the test set for different values of γ

as it is the prediction loss that the operator of the system would practically observe. As mentioned before, the relationship between the prediction loss and the accuracy ϵ is in general unknown, and dependent on the specific dataset. However, since for a given ϵ all classifiers derived at any value of γ achieve the same ϵ , we hypothesise that they also achieve the same prediction loss. If this is the case, we can still use our model to find the optimal operating point, starting from a target prediction loss (instead of a target ϵ). In case the dependency between prediction loss and ϵ is known, it is sufficient, for a target prediction loss, to obtain the corresponding ϵ and then solve our model. If not, one could set a starting value of ϵ , derive the optimal γ , and measure the prediction loss achieved. If this is not acceptable, one would modify the value of ϵ and iterate the process. Note that this would be less costly than using always a specific value of γ for each candidate ϵ until the sought prediction loss is achieved, because in each iteration our model would guarantee that the minimum cost is always achieved.

To validate our hypothesis, Fig. 3a and 3b show the prediction losses for a range of values of γ , when the value of epsilon is set to $10^{-3}, 10^{-4}$ in the synthetic and CovType datasets, respectively. It is clear that the prediction loss does not change with γ , and this suggests that our hypothesis is correct.

VII. CONCLUSION

In this paper we consider an IoT scenario where edge devices (such as IoT or personal mobile devices) generate data that must be processed in order to extract some knowledge. Examples can be found both in professional applications such as smart manufacturing, as well as in generic applications for smart cities and smart environments. In many of these examples, it is not appropriate (either for scalability or confidentiality reasons) to move all raw data to a central cloud where they are analysed. Instead, fog/mobile cloud computing approaches can be applied to keep data closer to where they are generated, while still extracting useful knowledge out of them through distributed data analytics. An open question is identifying what is the optimal data level at which to group raw data (i.e., the optimal number of nodes where to collect subsets of the raw data) for which the data analytics process achieves a target accuracy, while minimising the total network traffic. To this end we propose an analytical framework able to cope with this problem. We exemplify the complete derivation of the optimal operating point in the case of one of the reference distributed learning algorithms, named DSVRG. Exploiting analytical expressions of the number of data exchanged between the nodes to run the algorithm, we derive a closed form expression for the network traffic required to achieve a target accuracy. We thus derive an analytical expression of the optimal operating point, and study the properties of the cost function in the range of possible grouping configurations (i.e., for the possible values of the number of Collection Points). The analysis highlights cases where the optimal operating point is either moving all the data in the cloud, or keeping them local where they have been generated. We also highlight conditions for which intermediate levels of grouping achieve the least network traffic. Results of our model are validated by simulation both with synthetic and with real datasets, showing that our model is very precise in predicting the optimal operating point. Therefore, we conclude that our model provides a practical and reliable tool to configure fog computing systems for distributed data learning.

ACKNOWLEDGMENT

This work is partly funded by the EC under the H2020 REPLICATE (691735), SoBigData (654024) projects and and AUTOWARE (723909).

REFERENCES

- [1] M. Conti, S. K. Das, C. Bisdikian, M. Kumar, L. M. Ni, A. Passarella, G. Roussos, G. Tröster, G. Tsudik, and F. Zambonelli, "Looking ahead in pervasive computing: Challenges and opportunities in the era of cyber-physical convergence," *Pervasive and Mobile Computing*, vol. 8, no. 1, pp. 2 – 21, 2012.
- [2] E. Borgia, "The internet of things vision: Key features, applications and open issues," *Computer Communications*, vol. 54, pp. 1 – 31, 2014. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0140366414003168>
- [3] ETSI TC M2M. ETSI TS 102 690 v2.1.1 (2013-10) – Machine-to-Machine Communications (M2M); Functional Architecture, 2013. [Online]. Available: <http://tinyurl.com/hemc3po>
- [4] J. Manyika, M. Chui, J. Bughin, R. Dobbs, P. Bisson, and A. Marrs. (McKinsey Global Institute, 2013) Disruptive technologies: Advances that will transform life, business, and the global economy. [Online]. Available: <http://tinyurl.com/zpkr7m6>
- [5] Cisco. (2017) Cisco visual networking index: Global mobile data traffic forecast update, 2016–2021. [Online]. Available: <http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html>
- [6] H. Kagermann, W. Wahlster, and J. Helbig, "Recommendations for implementing the strategic initiative industrie 4.0," German National Academy of Science and Engineering, Tech. Rep., 2013.
- [7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12. New York, NY, USA: ACM, 2012, pp. 13–16. [Online]. Available: <http://doi.acm.org/10.1145/2342509.2342513>
- [8] P. G. Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," *ACM Sigcomm Computer Communication Review*, 2015.
- [9] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct 2009.
- [10] L. Valerio, A. Passarella, and M. Conti, "Accuracy vs. traffic trade-off of learning iot data patterns at the edge with hypothesis transfer learning," in *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, Sept 2016, pp. 1–6.
- [11] —, "Hypothesis transfer learning for efficient data computing in smart cities environments," in *International Conference on Smart computing (SMARTCOMP 2016)*, St. Louis, Missouri, IEEE, Ed., 2016.
- [12] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: A survey," *Wireless Commun.*, vol. 14, no. 2, pp. 70–87, Apr. 2007. [Online]. Available: <http://dx.doi.org/10.1109/MWC.2007.358967>
- [13] R. Rajagopalan and P. K. Varshney, "Data-aggregation techniques in sensor networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 8, no. 4, pp. 48–63, Fourth 2006.
- [14] H. Alzaid, E. Foo, and J. G. Nieto, "Secure data aggregation in wireless sensor network: A survey," in *Proceedings of the Sixth Australasian Conference on Information Security - Volume 81*, ser. AISC '08. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2008, pp. 93–105. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1385109.1385127>
- [15] P. Jesus, C. Baquero, and P. S. Almeida, "A survey of distributed data aggregation algorithms," *IEEE COMMUNICATION SURVEYS and TUTORIALS*, vol. 17, no. 1, 2015.
- [16] J. Lee, T. Ma, and Q. Lin, "Distributed Stochastic Variance Reduced Gradient Methods," *arXiv:1507.07595 [cs, math, stat]*, jul 2015. [Online]. Available: <http://arxiv.org/abs/1507.07595>
- [17] A. Ng, "Feature selection, l1 vs. l2 regularization, and rotational invariance," in *Intl Conference on Machine Learning*, 2004.
- [18] R. Johnson and T. Zhang, "Accelerating Stochastic Gradient Descent using Predictive Variance Reduction," *Nips*, vol. 1, no. 3, pp. 315–323, 2013. [Online]. Available: [http://papers.nips.cc/paper/4937-accelerating-stochastic-gradient-descent-using-predictive-variance-reduction.pdf%5Cnhttp://papers.nips.cc/paper/4937-accelerating-stochastic-gradient-descent-using-predictive-variance-reduction%5CnAll Papers/J/Johnson and Zhan](http://papers.nips.cc/paper/4937-accelerating-stochastic-gradient-descent-using-predictive-variance-reduction.pdf%5Cnhttp://papers.nips.cc/paper/4937-accelerating-stochastic-gradient-descent-using-predictive-variance-reduction%5CnAll%20Papers/J/Johnson%20and%20Zhan)
- [19] L. Bottou, "Stochastic learning," in *Advanced Lectures on Machine Learning*, ser. Lecture Notes in Artificial Intelligence, LNAI 3176, O. Bousquet and U. von Luxburg, Eds. Berlin: Springer Verlag, 2004, pp. 146–168. [Online]. Available: <http://leon.bottou.org/papers/bottou-mlss-2004>
- [20] K. Lee, J. Lee, Y. Yi, J. Rhee, and S. Chong, "Mobile Data Offloading: How Much Can WiFi Deliver?" *IEEE/ACM Transactions on Networking*, vol. 21, no. 2, pp. 536–550, April 2013.