Distributed Differentially Private Stochastic Gradient Descent: An Empirical Study

István Hegedűs and Márk Jelasity
University of Szeged, MTA-SZTE Research Group on AI, Szeged, Hungary
Email: {ihegedus,jelasity}@inf.u-szeged.hu

***Abstract*—In fault-prone large-scale distributed environments stochastic gradient descent (SGD) is a popular approach to implement machine learning algorithms. Data privacy is a key concern in such environments, which is often addressed within the framework of differential privacy. The output quality of differentially private SGD implementations as a function of design choices has not yet been thoroughly evaluated. In this study, we examine this problem experimentally. We assume that every data record is stored by an independent node, which is a typical setup in networks of mobile devices or Internet of things (IoT) applications. In this model we identify a set of possible distributed differentially private SGD implementations. In these implementations all the sensitive computations are strictly local, and any public information is protected by differentially private mechanisms. This means that personal information can leak only if the corresponding node is directly compromised. We then perform a set of experiments to evaluate these implementations over several machine learning problems with both logistic regression and support vector machine (SVM) loss functions. Depending on the parameter setting and the choice of the algorithm, the performance of the noise-free algorithm can be closely approximated by differentially private variants.**

***Keywords*—distributed differential privacy, stochastic gradient descent, machine learning**

I. Introduction

In the era of smart appliances and the Internet of Things, *distributed* data processing and data mining are gradually becoming evermore important. One reason is that distributed computing allows better scalability compared to cloud-based solutions by exploiting local resources and networks, as proposed e.g. by Cisco in its ongoing fog computing initiative [1]. Another reason is the increasing need for privacy as the personal data collected and stored by ubiquitous personal computing devices such as smart meters, sensors, or mobile devices, is becoming richer and richer.

Here, we are concerned with the scenario in which each networked device stores only a small amount of data (typically collected locally), while there are many participating devices in the network. This model covers a wide range of applications including smart metering [2], collaborative mobile platforms [3] and Internet of Things platforms [4].

We focus on stochastic gradient descent (SGD) as our learning algorithm of choice. SGD visits all data records in a random order and updates a model approximation based on each record using the local gradient at that record. In general, SGD is a preferable method in large scale data mining [5] due to its scalability and simplicity. In our edge computing scenario the simplicity of SGD is a very important advantage: all the sensitive computations can be made strictly local to network nodes, as we will show later. In addition, no aggregation, synchronization, or central collection of data is necessary. While it is possible to achieve significant speedup using a number of techniques that require such aggregation and synchronization operators [6], in this paper we focus on "vanilla" SGD that—as we will show—serves as a worst case in terms of the level of accuracy that can be achieved using a given amount of information leakage. The practicality of SGD in the system model above (without privacy preservation) was demonstrated earlier [7].

The problem we tackle is to provide a practical evaluation of the quality (prediction accuracy) of SGD as a function of several important design choices we identify. We work within the framework of differential privacy [8], where information leakage can be explicitly bounded by adding appropriately engineered noise to the output. Without differentially private mechanisms, even if all the computations are performed securely, the privacy of data is still not guaranteed as the output of any secure computation might leak information about individual data records indirectly.

The differential privacy of model fitting via optimization, and in particular SGD was investigated earlier in a number of publications. Generic frameworks, such as PINQ [9] and GUPT [10], have been proposed that do not readily lend themselves to secure distributed implementations. Chaudhuri et al. propose a method based on adding noise to the end result only, or perturbing the objective function itself [11], [12]. This approach does not allow for an obvious secure distributed implementation either. The method that we build on was proposed by Song et al. [13]. There, instead of perturbing the objective function, each update during the iterative gradient descent procedure is made differentially private. This allows for fully local gradient updates where the resulting gradient and the updated model can be made public. This prevents any uncontrolled data leakage as long as the personal computing device is not compromised.

Our contribution is a thorough assessment of the practicality of differentially private SGD in fully distributed environments where each node has one data record only. In particular,

1) we propose several variants of possible differentially private SGD implementations and

2) we evaluate these algorithms over several databases and parameter settings using the loss functions of logistic regression and support vector machines (SVM).

II. Background

A. Stochastic Gradient Descent

The problem of classification is an important part of machine learning. Given a data set $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ of n observations, where an object or an example is represented by a pair of a feature vector $x \in R^d$ and the corresponding class label $y \in C$, where d is the dimension of the problem and C is the domain of class labels. In the case of binary classification the number of possible class labels is two (e.g. $C = \{0, 1\}$). The problem of classification is often expressed as finding the parameters w of a function $f_w : R^d \to C$ that can correctly classify as many examples in D as possible, as well as outside D (this latter property is called generalization). In other words, we are looking for a

$$w = \arg\min_w J(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_w(x_i), y_i) + \frac{\lambda}{2}\|w\|^2, \quad (1)$$

where the $\ell()$ is a loss function and $(\lambda/2)\|w\|^2$ is the regularization term with parameter λ. Function f_w is called the model of the data set. The regularization term helps the model to avoid overfitting the data set, thus helping generalization. The labeled data set is often split into two non-overlapping subsets: a training set for optimizing the parameters w of the model, and a test set for measuring the generalization performance of the optimized model.

