A Differentially Private Stochastic Gradient Descent Algorithm for Multiparty Classification

Arun Rajkumar and Shivani Agarwal
Department of Computer Science and Automation
Indian Institute of Science, Bangalore 560012, India
{arun_r, shivani}@csa.iisc.ernet.in

Abstract

We consider the problem of developing privacy-preserving machine learning algorithms in a distributed multiparty setting. Here different parties own different parts of a data set, and the goal is to learn a classifier from the entire data set without any party revealing any information about the individual data points it owns. Pathak et al [7] recently proposed a solution to this problem in which each party learns a local classifier from its own data, and a third party then aggregates these classifiers in a privacy-preserving manner using a cryptographic scheme. The generalization performance of their algorithm is sensitive to the number of parties and the relative fractions of data owned by the different parties. In this paper, we describe a new differentially private algorithm for the multiparty setting that uses a stochastic gradient descent based procedure to directly optimize the overall multiparty objective rather than combining classifiers learned from optimizing local objectives. The algorithm achieves a slightly weaker form of differential privacy than that of [7], but provides improved generalization guarantees that do not depend on the number of parties or the relative sizes of the individual data sets. Experimental results corroborate our theoretical findings.

1 Introduction

The traditional paradigm in machine learning has been that one is given a data set, and the goal is to learn a predictive model (such as a classifier) from this data set. Increasingly, however, one finds that data is distributed among multiple parties: financial data is distributed across multiple banks and credit agencies; medical records are distributed across multiple hospitals and health care institutions and so on. In such a setting, it becomes important to design algorithms that can learn a model from the overall (combined) data, while preserving the privacy of individual parties' data points. Differential privacy is a strong notion of privacy that first arose in the cryptography community [1, 2]; briefly, it ensures that the output of an algorithm operating on a data set does not allow any individual data point to be identified. There has been some interest recently in designing differentially private machine learning algorithms, with private versions of algorithms such as logistic regression and support vector machines being proposed [3, 4, 5]. These algorithms mostly focus on the single-party setting, where a single party owns the data, but due to its sensitive nature, it is important that any model that is learned from the data and released publicly does not reveal any information about individual data points. Recently, Pathak et al [7] considered learning a differentially private classifier in the multiparty setting, where in addition to not releasing information about individual data points to the public, one cannot share information about individual points among the different parties. The solution proposed by Pathak et al involves each party learning a local classifier on its own data, and then sharing this classifier in a privacy-preserving manner (through a cryptographic scheme) with a third party that constructs a differentially private aggregate classifier.

In this paper, we propose a different solution to the multiparty problem. Our algorithm is based on a stochastic gradient descent procedure, and amounts to performing Gaussian objective perturbation on the overall multiparty objective. The algorithm achieves a slightly weaker form of differential privacy than that of [7], but is more robust to the number of parties and the relative fractions of data owned by the different parties, which is reflected in the sample complexity and excess error bounds we obtain. We also contrast our method with that of [7] via experimental simulations and point out situations where our method outperforms the current state of the art for this problem.

Appearing in Proceedings of the 15^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2012, La Palma, Canary Islands. Volume XX of JMLR: W&CP XX.

Organization of the paper. We review some background material in Section 2, including the definition of differential privacy, current methods for achieving differential privacy in machine learning, the multiparty setting, and the algorithm of [7]. We describe our algorithm in Section 3. This is followed by privacy analysis in Section 4, generalization analysis in Section 5, and experimental results in Section 6.

2 Preliminaries and Background

2.1 Differential Privacy

Differential privacy [1] is a notion of privacy that guarantees that the presence or absence of any specific entry in a data set will not affect the output of an algorithm by much:

Definition. A randomized algorithm $\mathcal{A}$ is *ϵ-differentially private* if for all data sets D, D' that differ in a single element and all $\omega \in \Omega$, where Ω is the output space of $\mathcal{A}$:[1]

$$\frac{\mathbf{Pr}[\mathcal{A}(D) = \omega]}{\mathbf{Pr}[\mathcal{A}(D') = \omega]} \leq e^{\epsilon}\,.$$

The probability is over the randomization in the algorithm.

2.2 Differential Privacy in Machine Learning

Consider the standard problem of binary classification in machine learning, where one is given a data set $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ consisting of N example input points $\mathbf{x}_i$ from some space X with class labels $y_i \in \{\pm 1\}$, assumed to be drawn iid from some fixed but unknown distribution $\mathcal{Q}$ over $X \times \{\pm 1\}$, and the goal is to learn from D a classifier $h : X \to \{\pm 1\}$ that performs well on future examples from $\mathcal{Q}$. A number of authors have recently considered the design of differentially private algorithms for classification [3, 4, 5]. These works consider vector-valued data $X \subseteq \mathbb{R}^d$, and focus mainly on algorithms that learn from D a linear classifier of the form $h(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x})$, represented by a weight vector $\mathbf{w} \in \mathbb{R}^d$.

A popular class of algorithms for learning a linear classifier is that of regularized empirical risk minimization (ERM) algorithms, which minimize an objective of the form

$$J(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} \phi(y_i \mathbf{w}^\top \mathbf{x}_i) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2\,, \tag{1}$$

where $\|\mathbf{w}\|_2^2$ is the regularizing term, $\lambda > 0$ is a regularization constant, and ϕ is a margin-based loss function (such as the logistic loss $\ln(1 + \exp(-y_i \mathbf{w}^\top \mathbf{x}_i))$ which leads to logistic regression, or the hinge loss $\max(0, 1 - (y_i \mathbf{w}^\top \mathbf{x}_i))$ which leads to support vector machines). A regularized ERM algorithm by itself is clearly deterministic; two methods have been proposed to introduce randomization in such an algorithm so as to achieve differential privacy: output perturbation and objective perturbation.

[1] Here $\mathbf{Pr}$ denotes probability mass for discrete output spaces and probability density for continuous output spaces.

