Double Confidential Federated Machine Learning Logistic Regression for Industrial Data Platforms

A. Navia-Vázquez [1] **M. Vázquez-López** [1] **J. Cid-Sueiro** [1]

Abstract

Federated Machine Learning schemes have become very popular to train models without exposing the training data, but they reveal the trained model to the participants. We propose a "double confidential" scheme inspired by the risk-utility trade-off in which the final trained model is only available to the *aggregator*: some partial information is leaked with no performance penalty, the computational overhead is moderate, and independent -non-colluding- servers are not needed. These characteristics are necessary to deploy an Industrial Data Platform, as described in the Musketeer EU-H2020 project, where data from different industrial partners is to be combined to obtain an improved model. We evaluate the performance of the proposed scheme in a binary classification task solved with a Logistic Regression model on a variety of datasets and show that good performance is achieved with an affordable increase in complexity with respect to the standard federated approach.

1. Introduction

It is well known -and recently widely recognized- that the amount and diversity of training data is a key factor to obtain better Machine Learning (ML) models, as exposed at length in (Halevy et al., 2009). Therefore, being able to accumulate a large amount of relevant data to a given problem is, at least, as important as correctly selecting and tuning/training a particular ML model. However, data harvesting faces many barriers such as privacy regulations on Personal Data Information (PDI) (e.g. General Data Protection Regulation (GDPR)(Council of European Union, 2014)) and business interests or other legal/confidentiality restrictions. Techniques for dealing with this situation are generally referred as Privacy Preserving (PP) Technologies (PPTs), irrespective of whether privacy or confidentiality is to be preserved (Timan, 2019). We will be focusing on their application to Machine Learning (PPML), and we will restrict ourselves to the context of Industrial Data Platforms (IDP). This scenario represents a recent proposal aiming at facilitating the emergence of a true data economy: different parties want to share some information to obtain better ML models for a given economical purpose (IDC, 2016). Under the IDP approach, data is considered as a commodity and the economic value is obtained through the trained models and their applications. Our work is contextualized in the Musketeer UE-H2020 project (Musketeer, 2018), where data from several industrial partners are to be combined using different privacy preserving technologies to ultimately produce improved Machine Learning Models.

One PPML approach that has recently gained a lot of attention in the ML community is that of Federated Machine Learning (FML), after the seminal paper by Google (Konečný et al., 2016), due to its potential applicability to a distributed scenario with millions of data providers and almost an unlimited amount of training data (Yang et al., 2019). FML relies on gathering in an "aggregator" -or "master"- the gradient updates from every "data provider" -or "worker"-, to update the current model, which is broadcasted to the workers[1], and the procedure repeated until convergence. Therefore the model is public -shared with all the participants-. In our proposal we want to obtain a model only known to the aggregator, hence the "double confidential" (DC) term, i.e., confidentiality refers both to the training data and the resulting model.

Secure Multiparty Computing (SMC) could be a candidate PPT, but it requires to "distribute" the data -secret sharing concept- among two or more non-colluding servers running

[1]Department of Signal Theory and Communications, Univ. Carlos III, Madrid, Spain. Correspondence to: A. Navia-Vázquez <angel.navia@uc3m.es>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020.

[1]We have intentionally avoided the client/server naming because it usually refers to a communication architecture/model where a server provides resources and services to one or more clients (e.g. web servers). The master and workers described in the paper are just processes running in different machines that carry out some part of the algorithmic computations and that can communicate with each other under different communication models: client/server, peer-to-peer, etc.

in different organizational boundaries and never cooperating beyond the specified protocols, which is not always easy to guarantee. Another limitation is that all nonlinear functions must be approximated with polynomials, which ultimately may lead to algorithmic instabilities. Algorithms exist for matrix operations among two parties (Du & Atallah, 2001) but they usually rely on the Oblivious Transfer protocol, which soon becomes impractical due to its large communication overheads (Goldreich, 1999). Another candidate PPT is that of Homomorphic Encryption (HE), but the associated computational cost is usually unaffordable, especially when large encryption keys are used [(Timan, 2019)].