Gradient descent (GD) is an iterative method that can find the optimum of a convex function. It is often used for optimizing the above objective function. The parameter vector w is iteratively updated using the derivative of the objective function that is computed on the whole training set

$$\begin{aligned} w_{t+1} &= w_t - \eta_t(\frac{\partial J}{\partial w}) \\ &= w_t - \eta_t(\lambda w + \frac{1}{n}\sum_{i=1}^{n} \nabla\ell(f_w(x_i), y_i)), \end{aligned} \quad (2)$$

where η_t is the learning rate at time t that scales the size of the gradient step.

Stochastic gradient descent (SGD) is similar, only it visits each example one-by-one instead of working with the entire database. It computes the gradient based on only one training sample in an iteration instead of the whole training set. For index i the update rule becomes

$$w_{t+1} = w_t - \eta_t(\lambda w + \nabla\ell(f_w(x_i), y_i)). \quad (3)$$

SGD is more preferable on very large training sets, or in distributed applications. It has two restrictions regarding the learning rate, namely we have to have $\sum_t \eta_t^2 < \infty$ and $\sum_t \eta_t = \infty$. These are necessary conditions for convergence [6].

The two suitable and widely used optimization algorithms we focus on are *Logistic Regression* [14] and the linear *Pegasos SVM* [15]. Both have an associated loss function that we can use along with SGD to train the corresponding model.

Algorithm 1 Gossip Learning Framework

```
 1: $(x, y) \leftarrow$ local training example
 2: currentModel $\leftarrow$ initModel()
 3: loop
 4:     wait($\Delta$)
 5:     $p \leftarrow$ selectPeer()
 6:     send currentModel to $p$
 7: end loop

 8: procedure onReceiveModel($m$)
 9:     $m$.updateModel($x, y$)
10:     currentModel $\leftarrow m$
11: end procedure
```

In the case of logistic regression the optimization problem is expressed as a maximization problem, since it is more natural to think of it as maximizing the logarithm of the likelihood

$$w = \arg\max_w \frac{1}{n}\sum_{i=1}^{n} \ln P(y_i|x_i, w) - \frac{\lambda}{2}\|w\|^2, \quad (4)$$

where $y_i \in \{0, 1\}$, $P(0|x_i, w) = (1 + exp(w^T x))^{-1}$ and $P(1|x_i, w) = 1 - P(0|x_i, w)$.

Pegasos SVM is a linear SVM solver method, which looks for the hyperplane that maximizes the margin between the instances of different classes

$$w = \arg\min_w \frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - y_i w^T x_i) + \frac{\lambda}{2}\|w\|^2, \quad (5)$$

where we now have $y_i \in \{-1, 1\}$.

Although we have only discussed binary classification, we will experiment with the more general multi-class algorithms, where we have instances from K different classes ($C = \{0, 1, \ldots, K-1\}$). A popular approach is to learn K distinct binary classifiers [16], one for each class. In particular, for the SVM approach we optimize K hyperplanes at the same time [17]. Similarly, when using logistic regression the objective function can be readily generalized to multiple classes [16].

B. Distributed Machine Learning

In our system model we are given a network of a large number of computational units (e.g. PCs, smart phones, tablets, wearable units, or smart meters). The members of this network can communicate with each other by message passing. A node in this network can send a message to another node whose address is known locally. We assume that every node in this network has only one training example (but we can benefit from having more local data). The set of these isolated examples form our machine learning database. We would like to learn a model over these instances in a fully distributed manner while also preserving privacy.

The Gossip Learning Framework [7] is a possible way to learn models in this fully distributed environment. The basic idea is that in the network many models perform random walks and are updated at every node using the local example. In more detail, every node executes Algorithm 1. A node in the network first initializes a local model, then iteratively sends its local model to a randomly selected node in the network. The address of the randomly selected node is provided by a peer sampling service (e.g. the NewsCast [18] protocol). When a node receives a model, it updates it by its locally stored training example using the SGD update rule, and then stores the updated model as its local model. Using this protocol the models stored by the nodes will converge to the same global optimum.

Our study is inspired by gossip learning in the sense that we focus on SGD algorithms that are implemented through a random walk of the evolving model over the network. We will assume that this random walk itself is secure. Ideas for achieving secure random walks were outlined, for example, in [19]. Here, we focus on privacy. In order to achieve privacy, we will apply a differentially private variant of the local update step, as explained below.

C. Differentially Private SGD

Differential privacy [8] is concerned with the leakage of personal information due to publishing the results of a given query over a database. Even if performed securely, the result of a query can leak information about individual records, for example, the maximum of a set of values is an individual record in itself. Differential privacy is achieved if noise is added to the query result in such a way that the following definition is satisfied.

Definition 1 (Differential Privacy): A randomized query $F : \mathcal{D} \mapsto \mathbb{R}^d$ is *ϵ-differentially private* iff

$$\forall x : e^{-\epsilon} \leq \frac{P(F(D) = x)}{P(F(D') = x)} \leq e^{\epsilon} \tag{6}$$

for all pairs of databases D and D' that differ in at most one record, where $\mathcal{D}$ is the set of possible databases.

That is, if we change one element in the database, the same output should be expected with a probability close to that over the original database. This way, one record never "matters too much" thereby limiting the information leakage as a result of the query.

