Privacy-Secure Stochastic Gradient Descent

Reid M. Bixler
Computer Science Dept.
Virginia Tech
Blacksburg, VA 24060

Abstract

The modern Stochastic Gradient Descent (SGD) algorithm is an optimization method used to minimize an objective function. Stochastic Gradient Descent typically gets to this minimization by iteratively passing over all the training examples until the algorithm will eventually converge. However, in the context when said training examples are based off of potentially personally identifiable information (PII), there is a risk that data may end up being leaked through the algorithm. For instances when one might like to learn on such data, I propose a modification to the Stochastic Gradient Descent method that is privacy-secure in order to minimize the amount of data leakage while still being able to achieve convergence close enough to the original algorithm.

1 INTRODUCTION

The general descent algorithm is a fairly straightforward method that relies on the basis of taking the gradient of the objective function such that each update should be less than the previous (Boyd and Vandenberghe, 2004) such that:

$$f\left(x^{(k+1)}\right) \leq f\left(x^{(k)}\right) \tag{1}$$

The general descent method's update step relies on the fact that the function f is convex, such that it is possible to consistently find the gradient of the function without the possibility of having the updated step be larger than the previous. Specifically, the basic update step can be defined as:

$$x^{(k+1)} = x^{(k)} + t^{(k)}\Delta x^{(k)}, \tag{2}$$

where $t^{(k)}$ is a scalar value called the *step size* or *step length* at iteration k and $\Delta x^{(k)}$ can be seen as the *step* or *search direction* (Boyd and Vandenberghe, 2004). Generally, the step size is often a user-chosen value that will determine how far down the gradient the method will go with each update step. If the step size is too small, then the gradient descent algorithm will often take too long to converge towards the minimum while if the step size is too large then it may not converge due to overcompensation.

The general descent method can then be defined as 3 basic steps:

1. Determine the descent direction Δx
2. *Line search*. Choose a step size $t > 0$
3. *Update*. $x := x + t\Delta x$

With basic intuition on the convexity of our objective function, we can therefore presume that a good search direction will be the case when $\Delta x = -\nabla f(x)$. This widely used instance of the general descent algorithm is otherwise known as the *gradient algorithm* or the *gradient descent method* (Boyd and Vandenberghe, 2004). In this case, we modify our steps of the algorithm to update our search direction to be this gradient of the objective function:

1. $\Delta x := -\nabla f(x)$
2. *Line search*. Choose step size t via exact or backtracking line search.
3. *Update*. $x := x + t\Delta x$

In an alternative case of the gradient descent method, it is possible to stochastically update our algorithm by *incrementally* updating the search direction based off of the gradient of a single point in the given training examples rather than calculating the gradient of the whole dataset.

The reasoning behind this is that it may be possible to converge quicker by relying on the idea that each individual data point in the example set would be relatively similar to the whole dataset. This method would then be known as *stochastic gradient descent* (SGD) (Fletcher and Powell, 1963). A potentially better modification to this algorithm would be to 'mini-batch' a few training examples together to get a better gradient step (Saad, 1998). Regardless, by modifying the gradient descent method to stochastically update the search direction based off of a subset of the values tends to result in significantly fewer update steps required to achieve convergence towards a global minima (Bottou, 2010).

The goal of this work is then to modify this stochastic gradient descent method with the intention of maintaining the privacy of the information available within the training examples, as the previous method must look at individual data points (or batches of these points) to calculate the gradient of said points.

1.1 RELATED WORK

There have been a few instances of trying to design a privacy-secure algorithm for gradient descent and general optimization methods. There is a version of privacy-preserved stochastic gradient descent, but this approach relies on modifying the algorithm of SGD by sharing the weight parameters instead of the gradients, as well as using symmetric encryption to protect the weight parameters against an honest-but-curious server used as a common place for storage (Song et al., 2013).

Similarly, by introducing differentially private updates to the stochastic gradient descent it would be possible to realize privacy-preserving logistic regression in a cryptographic notion (Wu et al., 2013). This approach focuses on preserving the privacy of logistic regression between two parties by implementing a Pailliar cryptosystem which is different than modifying the update steps of the stochastic gradient descent method itself to preserve privacy.

Another example of a similar vector of research is a design of differentially private stochastic gradient descent for multiplarty classification which is developed for privacy preserving machine learning algorithms in a distributed multiplarty setting (Rajkumar and Agarwal, 2012). This research focuses on a similar approach as that of this paper, but instead focuses on a multiparty approach reliant on not actually modifying the algorithm itself but rather perturb the gradients themselves.

Finally, yet another similar work utilizes a two-party protocol for differentially private gradient descent (Wan et al., 2007) instead formulates gradient descent as a composition of functions and utilizes data partitions to guarantee the security of the data.