The UK's Information Commissioner's Office has recently released a document discussing the security risks in AI scenarios (UK's Information Commissioner's Office, 2020) under the UK's Data Protection Act (2018) and the General Data Protection Regulation (GDPR) perspective. In this guide, they recognize it is unrealistic to adopt a 'zero tolerance' approach to risks and propose instead to manage them. There exists a clear trade-off between confidentiality and statistical accuracy/computational cost, and we can assume that it is not possible to simultaneously achieve both high statistical accuracy, computational efficiency and confidentiality in the cryptographic sense, as pointed out in the above mentioned report. This argumentation also corresponds to the well known risk-utility trade-off (Duncan et al., 2001). We propose a method that tries to maximize performance, while minimizing the computational cost, communication requirements, and the amount of leaked information about the training dataset and the resulting model. We need to relax the strong security assumptions associated with SMC or HE approaches but always guaranteeing that the individual training samples are not revealed and the resulting model is only known to the master. The participating data providers must agree beforehand on the type of intermediate information that is being revealed: we will accept that some intermediate values or statistics are exposed, but guaranteeing that it is not possible to reverse-engineer them to obtain any individual training record or the model itself. An analogous security relaxation has also been adopted in [(Slavkovic et al., 2007)], where some statistical disclosure is allowed to achieve a full statistical analysis of a distributed dataset.

Summarizing, we propose a double confidential (DC) algorithm for Logistic Regression to be used in an IDP context, under the following assumptions:

- The training data does not leave the data provider organizational limits.
- The model is confidential for the workers, only known to the master.
- Some intermediate information is leaked, but individual training samples or the model itself cannot be inferred from that. The participants will be aware of this partial leakage and must agree beforehand.
- We do not rely on extra independent servers and hence we are free from the server collusion problem.

2. Experimental scenario

Benchmarking PPML techniques in fair conditions is not an easy task. On one hand, not every published result comes along with a companion software implementation. On the other hand, even when one is provided, it is hardly possible to compare implementations optimized to different degrees -from simple ones in *slow* interpreted languages to highly-optimized compiled software. Moreover, in some cases processing and communication times are not clearly separated, and/or only the computational cost per epoch is reported, while ignoring the number of iterations required for convergence. In this work we are interested in benchmarking the algorithms and protocols themselves -not any particular efficient software implementation- under a common framework. We run the experiments using the same contour conditions: same datasets and processors, same number of cores used per participant, a unified stopping criterion and the same performance measurements. We focus on a binary classification task solved using Logistic Regression, which is a very common and well performing approach to many classification problems. We will use a centralized version of the method to obtain baseline reference values (gold standard).

In the distributed scenario the learning process is shared among different participants: data providers or workers -each with a fraction of the training dataset- and an aggregator controlling the whole process. In Figure 1 we show an scenario with one master and M workers, such that worker m contributes with a portion of training data $\{\mathbf{X}^m, \mathbf{y}^m\}$. The master exchanges messages with the workers until the final trained model is obtained. We assume that these types of communications are available: direct "send" (master→worker), broadcast (master→*all* workers) and "roundrobin" (master→worker$_i$→worker$_j$→master).

We assume that input patterns have been conveniently (offline) pre-processed and transformed into numerical features and that the available data has been partitioned into three different datasets: train, validation and test[2] . The training patterns are used to adjust the parameters of the model, the validation set for hyperparameter selection and the test set to obtain the final performance measurements. All these datasets comprise pairs in the form $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i$ is a

[2]If the original dataset has already a test partition, we respect that and only split the rest of the data into training and validation using a $80\% - 20\%$ scheme.

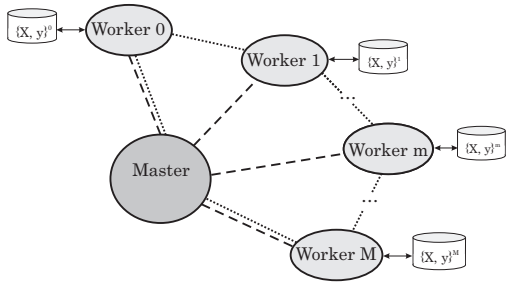


Figure 1. Distributed setup with a master and several workers ("data providers"). The dashed and dotted lines represent message passing under different communication protocols: "send" and "broadcast" -dashed- and "roundrobin" -dotted-.

collection of numerical input features[3] and y_i is the corresponding binary target value. Usually, these values are stored in matrix form, to facilitate the algorithmic formulation, such that by storing $\mathbf{x}_i^\intercal$ by rows we obtain the input matrix $\mathbf{X}$ (also known as "design matrix"), and by storing y_i values by rows, we obtain the target vector $\mathbf{y}$.