A randomized query typically means adding noise to an otherwise deterministic query. This added noise is designed specifically for a given query and parameter ϵ such that the definition of $\epsilon-$differential privacy is satisfied. In more detail, to generate the additive noise we need to pick a noise distribution and the right distribution parameters. A common approach to take is to first determine the so-called *sensitivity* of the query [8], [20]:

Definition 2 (Global Sensitivity): The *global L^1-sensitivity* Z_F of F is given by

$$Z_F = \max_{D,D' \text{ differ in one record}} \|F(D) - F(D')\|_1, \tag{7}$$

where $\|\cdot\|_1$ is the L^1 norm.

The definition can be generalized by replacing the L^1 norm with a different norm. The usual norms to apply are the L^1 norm and the L^2 norm. In the case of applying the L^1 norm, the following noise distribution can be used: we need to add to all the dimensions of the output independent noise drawn from $\mathrm{Laplace}(0, Z/\epsilon)$ (where Z is the global sensitivity of the query), which will result in $\epsilon-$differential privacy. For the L^2 norm, the noise vector should have a uniform random direction and a length drawn at random from $\mathrm{Laplace}(0, Z/\epsilon)$. Based on the theoretical results described in [20], noise can be generated for any other norms.

Now, for one SGD update (as defined in Equation (3)) the private query we need to compute is the gradient $\nabla\ell(f_w(x_i), y_i)$. If we can guarantee that this gradient is bounded, the sensitivity is given directly. We can then add the appropriate noise N_t to the gradient and perform the differentially private local update

$$w_{t+1} = w_t - \eta_t(\lambda w_t + \nabla\ell(f_w(x_i), y_i) + N_t). \tag{8}$$

We are then free to publish w_{t+1} and send it to the next node.

For running SGD we need multiple queries, because we need the gradients based on many learning examples multiple times. Having seen how one can protect a single update, let us mention two useful concepts from differential privacy: sequential and parallel composition of queries [21].

In a sequential composition we are given a series of queries $F_i,\ i = 1, \ldots, k$. It can be proven that if all of these queries are ϵ-differentially private then the entire sequence of these k queries will be $k \cdot \epsilon$-differentially private. Note that the queries can depend on the results of the previous queries.

However, in the special case where the k queries are executed over pairwise disjoint subsets $D_i,\ i = 1, \ldots, k$, a case we call parallel composition, then the entire sequence of queries will remain ϵ-differentially private. Most importantly, in the case of SGD we have parallel composition, since updates are typically performed on a disjoint subset, in our case on a single record. Of course the same record can be visited many times, and these updates will compose sequentially.

In general, we can think of each example having a privacy budget of ϵ, which is spent when the given example is visited but which is not affected otherwise. This way, when each example has spent its privacy budget of ϵ, the entire SGD algorithm over the entire database spent only ϵ as well due to parallel composition.

III. Algorithm and Analysis

Based on the previous discussion, from now on we focus on SGD, assuming that there is a distributed implementation based on a random walk over the network. To implement such a random walk one needs to rely on several middleware services such as peer sampling, as discussed in Section II. Here we treat random walk as an abstract service and will study experimentally a number of variants. Our only assumption about the implementation of the service is that the random walk itself is secure and it can potentially jump to any node in the network despite any technical hurdles such as NAT boxes.

The network nodes hold one training example (x, y) and they calculate the local gradient for a given model w and time t locally, and they also add the appropriate noise term N_t to achieve differential privacy according to Equation (8). They are free to publish the resulting w_{t+1} and to send it to the next node. The parameter ϵ of differential privacy is a globally known constant.

A. Privacy Budget

Parameter ϵ is often called the privacy budget, because owning to the different compositional properties of series of queries one can, for example, decide to run one query with parameter ϵ or two sequentially composing queries with parameter $\epsilon/2$, or several parallel queries with parameter ϵ. All of these options result in an overall ϵ-differential privacy. Now, let us elaborate on the management of the privacy budget for SGD.

As mentioned in Section II, every training example (i.e., every node) in effect has its own ϵ budget for the updates. This budget can be used in a number of different ways. One can, for instance, set a finite number of k allowed updates and use ϵ/k for each one. This means multiplying the magnitude of the noise term by k for each update. In the experimental evaluation we study this parameter looking at the cases of $k = 1$ and $k = 5$. We can also follow a different approach and divide ϵ into an infinite number of parts by using $\epsilon/2^t$ for update t. This way noise increases exponentially, but we can execute as many updates as we wish using the same example. Note, however, that SGD will not converge in this case due to the exponentially increasing noise, so this approach is practical only for a small finite number of rounds.

The above implies a deeper result: it is not possible to run SGD until convergence with differential privacy because we either compute only a finite number of updates (and SGD needs an unlimited number of updates for theoretical convergence) or the signal-to-noise ratio will tend to zero in the update rule, which also prevents convergence. So the best we can achieve in theory is an approximation based on a relatively small number of updates per sample. For a large number of samples, however, this may be sufficient as our experiments will demonstrate.

Let us point out a major difference between our differentially private SGD implementation and gossip learning. In our SGD implementation there is only one random walk in the entire network, while in gossip learning there are many parallel walks. However, if there are many walks in parallel, they all "burn" the privacy budget so each walk will be assigned a smaller number of updates inversely proportional to the number of walks. It is therefore essential to run only one walk. However, the state of the walk is public, so it is possible to continuously broadcast the latest model w_t in the network if needed. The broadcast can be implemented in a distributed way (e.g. via gossip), or via publishing the latest update on a server. With public key cryptography the broadcast can be implemented securely as well.