Looking at these previous research vectors, one can see that instead of modifying the most basic update step they have instead decided to implement some sort of overhead to SGD in order to allow for privacy. Instead, the focus of this research paper is to randomly fuzz/perturb each individual data point before the algorithm uses it for learning in an attempt to preserve privacy of the data while also still resulting in a generally good approximation of the model.

2 BACKGROUND

The dataset being used by stochastic gradient descent can be simply defined as a set of training examples Z where each individual example z is itself a pair of values (x, y) such that x is the input and y is some scalar output. The gradient descent algorithm itself must have some parameterized weight vector w such that we can define a function $f_w(x)$. We must define a loss function L where $L(\tilde{y}, y)$ calculates the cost differential between the guessed output $\tilde{y}$ and the actual output y. Our goal therefore is to find a function f that satisfies the case where we minimize the loss function $L(f_w(x), y)$. This itself can be simplified to be some function E that can be defined as $E(z_i, w_i) = L(f_w(x_i), y_i)$, where this calculates the error given example z_i and weight vector w_i. We can then modify our original gradient descent method to *estimate* the gradient based off of a randomly selected example z_i (or batch of examples z_i to z_j):

$$w^{(k+1)} = w^{(k)} - t^{(k)} \nabla_w E(z^{(k)}, w^{(k)}), \qquad (3)$$

where $t^{(k)}$ is the learning rate evaluated for iteration k that should be sufficiently small enough calculated via exact or backtracking line search (assuming a convex space). In this case, we are also taking the gradient of our function E based off of the single example z_i in order to incrementally move towards a minima.

In the case of the batch version of stochastic gradient descent, we must define a batch B_k which is itself a subset of the total training examples Z at iteration k. The algorithm can then be modified like so:

$$w^{(k+1)} = w^{(k)} - t^{(k)} \frac{1}{b} \sum_{z_i \in B_k} \nabla_w E(z^{(k)}, w^{(k)}), \quad (4)$$

where b is the size of the subset B_k such that the sum ends up being normalized.

The noticeable problem with this formulation, in regards to privacy, is simply the fact that the examples z_i must be itself viewed by the stochastic gradient descent algorithm in order to evaluate the gradient. This can result in a serious data leakage of personally identifiable information assuming that the example z_i were to contain such information.

3 FORMULATION

The goal of our method must be then to ensure that the stochastic gradient descent algorithm cannot calculate the gradients on the *original* examples z. Therefore we must take 2 things into account. Firstly, we must ensure that our initial set of weight vectors w_0 must be independent of our set of examples Z. This is to ensure that any sensitive data could not be potentially leaked because of where the algorithm itself is initialized to. A valid initialization of this set of weight vectors would be to initialize all of the weights within w_0 to be randomly distributed between $[-1, 1]$. Secondly, a the key to this modification of SGD, is to randomly fuzz the examples z before they are being used within the algorithm such that the privacy remains.

A simplistic approach to this formulation would be to simply add some noise vector N_k at each iteration k to guarantee that the examples z never leak data. This approach could be seen as:

$$w^{(k+1)} = w^{(k)} - t^{(k)} \nabla_w E(z^{(k)}, w^{(k)}) + N^{(k)}, \quad (5)$$

where $N^{(k)}$ is drawn from some basic distribution that ensures an even amount of noise is being added such that the eventual convergence is relatively within the actual minima (Song et al., 2013).

However, this would only offer a slight amount of differentiable privacy at the *end* of every update step, which could prove useful in some use cases, but still has potential to leak data within the method itself. The end goal of having a privacy-secure modification to SGD relies on the fact that the example z itself must not be visible by the update step when calculating the loss using the function E. The reason I suggest this is that were a malicious entity to have visibility of the innards of the SGD method, the entity must not be able to see the example z be used as a singular parameter within the method.

Therefore, we must formulate an alternative such that our random noise vector N_k modifies the values of z *before* the update step can potentially leak the data. As such, the proposed alternative is defining a *noise function* that can be seen as the following:

$$N(x) = x + R * n, \quad (6)$$

where the value R can be considered to be a pseudo-random number generated from the set $\{-1, 1\}$ in order to ensure an even distribution. The value n is simply a scalar value drawn independently from the density function:

$$\rho(x) \propto e^{-\alpha^2/2}, \text{where } \alpha > 0 \quad (7)$$

In this case, we can see that our value α is similar to that seen in Song et al., which can be defined as a *privacy parameter* discerning how much noise ends up being added to the given value x. Similarly, this privacy parameter can quantify how much privacy risk is being taken, where the closer that α is to 0 the more privacy-secure the modified SGD algorithm will be. For the case when x is a vector of dimension $\mathbb{R}^X$, then n will similarly be drawn from the same density function $\rho(x)$ for an equally sized vector of dimension X.