Output perturbation. This is based on a general method described in [1], in which one adds noise to the output of an underlying deterministic algorithm. The form of noise added depends on the sensitivity of the underlying algorithm, which measures the maximum change in the output of the algorithm when a single element in the input data set is changed [1, 4]:

Definition. Consider a vector-valued function $g : Z^N \to \mathbb{R}^d$, where Z is any arbitrary set. The L_2 *sensitivity* of g is defined as

$$S(g) = \max_i \max_{z_1,\ldots,z_N,z_i'} \|g(z_1, \ldots, z_i, \ldots, z_N) - g(z_1, \ldots, z_i', \ldots, z_N)\|_2\,.$$

In the above equation g can be viewed as a (deterministic) algorithm that computes a statistic of the data set $D = \{z_1, \ldots, z_N\}$. Given such an algorithm that has sensitivity $S(g) \leq C$ and a desired privacy parameter ϵ, the output perturbation method returns $g(D) + \boldsymbol{\eta}$, where $\boldsymbol{\eta} \in \mathbb{R}^d$ is a noise vector generated randomly according to the density

$$\mu(\boldsymbol{\eta}) \quad \propto \quad \exp\left(-\beta\|\boldsymbol{\eta}\|_2\right), \tag{2}$$

where $\beta = \frac{\epsilon}{C}$. It can be shown that the above output perturbed version of the algorithm g provides ϵ-differential privacy [1, 4]. It can also be shown that several regularized ERM algorithms, when viewed as taking as input a data set $D \in (\mathbb{R}^d \times \{-1, 1\})^N$ and producing as output a weight vector $g(D) = \mathbf{w}_D$ that minimizes the objective in Eq. (1), have bounded sensitivity, and therefore output perturbation can be applied to these algorithms; this includes for example logistic regression, which (assuming all data points lie in a unit ball) has L_2 sensitivity at most $\frac{2}{\lambda N}$ [4, 5].

Objective perturbation. In this method [4], instead of randomly perturbing the output classifier, one perturbs the objective minimized by the algorithm. Specifically, the objective perturbed version of regularized ERM minimizes

$$\widetilde{J}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} \phi(y_i \mathbf{w}^\top \mathbf{x}_i) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \frac{1}{N}\boldsymbol{\eta}^\top \mathbf{w}\,, \tag{3}$$

where $\boldsymbol{\eta}$ is again a vector drawn according to a density of the form in Eq. (2) (but with a different parameter β). It can be shown that under certain assumptions on the form of the loss ϕ and the regularization term and with the right choice of the parameter β in the above density, one can prove ϵ-differential privacy for objective perturbation as well [4].

2.3 Multiparty Classification

Consider now binary classification in a multiparty setting: the training data set $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ is now distributed among K parties $P_1, P_2, \ldots, P_K$, so that the k^{th} party P_k possesses a subset of the data $D_k \subseteq D$; it is assumed that the indices of data points in D that are assigned to any two distinct parties P_k, P_l $(k \neq l)$ are disjoint

(i.e. that $\{D_1, \ldots, D_K\}$ forms a *partition* of D). The goal as before is to learn a classifier $h : X \to \{\pm 1\}$ from the complete data set D. However, each party wishes not to disclose any information about any of the individual data points it owns. The challenge therefore lies in learning a classifier that preserves the privacy of each party.

2.4 Related Work

The multiparty classification problem was considered recently by Pathak et al [7]. The solution they propose involves a third party which gathers information from the individual parties and computes a private classifier. Specifically, each party P_k computes a local classifier $\mathbf{w}_k \in \mathbb{R}^d$ by minimizing the local regularized ERM objective on its own data set $D_k = \{(\mathbf{x}_1^k, y_1^k), \ldots, (\mathbf{x}_{N_k}^k, y_{N_k}^k)\}$:

$$J_k(\mathbf{w}) = \frac{1}{N_k} \sum_{j=1}^{N_k} \phi(y_j^k \mathbf{w}^\top \mathbf{x}_j^k) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \,. \tag{4}$$

The third party aims to aggregate the local classifiers $\mathbf{w}_k$ through averaging and release an output perturbed version of the average:

$$\mathbf{w}_{\text{priv}} = \frac{1}{K} \sum_{k=1}^{K} \mathbf{w}_k + \boldsymbol{\eta} \,, \tag{5}$$

where $\boldsymbol{\eta}$ is an appropriate noise vector, e.g, with density as in Eq. (2), with appropriate β.[2] A straightforward implementation of this idea is problematic since directly sharing $\mathbf{w}_k$ runs the risk of the third party potentially being able to identify some of the individual data points in D_k from which $\mathbf{w}_k$ was learned; to overcome this difficulty, Pathak et al propose a cryptographic scheme based on additively homomorphic encryption, which allows the third party to securely compute the above aggregate classifier without getting direct access to any of the individual classifiers $\mathbf{w}_k$. Pathak et al also provide theoretical analysis comparing the above classifier $\mathbf{w}_{\text{priv}}$ with the minimizer $\mathbf{w}^*$ of the overall multiparty regularized ERM objective $J(\mathbf{w})$ (Eq. (1)); in particular, they obtain a high probability bound (over $D \sim Q^N$ and the noise vector $\boldsymbol{\eta}$) on the excess ϕ-error of $\mathbf{w}_{\text{priv}}$ over that of $\mathbf{w}^*$ (i.e. on the difference between the expected ϕ-loss of $\mathbf{w}_{\text{priv}}$ and that of $\mathbf{w}^*$, $\mathbf{E}_{(\mathbf{x},y)\sim\mathcal{Q}}[\phi(y\mathbf{w}_{\text{priv}}^\top \mathbf{x})] - \mathbf{E}_{(\mathbf{x},y)\sim\mathcal{Q}}[\phi(y\mathbf{w}^{*\top}\mathbf{x})]$).

While the algorithm of Pathak et al [7] is simple conceptually, it is not clear why averaging local regularized ERM classifiers learned by the individual parties is the right way to approximate the overall regularized ERM classifier; indeed, the excess error bounds of Pathak et al over the minimizer $\mathbf{w}^*$ of the overall multiparty objective become looser as the number of parties K increases or the size of the smallest data set $\min_k N_k$ decreases. In what follows, we propose a privacy-preserving algorithm that directly attempts to approximate the minimizer $\mathbf{w}^*$ of the overall multiparty regularized ERM objective $J(\mathbf{w})$ (Eq. (1)). The algorithm is based on a stochastic gradient descent procedure; in fact, as we will see, the algorithm can equivalently be viewed as performing objective perturbation on the multiparty objective, although with a different type of noise than that used in the (single-party) objective perturbation algorithm of [4]. Due to the different noise, we are able to guarantee only a slightly weaker form of (ϵ, δ)-differential privacy (this will be defined later); however the excess error bounds that we prove over $\mathbf{w}^*$ are independent of the number of parties involved and the size of the smallest data set, and in this sense are superior to the bounds of [7].