As a last point connected to using the budget, let us consider the exact method of peer sampling used by our random walk. If we use uniform sampling with replacement then the walk will take useless steps when it visits a training example that has no more budget left. To be precise, the probability that a node is not visited at all during the first n updates in a network of size n is $\exp(-1)$ according to the Poisson distribution, which is a considerably large probability. Depending on the budget management option, this results in wasted bandwidth and time. This shows that the ideal random walk should use sampling without replacement, that is, it should follow a permutation of the network, and when all nodes have been visited, it should start a new permutation until the privacy budget has been spent.

B. Sensitivity Analysis

In our study, we focus on logistic regression and Pegasos SVM with gradients in Equations (9) and (10), respectively.

$$\frac{\partial J}{\partial w} = x(y - P(1|x,w)) = x(y - \frac{e^{w^T x}}{1+e^{w^T x}}) \quad y \in \{0,1\} \tag{9}$$

$$\frac{\partial J}{\partial w} = \delta(yw^T x < 1)yx, \quad y \in \{-1,1\} \tag{10}$$

In both cases the gradient is a linear function of x and its length is not larger than that of x. This immediately gives us the sensitivity of $2 \cdot \max_x \|x\|$. It is more convenient to normalize the training examples so that $\max_x \|x\| = 1$, in which case the sensitivity is 2. From now on, without loss of generality, we assume the examples are normalized this way. It is possible to achieve $\max_x \|x\| = 1$ through several different normalization methods. We will discuss these in the description of our experiments. Note that we need to protect only the gradient since the update rule in Equation 8 can be computed using public information as long as $(\nabla\ell(f_w(x_i), y_i) + N_t)$ is known.

C. Notes on Privacy and Security

It is very important to stress that any uncorrupted node is protected by this scheme regardless of fabricated input or the security of the random walk in general. In other words, even if the random walk is compromised and a given uncorrupted node gets arbitrary input and gets queried an arbitrary number of times, the node will be protected by ϵ-differential privacy.

Here, we focus on privacy only. Based on the comment above, this is indeed an independent problem as we can guarantee the privacy of uncompromised nodes regardless of the security of any other components of the implementation. Nevertheless, security is still necessary in a complete system as without it the global output can be corrupted and vandalized. In particular, the random walk needs to be secure to maintain an unbiased sampling of the learning examples. In addition, an adequate protection is needed against vandalism, when adversaries or faulty nodes inject arbitrary information into the system. However, again, the privacy of local data is completely in the hands of the local node, independently of the outside world.

IV. Experiments

Here, we present the experimental evaluation of our private gradient methods on realistic, real-life machine learning data sets. The goals of these experiments are twofold. First, we demonstrate the practical applicability of differential privacy in general under the system assumptions we presented earlier. This includes the speed of convergence and the quality of the end result. Second, we wish to explore several design options and offer practical advice on how to parameterize the differentially private mechanism we propose.

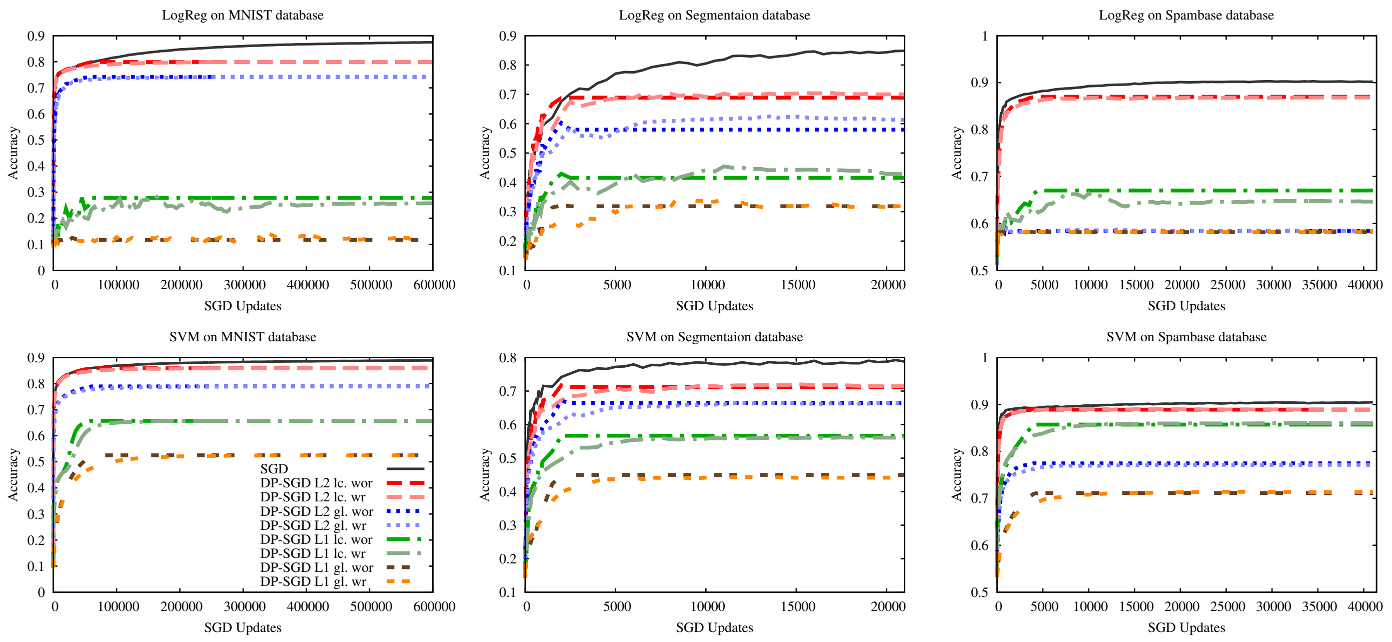

Fig. 1. Algorithm DP-SGD-1 with different vector norm (L^1 vs. L^2), normalization (local vs. global), and sampling (with or without replacement) with privacy budget $\epsilon = 1$. The x-axis spans a number of updates 10 times the size of the data set.