3. Algorithmic description: Double Confidential Logistic Regression

Consider sample set $\mathcal{S} = \{(\mathbf{x}_i, y_i), i = 1, ..., N\}$ of statistically independent samples. In the Logistic Regression (LR) model, binary labels y_i are assumed to be drawn from conditional Bernoulli distributions with parameters $p_i = \mathbb{E}\{y_i|\mathbf{x}_i\}$ that are given by

$$p_i = P(y_i = 1|\mathbf{x}_i; \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{x}_i^\intercal \mathbf{w})} = \sigma(\mathbf{x}_i^\intercal \mathbf{w}) \quad (1)$$

where $\sigma(.)$ is commonly known as the logistic function. Given the sample set, $\mathcal{S}$, model parameter $\mathbf{w}$ can be estimated by minimizing the empirical risk based on the cross entropy, which is a convex function of the weights.

$$\begin{aligned} L(\mathbf{w}) &= -\sum_{i=1}^{N} \left(y_i \log(p_i) + (1 - y_i) \log(1 - p_i)\right) \\ &= \sum_{i=1}^{N} \left(\log(1 + \exp(\mathbf{x}_i^\intercal \mathbf{w})) - y_i \mathbf{x}_i \mathbf{w}\right) \end{aligned} \quad (2)$$

3.1. Model optimization via Gradient Descent

Maybe the most straightforward model optimization approach is that based on the Gradient Descent rule:

$$\hat{\mathbf{w}}^{(t+1)} = \hat{\mathbf{w}}^{(t)} - \mu \nabla_{\hat{\mathbf{w}}}^{(t)} \quad (3)$$

[3]It is common that the first input feature is a constant value (e.g. 1), to account for a bias term in the model or "intercept".

Algorithm 1 FML general procedure

Initialization: $\mathbf{w}^{(0)} = \mathbf{0}$
repeat
  The Master broadcasts $\hat{\mathbf{w}}^{(t)}$ to the workers.
  The Master averages the gradients from the workers to build the global gradient $\nabla_{\hat{\mathbf{w}}}^{(t)} = \sum_{j=1}^{M} \nabla_{\hat{\mathbf{w}}}^{(t)}[j]$
  The master updates the model: $\hat{\mathbf{w}}^{(t+1)} = \hat{\mathbf{w}}^{(t)} - \mu \nabla_{\hat{\mathbf{w}}}^{(t)}$
until $||\hat{\mathbf{w}}^{(t+1)} - \hat{\mathbf{w}}^{(t)}|| < \eta$

where $\hat{\mathbf{w}}^{(t)}$ is the weight vector estimation at time t, $\nabla_{\hat{\mathbf{w}}}^{(t)}$ is the gradient of the empirical risk at that time, and μ is the learning rate. In the LR case, the gradient is simply:

$$\nabla_{\hat{\mathbf{w}}}^{(t)} = \sum_{i=1}^{N} (\sigma(\mathbf{x}_i^\intercal \hat{\mathbf{w}}^{(t)}) - y_i)\mathbf{x}_i \quad (4)$$

The combination of Eqs. (3) and (4) yields a very simple and highly efficient training procedure with guaranteed convergence -in the LR case- because $L(\mathbf{w})$ is convex.

3.2. Confidential Gradient Descent (C-GD)

The application of the GD procedure in a distributed scenario is described in Alg. 1. It is formally equivalent to the Federated Machine Learning (FML) procedure described in (Yang et al., 2019), where the model is revealed to the workers at every step but the training data is kept confidential[4]. In the next section we will extend this approach to the case in which the model is also hidden from the workers.

3.3. Double Confidential Distributed Logistic Regression

To deploy double confidential schemes where the model parameters are not shared with the workers, we need to securely implement some operations such as the dot product $\mathbf{x}_i^\intercal \mathbf{w}$ in (1), but to comply with the requirements described at the end of the Introduction, we cannot implement the SMC scheme in (Dahl et al., 2018; Ryffel et al., 2018) or use HE, as discussed in Section 1. We propose here a scheme that preserves the model and training data confidentiality at the cost of revealing some intermediate operation results.

To compute the gradient in Eq. (4), every worker needs to obtain the current model outputs $o_i^{(t)} = \sigma(\mathbf{x}_i^\intercal \hat{\mathbf{w}}^{(t)})$, but a direct access to these values would reveal the model weights $\hat{\mathbf{w}}^{(t)}$, since the sigmoid function can be inverted and, from a collection of P equations $\{s_i = \mathbf{x}_i^\intercal \hat{\mathbf{w}}^{(t)}\}$, the weights

[4]For improved confidentiality, the gradients can be averaged with a Secure Sum Protocol relying on the roundrobin mechanism, as described in (Mehnaz et al., 2017).