[2] The work of Pathak et al [7] actually uses a noise vector $\boldsymbol{\eta}$ drawn from a multivariate Laplace density $\mu(\boldsymbol{\eta}) \propto \exp(-\beta\|\boldsymbol{\eta}\|_1)$ for appropriate β; however the density described above also achieves ϵ-differential privacy.

3 A New Algorithm for Differentially Private Multiparty Classification

3.1 Basic Idea

Let us begin by considering a third party which wants to minimize the overall multiparty objective $J(\mathbf{w})$ (see Eq. (1)) by running a gradient descent algorithm. We will assume that the loss function ϕ is convex and differentiable, which will ensure that gradients exist and that the minimizer of the objective is unique. To run such an algorithm, all that the third party needs is the gradient information at any given $\mathbf{w}_t$. The gradient of $J(\mathbf{w})$ at $\mathbf{w}_t$ is given by

$$\nabla J(\mathbf{w}_t) = \frac{1}{N} \sum_{i=1}^{N} \phi'(y_i \mathbf{w}_t^\top \mathbf{x}_i)(y_i \mathbf{x}_i) + \lambda \mathbf{w}_t \,, \tag{6}$$

which can equivalently be written as

$$\nabla J(\mathbf{w}_t) = \frac{1}{N} \sum_{k=1}^{K} \sum_{j=1}^{N_k} \phi'(y_j^k \mathbf{w}_t^\top \mathbf{x}_j^k)(y_j^k \mathbf{x}_j^k) + \lambda \mathbf{w}_t \,. \tag{7}$$

Clearly, the gradient of $J(\mathbf{w})$ at any $\mathbf{w}_t$ can be computed from pieces of information from the K different parties. In particular, if party P_k provides the gradient

$$\nabla G_k(\mathbf{w}_t) = \sum_{j=1}^{N_k} \phi'(y_j^k \mathbf{w}_t^\top \mathbf{x}_j^k)(y_j^k \mathbf{x}_j^k) \tag{8}$$

of the local cumulative loss $G_k(\mathbf{w}) = \sum_{j=1}^{N_k} \phi(y_j^k \mathbf{w}^\top \mathbf{x}_j^k)$, then the third party can compute $\nabla J(\mathbf{w}_t)$ by combining these as follows:

$$\nabla J(\mathbf{w}_t) = \frac{1}{N} \sum_{k=1}^{K} \nabla G_k(\mathbf{w}_t) + \lambda \mathbf{w}_t \,. \tag{9}$$

This method is sufficient to ensure that the third party, by running a gradient descent procedure with appropriately chosen $\mathbf{w}_t$, converges to the minimizer of $J(\mathbf{w})$. However, so far, we have not taken any privacy considerations into account.

3.2 Privacy Considerations: Attempt 1

In sharing the gradients $\nabla G_k(\mathbf{w}_t)$ as above, which are computed from the local data set D_k, a party P_k runs the risk of potentially compromising the privacy of its individual data points. To preserve the privacy of its data, each party can consider providing a differentially private version of these gradients. Specifically, if we view each party as running an algorithm that given D_k as input computes $\nabla G_k(\mathbf{w}_t)$ as output (at some specified $\mathbf{w}_t$), we can obtain a differentially private version via output perturbation. In particular, note that the algorithm computing the gradient has bounded sensitivity:

Proposition 3.1. If the loss function ϕ is L-Lipschitz and all data points lie in a unit ball, then the L_2 sensitivity of the algorithm that computes $\nabla G_k(\mathbf{w}_t)$ from D_k is at most $2L$.

The proof is provided in the appendix. Thus, when requested for gradient information at $\mathbf{w}_t$, party P_k can return the following:

$$\nabla \widehat{G}_k(\mathbf{w}_t) = \nabla G_k(\mathbf{w}_t) + \boldsymbol{\rho}_t^k \,, \tag{10}$$

where $\boldsymbol{\rho}_t^k \in \mathbb{R}^d$ is an appropriate noise vector, e.g. with density as in Eq. (2), with appropriate β. In particular, for the gradient evaluation at $\mathbf{w}_t$ to achieve ϵ-differential privacy, one can take $\beta = \frac{\epsilon}{2L}$.

Privacy breach. Unfortunately, the above is not sufficient to guarantee ϵ-differential privacy of the procedure as a whole. To see this, note that each of the parties has added a zero mean, finite variance noise to the output that it sends to the third party. The third party now normalizes the sum of these outputs by N and adds $\lambda \mathbf{w}_t$ to it:

$$\nabla \widehat{J}(\mathbf{w}_t) = \frac{1}{N} \sum_{k=1}^{K} \nabla \widehat{G}_k(\mathbf{w}_t) + \lambda \mathbf{w}_t \,. \tag{11}$$

One can view the third party as having now computed a noisy version of the gradient of $J(\mathbf{w})$ at $\mathbf{w}_t$, where the noise added is a (finite) sum of zero mean, finite (co)variance random vectors, and is therefore a zero mean, finite (co)variance random vector itself:

$$\nabla \widehat{J}(\mathbf{w}_t) = \nabla J(\mathbf{w}_t) + \frac{1}{N} \sum_{k=1}^{K} \boldsymbol{\rho}_t^k \,. \tag{12}$$

It is well known in the stochastic approximation literature [8] that if only noisy versions of the gradients of a function are available, then as long as the noise is a zero mean, finite (co)variance random vector, by choosing suitable step sizes, one can converge to the minimizer of the function almost surely. This remarkable property then implies that the third party can converge to the true minimizer $\mathbf{w}^*$ of the overall objective $J(\mathbf{w})$, which can potentially allow it to recover information about some of the individual data points in D. Similarly, by performing gradient descent on the noisy gradients of only the k^{th} party P_k, the third party can also converge to the minimizer of party P_k's objective $J_k(\mathbf{w})$, thereby allowing it to potentially recover information specifically about the data points in D_k.