A. *Algorithm Variants*

Our algorithm variants are based on the algorithms described in Section III. We now give the parameter settings we used exactly, and we introduce the notations for these settings. In all cases we run SGD with the differentially private SGD update rule in Equation (8). The learning rate was set to $\eta_t = t^{-\frac{1}{2}}$ and $\lambda = 10^{-4}$. We will refer to this algorithm as DP-SGD.

The privacy budget we used was set to $\epsilon = 1$ unless otherwise stated. We also experimented with $\epsilon = 0.1$ to illustrate a stricter privacy requirement. As of using this budget, we study the three methods presented in Section III. The first is the method where, for all the examples, we use up the entire budget ϵ in a single update. We refer to this variant as DP-SGD-1. The remaining two variants are referred to as DP-SGD-5 and DP-SGD-∞, respectively. These correspond to the cases when we allow 5 updates using each example, all of which use up a budget of $\epsilon/5$, and when we allow any number of updates with update t using a budget of $\epsilon/2^t$.

The sampling of the next step of the random walk (that is, the next sample to update with) was selected independently at random in each step with or without replacement. In the case of sampling without replacement, when the samples run out, we start the sampling with the full set again.

As of selecting the vector norm in the definition of global sensitivity, we experiment with the L^2 norm and the L^1 norm. This defines the noise distribution, that is, the distribution of N_t in Equation (8), as described in Section II-C.

Finally, as a baseline, we also present the performance of SGD without the noise term N_t. We will refer to this variant simply as SGD.

TABLE I. THE MAIN PROPERTIES OF THE DATA SETS

	MNIST	Segmentation	Spambase
Training set size	60 000	2310	4140
Test set size	10 000	210	461
Number of features	784	19	57
Number of classes	10	7	2
Class-label distribution	uniform	uniform	6:4

Regarding specific algorithms needed to implement SGD, we include logistic regression and the Pegasos algorithm (see Equations (9) and (10)). Depending on the data set at hand, we used the multi-class variants of these algorithms. We will refer to these algorithms as LogReg and SVM.

B. *Data Sets and Preprocessing*

We selected our databases from different machine learning domains with different properties. Our first data set, called MNIST [22], contains gray level images of handwritten digits (from 0 to 9) of size 28×28. In the Image Segmentation data set the goal is to assign the pixels to one of the 7 hand-labeled segments, based on a set of high-level features. Finally, in the Spambase data set the task is to detect spam e-mails, again, based on a set of high level features such as the frequencies of suspicious words, etc. The last two data sets are part of the UCI machine learning repository [23]. The main properties of these data sets are summarized in Table I.

We performed the following preprocessing steps on the data sets. In all cases we first normalized the features one-by-one by linearly transforming them into the [0,1] interval, based on the maximum and minimum value of the given feature in the training data set. Next we performed one of