With this noise function defined we can therefore finally formulate the privacy-secure stochastic gradient descent method as:

$$w^{(k+1)} = w^{(k)} - t^{(k)} \nabla_w E(N(z^{(k)}), w^{(k)}), \quad (8)$$

For this case, we must ensure that whatever value calculates the result of $N(z^{(k)})$ is secure from any malicious entity that could leak or view the original values of z. In the case where we are doing batches of size b we must similarly normalize our resulting values such that:

$$w^{(k+1)} = w^{(k)} - t^{(k)} \frac{1}{b} \sum_{z_i \in B_k} \nabla_w E(\frac{1}{b} N(z^{(k)}), w^{(k)}), \quad (9)$$

From this, we can ensure that the stochastic gradient descent method should be calculating an extremely similar gradient to that of the actual example z while still maintaining the privacy of this example using the noise function.

4 EMPIRICAL EVALUATION

In order to maintain that the privacy-secure modification to SGD calculates generally the same resulting loss than that of SGD, both algorithms were implemented in a basic Python program that relied upon a synthetic dataset. In these cases, both algorithms were implemented using the Python package `numpy`. Note that due to the inherent functionality of the stochastic gradient descent method, it is not possible to have both methods randomly draw the same examples at every iteration k, however as long as both methods converge to a point when the loss is past a

certain threshold (in this case, $t = 10^{-5}$) within relatively the same number of iterations then the methods are within a tolerable distance of one another.

The generated synthetic dataset has a total number of samples $N = 5000$ that drawn uniformly from a basic normal distribution. Both algorithms were run with batch size of 1 for sake of not having to implement the batched version of both SGD algorithms. As stated previously, both algorithms were run until the calculated gradient was below the tolerable threshold of 10^{-5}.

From these rudimentary experiments, it was seen that the privacy-secure modification to SGD resulted in the same convergence within a tolerable amount of extra iterations (5%), with the original SGD algorithm requiring a total of approximately 3000 iterations until convergence. When looking at the output of each epoch between the two algorithms, it is noticeable that while both loss functions are resulting in not the *exact* same values, they are within an average of 2.4% of one another. These disparities can be more likely attributed to the fact that stochastic gradient descent must randomly draw each sample 1 at a time, which could potentially be different than the other method's at the same iteration. Regardless, these results show that the privacy-secure modification to SGD seems to result in a similar convergence to that of SGD.

Analytically looking at this privacy-secure SGD, it becomes apparent that all the modification is doing is slightly fuzzing the example z within a value that tends towards 0 depending on the given value of α. When running with a value of α much larger than 0, then it becomes apparent that the privacy-secure SGD requires more than the tolerable extra iterations (5%) but still tends to calculate generally to the same minima.

5 CONCLUSION

In this paper, a privacy-secure modification to stochastic gradient descent was formulated that should result in essentially similar results that would be obtained by the non-modified SGD algorithm while still ensuring the privacy of the data within the given example set Z. This privacy-secure modification can be used for both the basic stochastic method that looks at each individual example z_i as well as allows for running on the mini-batch stochastic method that looks at examples z_i to z_j (all randomly drawn from the set Z).

This modified stochastic gradient descent algorithm should prove useful for cases when the user does not wish to have data leakage while still relying on the basic formula and steps of the normal SGD method with slight variation. The noise function itself can be substituted for similar noise functions which could prove to be an interesting research vector that was not analyzed within this work.

Similarly, it would be interesting to see the effects of the α value for instances when the number of iterations well past the tolerable 5% margin and how the sacrifice of iteration time would weigh against the benefits of the privacy provided. Another potential future work off of this work would be to look at alternative optimization algorithms and look for ways to ensure the privacy of the examples provided.

References

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Fletcher, R. and Powell, M. J. (1963). A rapidly convergent descent method for minimization. *The computer journal*, 6(2):163–168.

Rajkumar, A. and Agarwal, S. (2012). A differentially private stochastic gradient descent algorithm for multiparty classification. In *Artificial Intelligence and Statistics*, pages 933–941.

Saad, D. (1998). Online algorithms and stochastic approximations. *Online Learning*, 5.

Song, S., Chaudhuri, K., and Sarwate, A. D. (2013). Stochastic gradient descent with differentially private updates. In *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, pages 245–248. IEEE.

Wan, L., Ng, W. K., Han, S., and Lee, V. (2007). Privacy-preservation for gradient descent methods. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 775–783. ACM.

Wu, S., Teruya, T., Kawamoto, J., Sakuma, J., and Kikuchi, H. (2013). Privacy-preservation for stochastic gradient descent application to secure logistic regression. In *27th Annual Conference of the Japanese Society for Artificial Intelligence, 3L1-OS-06a-3*.