So what went wrong even after we perturbed the gradients at each iteration? The problem is that while each perturbed gradient $\nabla \widehat{G}_k(\mathbf{w}_t)$ is ϵ-differentially private, after providing T such gradients, say at $\mathbf{w}_1, \ldots, \mathbf{w}_T$, any party P_k is assured of only $(T\epsilon)$-differential privacy. To see this, note that after T iterations, the (perturbed) output computed from D_k actually consists of $(\nabla \widehat{G}_k(\mathbf{w}_1), \ldots, \nabla \widehat{G}_k(\mathbf{w}_T)) \in (\mathbb{R}^d)^T$. Therefore, if D'_k differs from D_k in a single element and $\nabla \widehat{G}'_k(\mathbf{w}_t)$ denotes the (perturbed) gradient at $\mathbf{w}_t$ computed from D'_k, then by independence of the perturbations at different iterations, we have

$$\begin{aligned} &\frac{\mathbf{Pr}[(\nabla \widehat{G}_k(\mathbf{w}_1), \ldots, \nabla \widehat{G}_k(\mathbf{w}_T)) = (\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_T)]}{\mathbf{Pr}[(\nabla \widehat{G}'_k(\mathbf{w}_1), \ldots, \nabla \widehat{G}'_k(\mathbf{w}_T)) = (\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_T)]} \\ &= \frac{\mathbf{Pr}[\nabla \widehat{G}_k(\mathbf{w}_1) = \boldsymbol{\omega}_1]}{\mathbf{Pr}[\nabla \widehat{G}'_k(\mathbf{w}_1) = \boldsymbol{\omega}_1]} \cdots \frac{\mathbf{Pr}[\nabla \widehat{G}_k(\mathbf{w}_T) = \boldsymbol{\omega}_T]}{\mathbf{Pr}[\nabla \widehat{G}'_k(\mathbf{w}_T) = \boldsymbol{\omega}_T]} \\ &\leq (e^\epsilon)^T = e^{T\epsilon} \,. \end{aligned} \tag{13}$$

Thus after a large number of iterations T, the privacy guarantee of the above approach is effectively meaningless.

One possible solution to the above problem is to vary ϵ on each iteration, i.e. to require the (perturbed) gradient at $\mathbf{w}_t$ to be ϵ_t-differentially private, where $\sum_{t=1}^{\infty} \epsilon_t \leq \epsilon$ (e.g. one can take $\epsilon_t = \frac{\epsilon}{2^t}$). This will clearly ensure that each party is guaranteed ϵ-differential privacy overall, even after an arbitrarily large number of iterations T. However, as ϵ_t becomes smaller, the noise $\boldsymbol{\rho}_t^k$ (which is now sampled from the density in Eq. (2) with $\beta_t = \frac{\epsilon_t}{2L}$) has larger and larger (co)variance. Consequently, after a sufficiently large number of iterations, the perturbed gradients will no longer have much relation to the original gradients – in fact they will be highly random – and therefore the weight vector learned by the gradient descent procedure on these highly noisy gradients will also have a high degree of randomness, with relatively little dependence on the actual data D. This means that the classifier learned by such an approach is unlikely to have good generalization performance.

Below we propose an alternative solution that will allow us to preserve the privacy of individual parties' data points while also providing good generalization guarantees.

3.3 Privacy Considerations: Attempt 2

We saw above that perturbing the individual parties' gradients according to Eq. (10) allows the gradient descent procedure to converge to the exact minimizer $\mathbf{w}^*$ of the overall multiparty objective $J(\mathbf{w})$, thereby compromising privacy. Instead, we would like the procedure to learn a weight vector that is close enough to $\mathbf{w}^*$ to yield good generaliza-

tion performance, but that is sufficiently random to provide good privacy guarantees. To this end, we consider adding another noise vector $\boldsymbol{\eta}^k$, which is sampled only *once* by each party P_k, to each gradient:

$$\nabla \widehat{G}_k(\mathbf{w}_t) = \nabla G_k(\mathbf{w}_t) + \boldsymbol{\rho}_t^k + \boldsymbol{\eta}^k \,, \tag{14}$$

where $\boldsymbol{\rho}_t^k \in \mathbb{R}^d$ is sampled from the density in Eq. (2) with $\beta = \frac{\epsilon}{2L}$ as before; the distribution of $\boldsymbol{\eta}^k$ will be described shortly.

Now, even if the third party runs a gradient descent procedure infinitely with the noisy gradients from each party given as above, it can only get rid of the effect of the $\boldsymbol{\rho}_t^k$'s, whereas the effect of the fixed noise $\boldsymbol{\eta}^k$ remains constant. In particular, the third party now has noisy gradients of the form

$$\nabla \widehat{J}(\mathbf{w}_t) = \frac{1}{N}\sum_{k=1}^{K} \nabla \widehat{G}_k(\mathbf{w}_t) + \lambda \mathbf{w}_t \tag{15}$$

$$= \nabla J(\mathbf{w}_t) + \frac{1}{N}\sum_{k=1}^{K} \boldsymbol{\eta}^k + \frac{1}{N}\sum_{k=1}^{K} \boldsymbol{\rho}_t^k \,, \tag{16}$$

and therefore can converge only to the minimizer of

$$\widetilde{J}(\mathbf{w}) = J(\mathbf{w}) + \left(\frac{1}{N}\sum_{k=1}^{K} \boldsymbol{\eta}^k\right)^\top \mathbf{w} \,. \tag{17}$$

Note that $\boldsymbol{\rho}_t^k$'s do not feature in the above objective as they can be viewed as noise vectors which corrupt gradient evaluations of the above objective (the corrupting noise must be sampled independently on each iteration for its effect to be removed, which is why $\boldsymbol{\eta}^k$ still appears in the objective). Also observe that the above objective is just a perturbed version of the overall multiparty objective $J(\mathbf{w})$. Similarly, if the third party attempts to run gradient descent using only party P_k's noisy gradients, it can now converge to the minimizer of only $\widetilde{J}_k(\mathbf{w}) = J_k(\mathbf{w}) + \frac{1}{N_k}{\boldsymbol{\eta}^k}^\top \mathbf{w}$.

We will show below that with an appropriate choice of noise vectors $\boldsymbol{\eta}^k$ one can obtain both privacy and generalization guarantees for the above procedure. Before this we answer a few questions that remain.