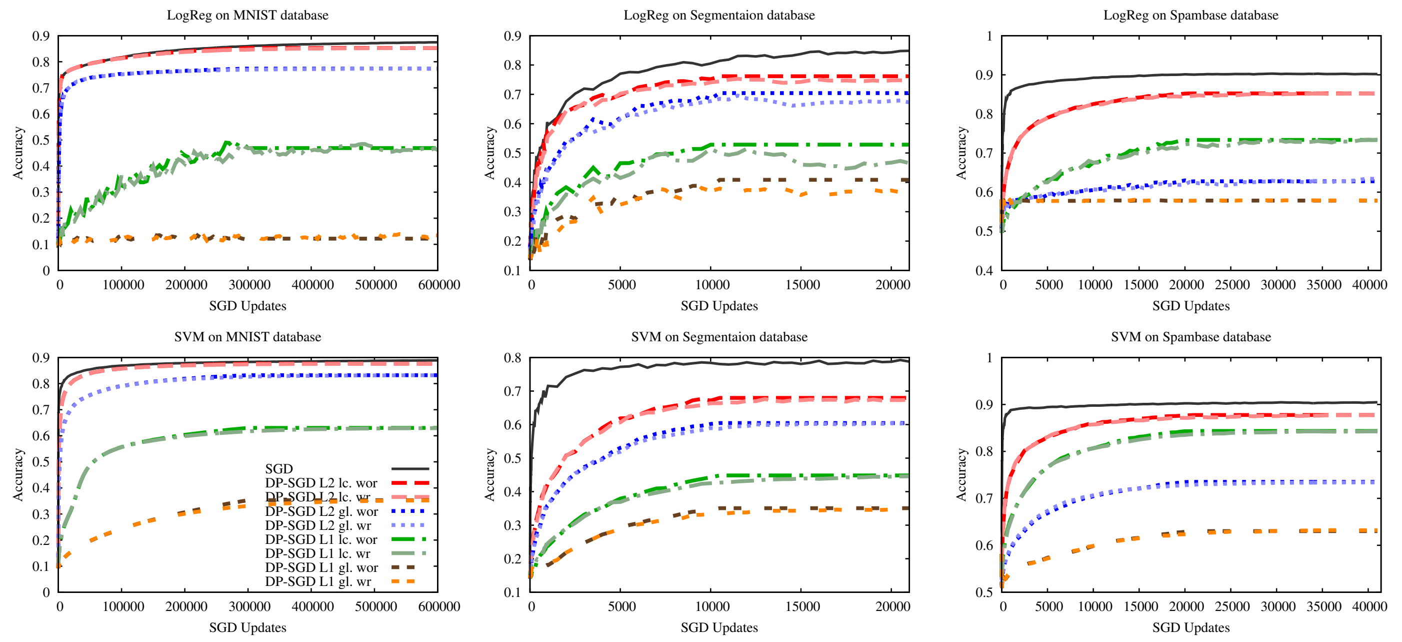

Fig. 2. Algorithm DP-SGD-5 with different vector norm (L^1 vs. L^2), normalization (local vs. global), and sampling (with or without replacement) with privacy budget $\epsilon = 1$. The x-axis spans a number of updates 10 times the size of the data set.

the two following normalization steps. As the first option, we projected every feature vector to the unit ball by normalizing their length to 1. Here, length is understood in terms of the L^2 norm or the L^1 norm, depending on the noise distribution in use. We call this type of normalization *local normalization*. In this case, the topological structure is somewhat distorted, but the signal-to-noise ratio is maximized as the amount of noise to be added is independent of the individual examples. As the other option, we tested *global normalization*, where the normalization coefficient was globally determined based on the vectors of maximal length. This way, only the lengths of the examples of maximal length become 1, the other vectors will become shorter than 1 and the topological structure of the data set is preserved. But, in this case, the signal-to-noise ratio might become very low for short vectors. Again, length is understood in terms of the L^2 norm or the L^1 norm, depending on the noise distribution in use.

C. Discussion

In order to measure the performance of the algorithms we computed their classification accuracy, namely the fraction of correctly classified instances:

$$\text{Accuracy} = \frac{1}{n}\sum_{i=1}^{n}\delta(y_i = f_w(x_i)), \tag{11}$$

where n is the number of test examples. We measured accuracy in regular intervals after a given number of updates. We then plotted the average of 20 independent measurements (where the various measurements differ due to the randomness of sampling the data).

The outcome of the experiments is shown in three sets of plots in Figures 1, 2 and 3 for the three different algorithms DP-SGD-1, DP-SGD-5 and DP-SGD-∞, respectively. From them, we can draw several interesting conclusions. First, applying the L^2 norm is clearly superior in all sets of experiments, independently of how the other parameters are set. The explanation might be that under the L^2 norm the added noise is much less likely to be "sparse", that is, noise is spread more evenly over the coordinates, which in turn causes less disturbance for the gradient steps.

Another clear, and somewhat surprising observation is that it is much better to apply local normalization on the training samples in all the cases we examined. This is good news because in our distributed setting local normalization is obviously a local operation and so it is much cheaper to implement than global normalization. The result is somewhat surprising though, because global normalization preserves the topological structure, whereas local normalization does not. However, the relative noise on a given node might be very large in the case of global normalization since many examples will be shrunk to less than the maximal length. This might be a more important factor than exact topological structure.

The effect of sampling the random walk (i.e., neighbor selection) is more subtle. Looking at algorithms DP-SGD-1 and DP-SGD-5 one might think that there is no significant difference between these sampling methods. However, in the case of DP-SGD-∞ the difference becomes dramatic. After a certain number of iterations sampling with replacement causes a major drop in accuracy. The reason is that with this sampling there will be many nodes that are visited much more often than 10 times (note that on average, each node is visited 10 times in all runs, but with sampling with replacement there is variance in the number of times a node is sampled). At the same time, the privacy budget for step t is only $\epsilon/2^t$, which results in an exponentially increasing amount of noise in each step. After a point noise becomes so large that the signal-to-noise ratio