could be obtained[5]. In general terms, it is safer to compute the state values s_i at the Master, since the equations $\{\mathbf{x}_i^\intercal \hat{\mathbf{w}}^{(t)} = s_i\}$ cannot be inverted to obtain the input values $\mathbf{x}_i$, even if $\hat{\mathbf{w}}^{(t)}$ and s_i are known. To prevent a potential disclosure after several steps of the algorithm, every worker has to randomize the ordering of its data at every step, which does not affect the gradient computation. We propose to use the Secure Dot Product (SDP) computation described in (Zhu & Takagi, 2015) to securely obtain $\mathbf{s}[j] = \mathbf{X}[j]\hat{\mathbf{w}}^{(t)}$ at the master, without exposing the model. Note that any other SDP protocol could alternatively be used, we are simply using the algorithm in (Zhu & Takagi, 2015) as a tool. We will denote this operation as $< \cdot, \cdot >_{\text{SDP}}$. Briefly, the master generates some transformed data and random masking values $[\tilde{\mathbf{w}}, \mathbf{M}_w]$ and shares them with the worker which can also compute its transformed data and masking $[\tilde{\mathbf{X}}, \mathbf{M}_X] = f_{\text{SDP}}^{(W)}(\mathbf{X}, \tilde{\mathbf{w}}, \mathbf{M}_w)$. Finally, the master can compute the result as $\mathbf{s}[j] = f_{\text{SDP}}^{(M)}(\mathbf{w}, \mathbf{M}_w, \tilde{\mathbf{X}}, \mathbf{M}_X)$. For more details about this protocol, please see (Zhu & Takagi, 2015). The master can compute the outputs for worker j as

$$\mathbf{o}^{(t)}[j] = \sigma(\mathbf{s}^{(t)}[j]) \tag{5}$$

which are needed by worker j to compute the gradients. As already mentioned, these values cannot be directly sent to the worker, so we proceed with a second SDP protocol, such that as a result the master obtains the average gradient from every worker:

$$\nabla_{\hat{\mathbf{w}}}^{(t)} = \mathbf{X}[j]^T(\mathbf{o}[j] - \mathbf{y}[j]) =< \mathbf{X}[j], \mathbf{o}[j] >_{\text{SDP}} - \mathbf{r}[j] \tag{6}$$

where $\mathbf{r}[j] = \mathbf{X}[j]^T\mathbf{y}[j]$ is the data cross-correlation vector, which does not depend on the weights and hence it only needs to be computed and transmitted once. Although $\mathbf{r}[j]$ carries statistical information about the data, it does not reveal any particular training point. Therefore, the global gradient $\nabla_{\hat{\mathbf{w}}}^{(t)}$ can be computed as the average of terms $\nabla_{\hat{\mathbf{w}}}^{(t)}[j]$ received from all workers, such that Eq. (3) can now be used to update the model weights. As a part of the learning process, the master receives some partial information from every Worker, namely $\mathbf{o}^{(t)}[j]$, $\nabla_{\hat{\mathbf{w}}}^{(t)}[j]$ and $\mathbf{r}[j]$. The DC-GD procedure is Summarized in Algorithms 2 and 3.

Algorithm 2 DC-GD algorithm (at Master)

Inputs: μ (learning rate), η (convergence threshold), $\mathbf{w}^{(0)} = \mathbf{0}$
for every worker$[j]$ **do**
  $\mathbf{r}[j] =$ worker$[j]$.get_r()
end for
repeat
  for every worker$[j]$ **do**
    Obtain states using SDP:
      $\{\tilde{\mathbf{X}}, \mathbf{M}_x\} =$ worker$[j]$.get_s$(\tilde{\mathbf{w}}^{(t)}, \mathbf{M}_w)$;
      $\mathbf{s}[j] = f_{\text{SDP}}^{(\text{M})}(\mathbf{w}^{(t)}, \mathbf{M}_w, \tilde{\mathbf{X}}, \mathbf{M}_x)$
    Compute outputs: $\mathbf{o}^{(t)}[j] = \sigma(\mathbf{s}[j])$
    Get gradients using SDP:
      $\{\tilde{\mathbf{X}}^*, \mathbf{M}_x^*\} =$ worker$[j]$.get_g$(\tilde{\mathbf{o}}, \mathbf{M}_o)$;
      $\nabla_{\hat{\mathbf{w}}}^{(t)}[j] = f_{\text{SDP}}^{(\text{M})}(\mathbf{o}, \mathbf{M}_o, \tilde{\mathbf{X}}^*, \mathbf{M}_x^*) - \mathbf{r}[j]$
  end for
  Global gradient: $\nabla_{\hat{\mathbf{w}}}^{(t)} = \frac{1}{M}\sum_j \nabla_{\hat{\mathbf{w}}}^{(t)}[j]$
  Weight update: $\hat{\mathbf{w}}^{(t+1)} = \hat{\mathbf{w}}^{(t)} - \mu\nabla_{\hat{\mathbf{w}}}^{(t)}$
until $\|\mathbf{w}^{(t+1)} - \mathbf{w}^{(t)}\|/\|\mathbf{w}^{(t)}\| < \eta$