Are $\boldsymbol{\rho}_t^k$'s Still Necessary?
A natural question is: what role do the $\boldsymbol{\rho}_t^k$'s now play in the above procedure? Since their effect will in any case be eliminated by the gradient descent procedure, is it still necessary to include them? To answer this question, consider the case where the k^{th} party P_k does not add the $\boldsymbol{\rho}_t^k$ noise vectors. Thus when requested for a gradient at $\mathbf{w}_t$, party P_k simply returns

$$\nabla \widehat{G}_k(\mathbf{w}_t) = \nabla G_k(\mathbf{w}_t) + \boldsymbol{\eta}^k \,. \tag{18}$$

Now consider the difference of gradient evaluations by party P_k at consecutive iterations t and $t+1$:

$$\nabla \widehat{G}_k(\mathbf{w}_t) - \nabla \widehat{G}_k(\mathbf{w}_{t+1}) = \nabla G_k(\mathbf{w}_t) - \nabla G_k(\mathbf{w}_{t+1}) \tag{19}$$

As can be seen, the effect of the noise $\boldsymbol{\eta}^k$ has been eliminated. The third party could potentially now use the difference between these true gradients to obtain information about the data points in D_k from which these gradients are computed, thereby compromising the privacy of P_k. Adding the $\boldsymbol{\rho}_t^k$ vectors that get sampled during every iteration ensures that such a privacy leakage is prevented.

Distribution of $\boldsymbol{\eta}^k$: This distribution from which $\boldsymbol{\eta}^k$ vectors are sampled will be critical in proving the privacy of the classifier that the third party computes. Recall from above that the algorithm proposed can be viewed as minimizing a perturbed form of $J(\mathbf{w})$, the perturbation being given by $\boldsymbol{\eta}^\top \mathbf{w}$ where $\boldsymbol{\eta} = \frac{1}{N}\sum_{k=1}^{K} \boldsymbol{\eta}^k$. We know from [4] that such an objective perturbation will preserve ϵ-differential privacy if $\boldsymbol{\eta}$ has density of the form in Eq. (2), with appropriate β. However we cannot apply this result directly to our setting since it is not clear how to choose the independent $\boldsymbol{\eta}^k$ such that the resulting (normalized) sum $\boldsymbol{\eta}$ is distributed as Eq. (2). Instead, we will consider taking each $\boldsymbol{\eta}^k$ to be a multivariate Gaussian with zero mean and diagonal covariance matrix with diagonal entries $\frac{\sigma^{*2}}{K}$. The sum $\sum_{k=1}^{K} \boldsymbol{\eta}^k$ is then again a multivariate Gaussian with zero mean and diagonal covariance matrix, with diagonal entries given by σ^{*2}. In the next section, we show that by choosing an appropriate value for σ^*, we can guarantee a slightly weaker form of (ϵ, δ)-differential privacy. The final algorithm, consisting of the third party's functionality together with the individual parties' functionality, is shown in Algorithm 1.

Algorithm 1 PSGD (Private Stochastic Gradient Descent: Third Party's Functionality)

Input:
1. Handles $(P_k)_{k=1}^K$
2. Constants: ϵ, δ,**Max_iterations**, λ, N, c (upper bound on $\phi''(z)\ \forall z$)
3. $\{z_t\}$ such that $\sum_{t=1}^{\infty} z_t = \infty$ and $\sum_{t=1}^{\infty} z_t^2 < \infty$

Output: Differentially private classifier $\mathbf{w}_{\text{psgd}}$

[Δ, $\tilde{\epsilon}$] = **computeSlack**(ϵ,λ,N,c)
Initialize $\mathbf{w}_0$
$t \leftarrow 0$
while $t <$ **Max_iterations do**
  for $k = 1$ **to** K **do**
    $\nabla G_k(\mathbf{w}_t) = \boldsymbol{P_k(\mathbf{w}_t, \tilde{\epsilon}, \delta)}$
  end for
  $\nabla J(\mathbf{w}_t) \leftarrow \frac{1}{N}\sum_{k=1}^{K} \nabla G_k(\mathbf{w}_t) + (\lambda + \Delta)\mathbf{w}_t$
  $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - z_t \nabla J(\mathbf{w}_t)$
end while
$\mathbf{w}_{\text{psgd}} \leftarrow \mathbf{w}_t$
return $\mathbf{w}_{\text{psgd}}$

Procedure 2 computeSlack

Input: ϵ, λ, N, c
Output: Δ, $\tilde{\epsilon}$

$\tilde{\epsilon} = \epsilon - 2\log(1 + \frac{c}{N\lambda})$
if $\tilde{\epsilon} > 0$ **then**
  $\Delta = 0$
else
  $\Delta = \frac{c}{N(\exp(\frac{\epsilon}{4})-1)} - \lambda$
  $\tilde{\epsilon} = \frac{\epsilon}{2}$
end if
return $[\Delta, \tilde{\epsilon}]$

Procedure 3 P_k (Individual Party P_k's Functionality)

Input:

1. Data set $D_k = (\mathbf{x}_j^k, y_j^k)_{j=1}^{N_k}$ corresponding to party P_k
2. Current iterate of classifier $\mathbf{w}_t$
3. Constants: $\tilde{\epsilon}$, δ, L (Lipschitz constant for ϕ)
4. Noise vector $\boldsymbol{\eta}^k$ sampled from a multivariate Gaussian with mean $\mathbf{0}$ and diagonal covariance matrix with diagonal entries $\frac{\sigma^{*2}}{K}$ where $\sigma^* := \sigma^*(d, \tilde{\epsilon}, \delta)$ is chosen such that

$$\mathbf{Pr}\left(U \leq \frac{(\sigma^*\tilde{\epsilon} - 2)^2}{4\sigma^{*2}}\right) = 1 - \delta$$

where U is a χ^2 distributed random variable with d degrees of freedom. (Note that $\boldsymbol{\eta}^k$ is sampled *once* by each party and the same vector is used in all iterations)

Output: Perturbed gradient $\nabla\widehat{G}_k(\mathbf{w}_t)$ of cumulative loss of $\mathbf{w}_t$ on D_k

$\nabla G_k(\mathbf{w}_t) = \sum_{j=1}^{N_k} \phi'(y_j^k \mathbf{w}_t^\top \mathbf{x}_j^k)(y_j^k \mathbf{x}_j^k)$
Draw $\boldsymbol{\rho}_t^k$ according to density in Eq. (2) with $\beta = \frac{\epsilon}{2L}$
$\nabla\widehat{G}_k(\mathbf{w}_t) \leftarrow \nabla G_k(\mathbf{w}_t) + \boldsymbol{\rho}_t^k + \boldsymbol{\eta}^k$
return $\nabla\widehat{G}_k(\mathbf{w}_t)$

4 Privacy Analysis

Recall from above that the stochastic gradient descent procedure described in Section 3 (Algorithm 1) effectively finds the minimizer of a perturbed form of the overall (multiparty) regularized ERM objective $J(\mathbf{w})$ (Eq. (1)), with the perturbation vector $\boldsymbol{\eta} \in \mathbb{R}^d$ being sampled from a multivariate Gaussian density with zero mean and diagonal covariance matrix. We therefore first obtain a result on the privacy of such a Gaussian objective perturbed regularized ERM classifier, and then apply this result to our setting. Before this we define the notion of (ϵ, δ)-differential privacy [10] which will be used in our results.