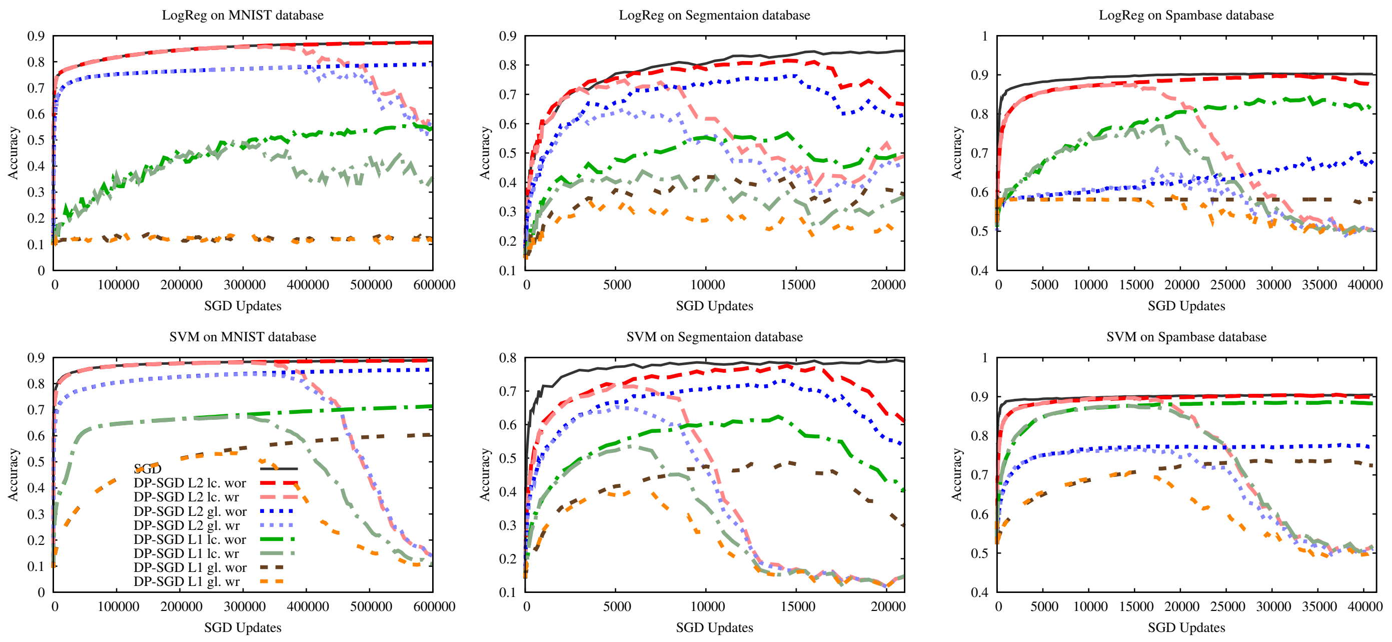

Fig. 3. Algorithm DP-SGD-∞ with different vector norm (L^1 vs. L^2), normalization (local vs. global), and sampling (with or without replacement) with privacy budget $\epsilon = 1$. The x-axis spans a number of updates 10 times the size of the data set.

tends to zero, at which point the accuracy starts to decrease. This is also true for sampling without replacement, only in that case the decrease in performance is observed only later because all nodes are visited an equal number of times so there are no outliers that inject large noise earlier.

The consequence of this last observation is that, although DP-SGD-∞ does achieve the best performance in a number of cases, it is less practical than the other options because one needs to implement sampling without replacement—a non-trivial task in fully distributed networks—and, more importantly, one needs to implement a stopping rule that detects when the accuracy starts to decrease. These goals are not impossible to accomplish, yet the other two algorithms can reach a similar performance while being much simpler and more robust.

Figure 4 illustrates the effect of the size of the privacy budget on the algorithms. It also allows us to compare the different algorithms directly under the best setting we found in the previous experiments.

Clearly, a lower privacy budget reduces the accuracy of all the algorithms, although to a different degree. Algorithm DP-SGD-1 seems to be the most robust. This is because although it performs only a single update step with a given node, at least that step has relatively little noise. Also, the size of the database plays an important role as well. With a small database (small network) we can make fewer gradient updates with reasonable noise. Indeed, with a database of infinite size one could achieve perfect convergence with an arbitrary finite amount of noise in each step as follows from the theory of SGD and stochastic approximation in general [5].

V. Conclusions

We evaluated an approach to privacy preserving SGD for distributed systems, where there are a large number of nodes, each of which stores small amounts of personal data. Since the privacy we achieve is just due to local operations, privacy can be compromised only if a given node is controlled by an adversary.

We argued that privacy can be maintained along with close to optimal accuracy and no communication overhead related to the unprotected distributed SGD algorithm. We also demonstrated that several design choices, such as the normalization of training data and the applied vector norm within the differential privacy scheme have a surprisingly large influence on the performance. We found that the L^2 norm and local normalization result in the best performance. In addition, depending on the privacy budget available, we found that it is advisable to use each example only a small number of times.

It should be mentioned that a better performance can be achieved if there is more training data on each node. Here, we focused on the worst case scenario where each node has only one record. If there are more records locally, one can rely on batch gradient descent, where the noise relative to the average batch gradient is smaller, and where the gradient itself carries less noise as well, so overall convergence will be more favorable.

References

[1] 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.

[2] A. Rial and G. Danezis, "Privacy-preserving smart metering," in *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society (WPES'11)*. New York, NY, USA: ACM, 2011, pp. 49–60.