Algorithm 3 DC-GD algorithm (at Worker "j")

Inputs: training data $\mathbf{X}, \mathbf{y}$
get_r():
Return $\mathbf{r}[j] = \mathbf{X}^\intercal[j]\mathbf{y}[j]$.
get_s($\tilde{\mathbf{w}}^{(t)}, \mathbf{M}_w$):
Return $\{\tilde{\mathbf{X}}[j], \mathbf{M}_x[j]\} = f_{\text{SDP}}^{(\text{W})}(\mathbf{X}[j], \tilde{\mathbf{w}}^{(t)}, \mathbf{M}_w)$
get_g($\tilde{\mathbf{o}}, \mathbf{M}_o$):
Return $\{\tilde{\mathbf{X}}^*[j], \mathbf{M}_x^*[j]\} = f_{\text{SDP}}^{(\text{W})}(\mathbf{X}[j], \tilde{\mathbf{o}}, \mathbf{M}_o)$

4. Experimental results

In this section we benchmark the proposed scheme and use the centralized results as a reference. We have selected a collection of datasets for binary classification, as indicated in Table 1, with a wide range of characteristics: from a few thousands training patterns to millions, from a few input features to thousands. We also show in that table the total size of every dataset in Mbytes. The variety in training conditions will allow us to obtain conclusions for a wide spectrum of application scenarios. The selected datasets are Census Income Data ("Income"), and Super-Symmetric particles ("Susy"), described in (Dua & Graff, 2019); binarized Forest Covertype, as in (Collobert et al., 2002); Web page ("w8a") database (Platt, 1998); MNIST database of handwritten digits[6] ("mnist") (LeCun et al., 1998) and the 20 Newsgroups data set ("news20"), transformed into four datasets ("news20-A", "news20-B", "news20-C", "news20-D") with variable number of input features (500, 1.000, 2.000 and 3.000), respectively (Lang, 1995).

All the experiments in this paper have been executed in the same machine: a double processor Intel Xeon E5-2640 v4 running at 2.40GHz with 10 cores/20 threads, 128 Gb of memory, 2x1Gbit network interface and 2x1Tbyte hard disk, running a Gentoo distribution optimized for this architecture. The same data partition has been used in all the experiments, and the same stopping criteria is applied for all algorithms

[5]Provided P is greater than the number of features and those equations are linearly independent.

[6]Transformed to the binary task of distinguish between even and odd numbers.

Table 1. Datasets used for benchmarking.

DATASET	NTR	NTST	NF	MB
INCOME	26.049	16.281	107	40,2
W8A	44.774	14.951	300	148,6
NEWS-A	15.996	2.001	500	76,4
MNIST	50.000	10.000	784	419,2
NEWS-B	15.996	2.001	1.000	152,7
NEWS-C	15.996	2.001	2.000	305,3
NEWS-D	15.996	2.001	3.000	457,8
COVTYPE	464.809	58.102	54	243,8
SUSY	3.600.000	500.000	18	724,8

Table 2. Performance of the algorithms in the centralized scenario: AUC and training time is shown for C-GD and DC-GD.

DATASET	AUC		TIME(S)	
	C-GD	DC-GD	C-GD	DC-GD
INCOME	0,89	0,89	0,7	6,3
W8A	0,92	0,92	5,6	55,9
NEWS20-A	0,84	0,84	5,7	56,3
MNIST	0,94	0,94	14,6	152,9
NEWS20-1000	0,86	0,86	11,1	126,0
NEWS20-2000	0,89	0,89	21,9	249,2
NEWS20-3000	0,88	0,88	33,4	373,8
COVTYPE	0,82	0,82	9,5	90,2
SUSY	0,85	0,85	35,5	360,9

($\eta = 0.005$). Any hyperparameters -such as the learning rate- have been estimated offline using cross-validation.

4.1. Centralized scenario

As a preliminary evaluation, we benchmark the C-GD (standard FML) and DC-GD (secret model) algorithms in a centralized scenario: a single process running on a single thread and with direct access to all data. As a performance measure we have used the Area Under Curve (AUC) of the Receiver Operating Characteristic (ROC) curve, since it reflects the overall quality of the classifier and it does not depend on the threshold selected to produce the final decisions. AUC values and the total training time are shown in Table 2. The results on the centralized scenario will serve as a reference (gold standard). We observe that C-GD and DC-GD achieve the same AUC values (no performance loss due to the double confidentiality property). The training time of DC-GD increases by roughly a factor of ten with respect to C-GD. This is the price we have to pay to protect the model from the workers. Note that no communication costs are incurred in this case, the increase is mainly due to the Secure Dot Product algorithm. This overhead could be reduced by using a more efficient SDP protocol, if such option exists.

4.2. Distributed scenario

We will benchmark here the algorithms under the distributed setup: the master and every worker will be separated processes, each one running on a single thread. In this scenario, the training dataset is distributed among the participating workers, such that in every experiment the same amount of training data is used. The objective here is to show that the distributed and privacy preserving setup does not reduce the performance with respect to the centralized approach, i.e., the performance figures are expected to be comparable. We will considerate scenarios with 1, 5, 10 and 15 workers, we cannot go beyond that because the available machine has 20 threads and we need some extra threads for communications and the master process. The case with only one worker represents a situation where all the training data is at the same place (the worker), but the model update takes place in another process (the master). It is a simplified case of the situation depicted in Figure 1, that can serve as a bridge between the centralized and distributed cases. As a first benchmark we compare the AUC performance of the algorithms. We have depicted in Figure 2 the results for C-GD in (a) and DC-GD in (b)[7]. It can be seen that the AUC values are very close to those obtained by the centralized algorithm, i.e., no significant performance loss is observed.

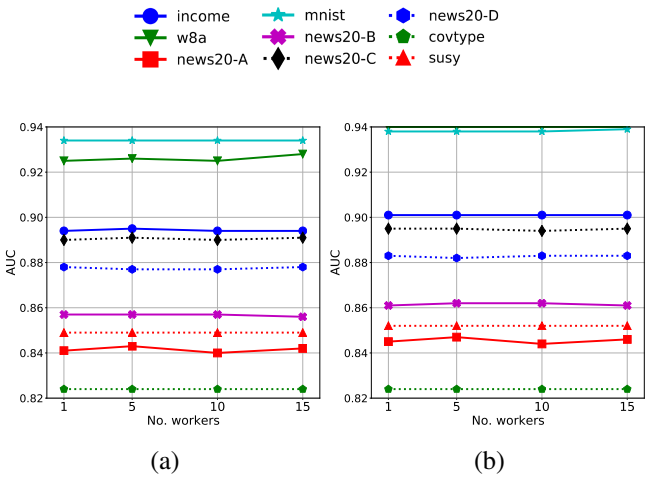
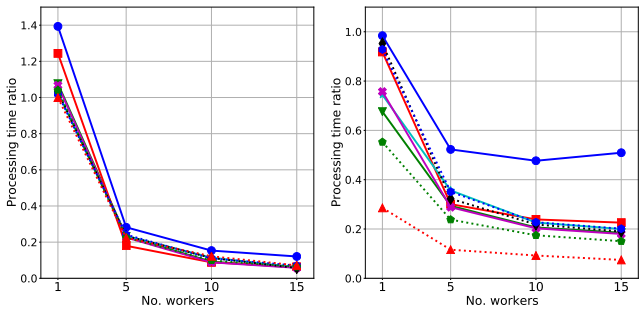


Figure 2. AUC in the distributed case, as a function of the number of workers. We show C-GD in (a), DC-GD in (b). The legend is the same for all Figures (only shown here).

We will differentiate between processing time and total computation time (that also includes communication and waiting/management times). In Figure 3 we have depicted the ratio of processing time in the distributed vs. the centralized case. As expected, we observe a decreasing tendency with the number of workers. This is obviously due to the

[7]The legend is only shown in this Figure, it's the same for the other figures.

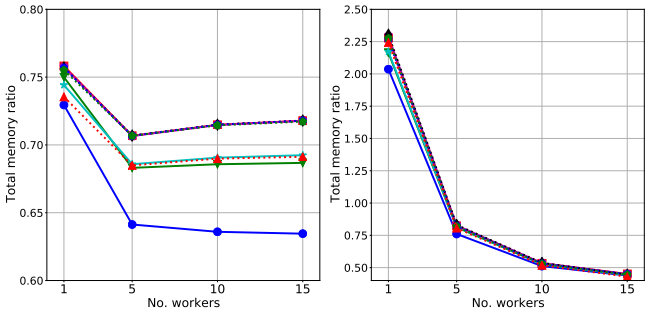
fact that the computations are shared among an increasing number of cores (one core per worker). Therefore the distributed versions of the algorithms are efficient in the sense that they speed up the computation when more workers are available. Furthermore, the ratio of processing time w.r.t. the centralized case of DC-GD is of the same order as that of C-GD.



(a) (b)

Figure 3. Ratio among processing time in the distributed case with respect to the centralized one, as a function of the number of workers. C-GD in (a), DC-GD in (b).

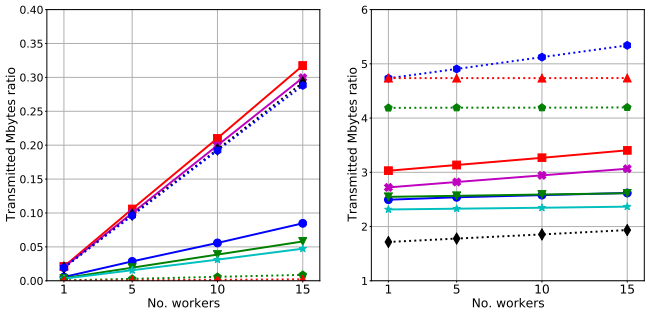
With respect to the maximum memory used by the participating processes, we have depicted in Figure 4 the ratio among maximum memory per process in the distributed scenario with respect to the maximum memory in the centralized case, and it is on the same order of magnitude in all cases.



(a) (b)

Figure 4. Ratio among the maximum memory needed per process in the distributed scenarios w.r.t. the maximum memory used in the centralized case, as a function of the number of workers (1, 5, 10, 15). C-GD in (a), DC-GD in (b).

The total amount of information (in Mbs) transmitted among the processes during every training episode is shown in Figure 5. It is normalized with respect to the size of every dataset to ease comparisons, such that 1 unit amounts to the size of the whole dataset. We observe that C-GD (FML) is very efficient, requiring -on average- the transmission of only the 8% of the dataset size, while DC-GD requires on average to transmit 3.2 times the size of the dataset.



(a) (b)

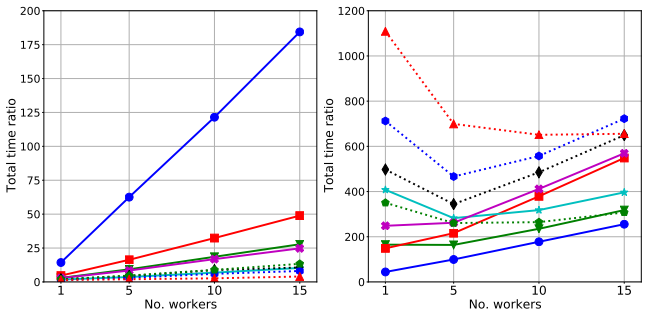
Figure 5. Ratio among the amount of information transmitted during training and the size of the database, as a function of the number of workers (1, 5, 10, 15). C-GD in (a), DC-GD in (b).

Finally, the total training time, that also takes into account the communication and distributed management overheads is larger for DC-GD, and it increases with the number of workers (more management/communications requirements). We have normalized all the training times with respect to the time required by the algorithms in the centralized case, and we show the results in Figure 6. For 15 workers, the averaged ratio for C-GD is about 37, whereas for DC-GD is 490, 13 times more. This extra communication time is due to the extra amount of information that DC-GD needs to transmit. It also depends on the communications efficiency[8]. The particular communications library used here is able to send data at a rate of -on average- 2.7 Mbytes per second. Speeding up communications by a given factor will reduce the transmission time by the same factor, thereby the potential reduction in the total training time could be significant if high speed communications are available.

5. Conclusions

We have proposed a Double Confidential FML scheme (DC-GD) to be used in a context of Industrial Data Platforms that preserves the confidentiality of the trained model. To satisfy the specific requirements of our application (no extra non-colluding nodes, no polynomial approximations to preserve high accuracy, affordable computational and communication resources) we have adopted an approach that balances the risk-utility trade-off: we allow some partial information

[8]The casuistry is so great -different technologies, different transmission bandwidths, different latency times, etc.- that it exceeds the scope of this paper to evaluate all the possibilities.



(a) (b)

Figure 6. Ratio among total training time in the distributed case with respect to the centralized case, as a function of the number of workers (1, 5, 10, 15). C-GD in (a), DC-GD in (b).

leakage but always preserving the confidentiality of the training data and the resulting model. We have benchmarked the proposed scheme with a variety of datasets, under common experimental conditions for a fair comparison of results. We observe that DC-GD does not loose accuracy with respect to the standard FML approach, the processing time in the centralized setup is ten times higher, mainly due to the SDP computation overhead. In the distributed setup, the computation time ratio w.r.t. the centralized case is on the same order of magnitude -no extra penalty for distributed computing- and decreases with the number of workers. The maximum amount of memory needed is comparable in both cases. On the negative side, DC-GD requires the transmission of 40 times more information than standard FML and hence the total training time is 13 times higher in our particular setup (throughput of 2.7 Mbytes per second), but further proportional reduction is possible with a faster network. DC-GC incurs in extra costs w.r.t standard FML approach, but facilitates any IDP deployment that requires a secret model as a result.

In future works we will extend the proposed scheme to cover more ML algorithms and further analyze/quantify the incurred information leakage.

Acknowledgements

This work was supported by the European Union's Horizon-2020 RIA Programme [Musketeer project, grant number 824988]; and the Spanish Ministry of Economy and Business [grant number TEC2017-83838-R].

References

Collobert, R., Bengio, S., Bengio, Y., Dietterich, T., Becker, S., and Ghahramani, Z. A parallel mixture of svms for very large scale problems. *Neural Computation*, 14:1105 – 1114, 2002.

Council of European Union. Council regulation (EU) no 269/2014, 2014. *http://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1416170084502uri=CELEX:32014R0269*.

Dahl, M., Mancuso, J., Dupis, Y., Decoste, B., Giraud, M., Livingstone, I., Patriquin, J., and Uhma, G. Private machine learning in tensorflow using secure computation. *ArXiv.org*, 2018.

Du, W. and Atallah, M. Privacy-preserving cooperative scientific computations. In *Proceedings. 14th IEEE Computer Security Foundations Workshop, 2001.*, pp. 273–282, 2001.

Dua, D. and Graff, C. UCI Machine Learning Repository. In *http://archive.ics.uci.edu/ml*, Irvine, CA: University of California, School of Information and Computer Science., 2019.

Duncan, G., Keller-McNulty, S., and Stokes, L. Disclosure risk vs. data utility: The ru confidentiality map. national institute of statistical sciences. Technical report, 2001.

Goldreich, O. Secure multi-party computation, 1999.

Halevy, A. Y., Norvig, P., and Pereira, F. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2): 8–12, 2009.

IDC. Industrial data platforms - key enablers of industry digitization, 2016.

Konečný, J., McMahan, H. B., Ramage, D., and Richtárik, P. Federated optimization: Distributed machine learning for on-device intelligence. ArXiv eprint 1610.02527, 2016.

Lang, K. NewsWeeder: Learning to Filter Netnews. In *Proceedings of the 12th International Conference on Machine Learning (ICML-95)*, pp. 331–339. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1995.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pp. 2278–2324, 1998.

Mehnaz, S., Bellala, G., and Bertino, E. A secure sum protocol and its application to privacy-preserving multi-party analytics. In *Proc. 22nd ACM on Symposium on Access Control Models and Technologies SACMAT '17*, pp. 219–230, 2017.

Musketeer. Machine learning to augment shared knowledge in federated privacy-preserving scenarios. european union's horizon 2020 research and innovation programme under grant agreement no 824988, 2018. *http://www.musketeer.eu*.

Platt, J. Fast training of support vector machines using sequential minimal optimization. In Schoelkopf, B., Burges, C., and Smola, A. (eds.), *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

Ryffel, T., Trask, A., Dahl, M., Wagner, B., Mancuso, J., Rueckert, D., and Passerat-Palmbach, J. A generic framework for privacy preserving deep learning. *ArXiv*, abs/1811.04017, 2018.

Slavkovic, A., Nardi, Y., and Tibbits, M. "secure" logistic regression of horizontally and vertically partitioned distributed databases. In *Proc. of the 17th IEEE International Conference on Data Mining Workshops*, pp. 723–728, 2007.

Timan, T., Z. M. e. Data protection in the era of artificial intelligence. trends, existing solutions and recommendations for privacy-preserving technologies. big data value association (bdva) position paper., 2019. *http://www.bdva.eu/sites/default/files/Data%20protection%20in%20the%20era%20of%20big%20data%20for%20artificial%20intelligence_BDVA_FINAL.pdf*.

UK's Information Commissioner's Office. Guidance on the ai auditing framework, draft guidance for consultation, 2020. *https://ico.org.uk/media/about-the-ico/consultations/2617219/guidance-on-the-ai-auditing-framework- draft-for-consultation.pdf*.

Yang, Q., Liu, Y., Chen, T., and Tong, Y. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10:1–19, 2019.

Zhu, Y. and Takagi, T. Efficient scalar product protocol and its privacy-preserving application. *International Journal of Electronic Security and Digital Forensics*, 7:1–19, 2015.