Definition. A randomized algorithm $\mathcal{A}$ is *(ϵ, δ)-differentially private* if for all data sets D, we can divide the output space Ω into two sets Ω_1 and Ω_2 (possibly depending on D) such that $\mathbf{Pr}[\mathcal{A}(D) \in \Omega_1] \leq \delta$ and for all data sets D' that differ from D in a single element and all $\omega \in \Omega_2$,

$$\frac{\mathbf{Pr}[\mathcal{A}(D) = \omega]}{\mathbf{Pr}[\mathcal{A}(D') = \omega]} \leq e^\epsilon .$$

This definition guarantees that algorithm $\mathcal{A}$ achieves ϵ-differential privacy with probability at least $1 - \delta$. The set Ω_1 contains all outputs that are considered privacy breaches according to ϵ-differential privacy; the probability of such an output is bounded by δ.

Privacy of Gaussian objective perturbation. Consider now the Gaussian objective perturbed regularized ERM algorithm shown in Algorithm 4. It can be shown that the algorithm provides (ϵ, δ)-differential privacy:

Algorithm 4 GOP (Gaussian Objective Perturbation)

Input:

1. Data set $D = (\mathbf{x}_j, y_j)_{j=1}^N$
2. Constants: $\epsilon, \delta, \lambda, c$

Output: (ϵ, δ)-differentially private classifier $\mathbf{w}_{\text{gop}}$

$[\Delta, \tilde{\epsilon}]$ = **computeSlack**(ϵ,λ,N,c)
Draw a vector $\boldsymbol{\eta} \in \mathbb{R}^d$, with each component of $\boldsymbol{\eta}$ drawn independently from $\mathcal{N}(0, \sigma^{*2})$ where σ^* is chosen as mentioned in Procedure 3
Compute $\mathbf{w}_{\text{gop}} = \arg\min_{\mathbf{w}} J(\mathbf{w}) + \frac{1}{N}\mathbf{w}^\top\boldsymbol{\eta} + \frac{\Delta}{2}\|\mathbf{w}\|^2$
return $\mathbf{w}_{\text{gop}}$

Theorem 4.1. If ϕ is convex and doubly differentiable with $\phi'(z) \leq 1$ and $\phi''(z) \leq c\ \forall z$, then Algorithm 4 is (ϵ, δ)-differentially private.

The proof is based on the techniques used to analyze objective perturbation in [4] and on properties of the Gaussian and χ^2 distributions; details are provided in the appendix.[3]

Application to our setting. As noted above, the stochastic gradient descent based multiparty algorithm we have proposed can be viewed as performing Gaussian objective perturbation on the overall multiparty objective. Therefore if the third party runs the PSGD procedure as outlined, the privacy guarantee of Theorem 4.1 holds in this case as well. However, the problem here is that the third party may attempt to run the stochastic gradient descent procedure using the noisy gradients provided by one of the individual parties P_k only. In this case, the privacy guarantee to the party P_k is weaker (in particular, it is guaranteed only (ϵ, δ_k)-differential privacy, where $\delta_k > \delta$ and depends on N_k.[4] Instead, we can make use of the cryptographic scheme used

[3] The χ^2 distribution with parameter d corresponds to the density of the squared L_2 norm of a vector of d independent standard normal random variables.

[4] Alternatively one can ensure (ϵ, δ)-differential privacy for the individual parites by making each party add noise with a variance that depends on its data set. In such a case, the overall objective

by [7] at each iteration, where the third party directly receives the *sum* of the noisy gradients from all the parties in a cryptographically secure manner. Once this is done, the same (ϵ, δ)-differential privacy guarantee we had for GOP holds for all the parties in the multiparty setting as well:

Theorem 4.2. If on each iteration t the third party receives the *sum* of noisy gradients from all the parties in a cryptographically secure manner as described above, then Algorithm 1 (with σ^* chosen as described in Procedure 3) is (ϵ, δ)-differentially private.

The proof involves establishing equivalence of Algorithm 1 and 4. Details are in the appendix.

5 Generalization Analysis

In this section, we turn to the error analysis of our algorithm. Recall that all data points are assumed to be generated iid from a fixed but unknown distribution $\mathcal{Q}$ over $X \times \{\pm 1\}$. We would like to obtain high probability bounds (over the draw of $D \sim \mathcal{Q}^N$ and the randomization in the algorithm) on the expected error of the learned classifier $\mathbf{w}$ on future examples from $\mathcal{Q}$, e.g. as measured by the ϕ-loss, $\mathbf{E}_{(\mathbf{x},y)\sim\mathcal{Q}}[\phi(y\mathbf{w}^\top\mathbf{x})]$. We first have the following result, which gives a high probability bound on the excess regularized empirical risk of $\mathbf{w}_{\text{gop}}$, $J(\mathbf{w}_{\text{gop}})$, over that of the minimizer $\mathbf{w}^*$ of $J(\mathbf{w})$:

Theorem 5.1. If ϕ is such that $\phi'(z) \leq 1\ \forall z$, $\phi''(z) \leq c\ \forall z$, all the data instances $\mathbf{x}_i$ lie in a unit ball and $\Delta = 0$, then with probability at least $1 - \delta'$ (over the draw of $D \sim \mathcal{Q}^N$ and randomization in the algorithm), the excess empirical regularized risk of the perturbed classifier $\mathbf{w}_{\text{gop}}$ learned by Algorithm 4 over the minimizer $\mathbf{w}^*$ of $J(\mathbf{w})$ is bounded as

$$J(\mathbf{w}_{\text{gop}}) \leq J(\mathbf{w}^*) + \frac{(c+\lambda)}{2N^2\lambda^2}\widehat{T}$$

where $\widehat{T} := \widehat{T}(d, \tilde{\epsilon}, \delta, \delta')$ satisfies the equation

$$\mathbf{Pr}\left(U \leq \frac{\widehat{T}}{\sigma^{*2}}\right) = 1 - \delta'$$

where U is a χ^2 random variable with d degrees of freedom and $\sigma^* = \sigma^*(d, \tilde{\epsilon}, \delta)$ is as chosen in Procedure 3.

The proof makes use of techniques similar to those used in [4], together with properties of Gaussian and χ^2 distributions; exact details are provided in the appendix.

The above result is better understood using the following proposition which makes the dependence of the quantity $\widehat{T}$ on the parameters d, $\tilde{\epsilon}$, δ, and δ' explicit.

Proposition 5.2. The following bound holds for parameters as in Theorem 5.1 and Algorithm 4

$$\widehat{T} = O\left(\frac{d^2 \log(\frac{1}{\delta}) \log(\frac{1}{\delta'})}{\tilde{\epsilon}^2}\right)\ \forall\ \delta, \delta' \leq \frac{1}{e}$$

The proof involves bounding the tail of χ^2 distribution using an inequality of [9] and is given in the appendix.

The following theorem bounds the excess expected ϕ-loss of $\mathbf{w}_{\text{gop}}$ over that of $\mathbf{w}^*$.

Theorem 5.3. Let ϕ be convex and doubly differentiable with $\phi'(z) \leq 1$ and $\phi''(z) \leq c\ \forall z$. Then there exists a constant κ such that for any fixed weight vector $\mathbf{w}_0$ and any $\delta' > 0$, if

$$N > \kappa \max\left(\frac{\|\mathbf{w}_0\|^2 \log(\frac{1}{\delta'})}{\tau^2}, \frac{c\|\mathbf{w}_0\|^2}{\tau\epsilon}, \frac{\|\mathbf{w}_0\|\widehat{T}^{\frac{1}{2}}}{\tau}\right)$$

where $\widehat{T} := \widehat{T}(d, \tilde{\epsilon}, \delta, \delta')$ is as in Theorem 5.1, then with probability at least $1 - 2\delta'$ (over $D \sim \mathcal{Q}^N$ and randomization in the algorithm), the output $\mathbf{w}_{\text{gop}}$ of Algorithm 4 satisfies

$$\mathbf{E}_{(\mathbf{x},y)\sim\mathcal{Q}}\left[\phi(y\mathbf{w}_{\text{gop}}^\top\mathbf{x})\right] - \mathbf{E}_{(\mathbf{x},y)\sim\mathcal{Q}}\left[\phi(y\mathbf{w}_0^\top\mathbf{x})\right] \leq \tau\,.$$

Again, the proof is based on methods used in [4]; details are included in the appendix.

Comparison with bounds of [7] and [4]: It is worth noting that the bounds presented above do not depend on the number of parties K or the relative sizes of the individual parties' data sets. In contrast, the bound in Theorem 4.3 of [7] which bounds the same quantity as in Theorem 5.1 contains three terms: while one term is independent of K and has the same dependence on $\tilde{\epsilon}, \delta'$, and d (assuming the privacy parameter δ is equal to the confidence parameter δ'; with an extra $\log(d)$ factor due to the different tail bound on the noise), there are two extra terms which do depend on the number of parties K. Thus in this sense, our bounds are superior to the bounds of [7]. The bound obtained in Theorem 5.3 (with $\delta = \delta'$, and using Proposition 5.2) is very similar to that obtained for (single-party) objective perturbation in Theorem 4 of [4] (again, their bound has an extra $\log(d)$ factor due to the different noise bound).

Note that while the above bounds are proved for $\mathbf{w}_{\text{gop}}$, the output of Algorithm 4, we also expect them to hold true for $\mathbf{w}_{\text{psgd}}$, the output of Algorithm 1. This is because theoretically both the algorithms minimize the same objective. In practice, this depends on the efficacy of the stochastic gradient descent procedure. The experiments in the following section confirm that the difference is not significant.

6 Experimental Results

In this section we present experimental results that compare the PSGD and GOP algorithms with the local aggregation method of Pathak et al [7], using both simulated and real-world data sets focusing on the effect of two parameters: the number of parties K, and the relative sizes of the individual data sets owned by different parties.

1. Simulated data. We generated 1000 training and 1000 test examples in $\mathbb{R}^{10} \times \{\pm 1\}$ as follows: each instance $\mathbf{x}$

can also be guaranteed (ϵ, δ)-differential privacy but the generalization bounds will depend on the number of parties K and the size of the smallest data set as in [7]

was drawn uniformly at random from the interior of a 10-dimensional unit hypersphere; a random weight vector $\mathbf{w} \in \mathbb{R}^{10}$ was then chosen as the true classification vector, and the instances were labeled according to $\text{sign}(\mathbf{w} \cdot \mathbf{x})$. This process was repeated 5 times to obtain 5 random data sets (each containing 1000 train and 1000 test examples); the results reported below are averaged over these 5 data sets.

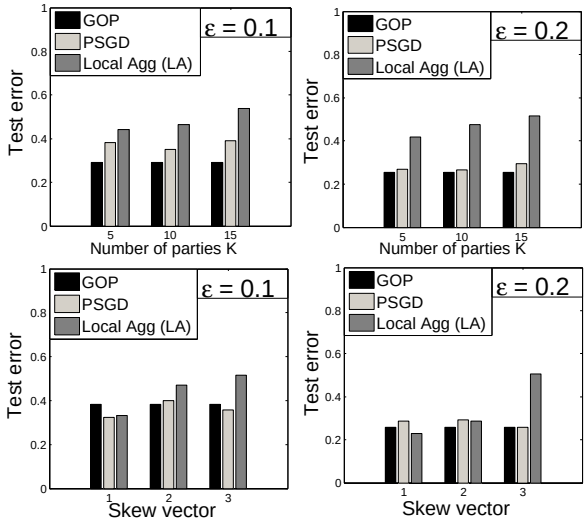

Figure 1: Effect of number of parties K (top) and relative sizes of individual data sets (bottom) on the PSGD and GOP algorithms and on the local aggregation (LA) method of Pathak et al [7] on the simulated data set. The x-axis for the bottom figure represents different skew vectors (1 = [0.2, 0.2, 0.2, 0.2, 0.2], 2 = [0.1, 0.2, 0.2, 0.25, 0.25] and 3 = [0.01, 0.29, 0.2, 0.25, 0.25]). Results with PSGD and GOP are with $\delta = 0.05$.

1a. Effect of number of parties K. We considered the effect of distributing the training set among K parties for different values of K, namely, $K = 5, 10$, and 15. Note that the classifier learned by the GOP algorithm does not depend on the number of parties; the purpose of this experiment was to evaluate its performance relative to the PSGD algorithm and the algorithm of [7] for different K. For each value of K, a *data split* vector was chosen uniformly at random from a $(K-1)$-dimensional simplex; this was used to decide the fraction of training data points to allocate to each party. The GOP and PSGD algorithms were run with $\delta = 0.05$; the results (Figure 1 (top)) show the average test error for $\epsilon = 0.1$ (left) and $\epsilon = 0.2$ (right). As can be seen, (a) increasing the number of parties causes a degradation in the performance of the algorithm of [7]; and (b) the performance of PSGD is similar to that of GOP and performs better than the algorithm of [7] even when only a small number of parties are involved.

1b. Effect of relative sizes of individual data sets. In this experiment we fixed the number of parties to $K = 5$ and considered the effect of different size splits of the data among the 5 parties. Three data split vectors were considered: [0.2, 0.2, 0.2, 0.2, 0.2],[0.1, 0.2, 0.2, 0.25, 0.25] and [0.01, 0.29, 0.2, 0.25, 0.25]. Each vector indicates the relative fraction of data owned by different parties. Figure 1 (bottom) shows the result for $\epsilon = 0.1$ (left) and $\epsilon = 0.2$ (right). Again, one can observe that (a) the algorithm of [7] depends heavily on the size of the smallest data set, with performance degrading as the smallest data set size reduces; and (b) the performance of PSGD and GOP are similar to each other and are not affected by the skew of the data sets.

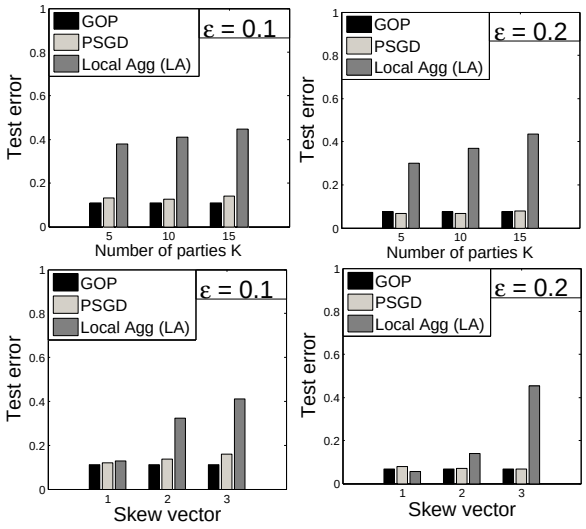

Figure 2: Effect of number of parties K (top) and relative sizes of individual data sets (bottom) on the PSGD and GOP algorithms and on the local aggregation (LA) method of Pathak et al [7] on the Wisconsin breast cancer data set. The x-axis for the bottom figure represents different skew vectors (1 = [0.2 0.2 0.2 0.2 0.2], 2 = [0.1, 0.2, 0.2, 0.25, 0.25] and 3 = [0.01, 0.29, 0.2, 0.25, 0.25]). Results with PSGD and GOP are with $\delta = 0.05$.

2. Real-world data. We compared the above algorithms on the Wisconsin breast cancer data set (569 instances, 30 dimensions) [11]. The data set was divided into 5 parts and 5 different (training, validation) splits were generated from it where the validation set in each split contained a different part of the data set. Results in Figure 2 are averaged over these 5 splits. The observations made in the simulated data set case hold true for this data set as well.

7 Conclusion

We considered the notion of differential privacy in multiparty settings and proposed a new stochastic gradient descent based algorithm that directly approximates the overall multiparty objective. We argued that our algorithm can be viewed as performing Gaussian objective perturbation on the overall objective and showed that it achieves (ϵ, δ)-differential privacy. We obtained bounds on the excess error of the learned classifier that are independent of the number of parties or relative sizes of individual parties' data sets. Experimental results show that our method has comparable accuracy with the local aggregation method of [7] in general and outperforms it when the minimum size of the dataset among all the parties is relatively small or when the number of parties is large.

Acknowledgements: We thank Harish Guruprasad and Manas Pathak for discussions related to this work. We would also like to thank the anonymous reviewers for their useful suggestions. This research was supported in part by a Ramanujan Fellowship to SA from the Department of Science and Technology, Government of India. AR's travel to the conference is supported in part by a Microsoft Research India Travel Grant.

References

[1] C. Dwork. Differential privacy. In *M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, ICALP (2), volume 4052 of Lecture Notes in Computer Science, pages 112. Springer, 2006*.

[2] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference, 2006*.

[3] K. Chaudhuri, C. Monteleoni: Privacy-preserving logistic regression. In *Neural Information Processing Systems, 2008*.

[4] K. Chaudhuri, C. Monteleoni, and A. Sarwate. Differentially private empirical risk minimization. In *Journal of Machine Learning Research, 12:1069–1109, 2011*.

[5] B. I. P. Rubinstein, P. L. Bartlett, L. Huang, and N. Taft. Learning in a large function space: Privacy-preserving mechanisms for SVM learning. In *http://arxiv.org/abs/0911.5708, 2009*.

[6] K. Sridharan, N. Srebro, and S. Shalev-Shwartz. Fast rates for regularized objectives. In *Neural Information Processing Systems, 2008*.

[7] M. Pathak, S. Rane, and B. Raj. Multiparty differential privacy via aggregation of locally trained classifiers. In *Neural Information Processing Systems, 2010*.

[8] H. Robbins,S. Monro. A stochastic approximation method. In *Annals of Mathematical Statistics, 22, 400–407, 1951*.

[9] B. Laurent and P. Massart. Adaptive estimation of a quadratic functional by model selection. In *Annals of Mathematical Statistics*, 28(5):1302–1338, 2000.

[10] M. Gotz, A. Machanavajjhala, G. Wang, X. Xiao, and J. Gehrke. Privacy in search logs. In *Computing Research Repository*, vol. abs/0904.0682, 2009.

[11] A. Frank, A. Asuncion, UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.