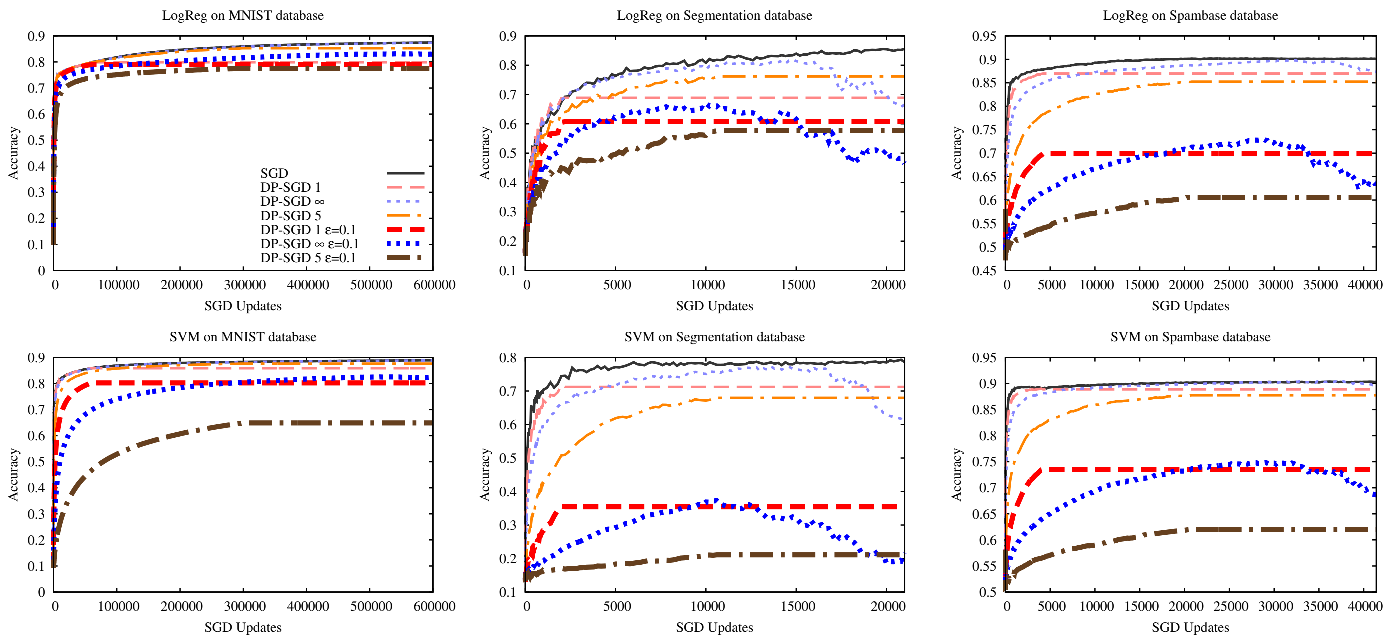

Fig. 4. The effect of the lower privacy budget $\epsilon = 0.1$ (i.e., higher privacy requirements) with the L^2 norm, local normalization, and uniform sampling without replacement. Curves with $\epsilon = 1$ are repeated for comparison.

[3] A. S. Pentland, "Society's nervous system: Building effective government, energy, and public health systems," *Computer*, vol. 45, no. 1, pp. 31–38, January 2012.

[4] C.-W. Tsai, C.-F. Lai, M.-C. Chiang, and L. Yang, "Data mining for internet of things: A survey," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 1, pp. 77–97, 2014.

[5] L. Bottou and Y. LeCun, "Large scale online learning," in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. Saul, and B. Schölkopf, Eds. Cambridge, MA: MIT Press, 2004.

[6] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade*, ser. LNCS, G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Springer Berlin Heidelberg, 2012, vol. 7700, pp. 421–436.

[7] R. Ormándi, I. Hegedűs, and M. Jelasity, "Gossip learning with linear models on fully distributed data," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 4, pp. 556–571, 2013.

[8] C. Dwork, "A firm foundation for private data analysis," *Commun. ACM*, vol. 54, no. 1, pp. 86–95, January 2011.

[9] A. Friedman and A. Schuster, "Data mining with differential privacy," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'10)*. New York, NY, USA: ACM, 2010, pp. 493–502.

[10] P. Mohan, A. Thakurta, E. Shi, D. Song, and D. Culler, "Gupt: privacy preserving data analysis made easy," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD'12)*. New York, NY, USA: ACM, 2012, pp. 349–360.

[11] K. Chaudhuri and C. Monteleoni, "Privacy-preserving logistic regression," in *Advances in Neural Information Processing Systems 21, (NIPS)*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Curran Associates, Inc., 2008, pp. 289–296.

[12] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *J. Mach. Learn. Res.*, vol. 12, pp. 1069–1109, July 2011.

[13] S. Song, K. Chaudhuri, and A. D. Sarwate, "Stochastic gradient descent with differentially private updates," in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, December 2013, pp. 245–248.

[14] T. M. Mitchell, *Machine Learning*, 2nd ed., E. M. Munson, Ed. New York: McGraw-Hill, 1997.

[15] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: primal estimated sub-gradient solver for SVM," *Mathematical Programming B*, 2010.

[16] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[17] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *J. Mach. Learn. Res.*, vol. 2, pp. 265–292, March 2002.

[18] N. Tölgyesi and M. Jelasity, "Adaptive peer sampling with newscast," in *Euro-Par 2009*, ser. LNCS, H. Sips, D. Epema, and H.-X. Lin, Eds., vol. 5704. Springer-Verlag, 2009, pp. 523–534.

[19] K. Birman, M. Jelasity, R. Kleinberg, and E. Tremel, "Building a secure and privacy-preserving smart grid," *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 131–136, January 2015.

[20] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography*, ser. LNCS, S. Halevi and T. Rabin, Eds. Springer Berlin Heidelberg, 2006, vol. 3876, pp. 265–284.

[21] F. D. McSherry, "Privacy integrated queries: An extensible platform for privacy-preserving data analysis," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '09. New York, NY, USA: ACM, 2009, pp. 19–30.

[22] Y. Lecun, C. Cortes, and B. Christopher, J.C., "The MNIST database of handwritten digits."

[23] K. Bache and M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml