Erasmus University Rotterdam

Bachelor Thesis

Stochastic Gradient Descent with Differential Private Updates

Author:
Wouter Wessels 408055

Supervisor:
Ilker Birbil

July 5, 2018

Abstract

In the field of Machine Learning, Stochastic Gradient Descent is one of the most popular methods in use. That is because it is computational easy and scalable. However, one of the things it does not take into account is differential privacy, that is a term for the privacy loss when an individual's private information is being used for the creation of a data product. In this paper we will look what happens when we include differential privacy into the SGD algorithm, with Song et al. (2013) as example. We will compare the differential private algorithms with the normal ones and we will look at the effect of the batch sizes on the obtained objective values. Moreover we will look at the differential private algorithm with a different loss function. At the end it turns out that differential private algorithms are very volatile compared to the normal algorithms but when the batch size becomes larger, there is not much difference anymore in their performance.

1 Introduction

Data nowadays is becoming more available and easier to acquire in digitized formats. Many companies use it as a marketing tool to promote their products on the behavior of the customers. In many cases data contains private information, for example medical or financial records. Therefore, taking into account privacy is becoming more important and designing algorithms including a user's privacy is a popular research topic. While guaranteeing and improving this privacy, one also wants to maximize the information retrieval. Information retrieval is the science for searching information in a document, searching for documents themselves and for databases in texts, images and sounds. The privacy loss of an individual due to their private information being used by creating a data product is called differential privacy. Differential privacy aims to maximize information retrieval and minimize the privacy loss. The problem of this paper is to make an algorithm that includes differential privacy. This is of course not easy as there is a trade-off between utility and privacy. The higher the utility of the algorithm, the lower the privacy (and the higher the privacy loss). Moreover differential privacy may have a bad influence on the Mean Squared Error of certain estimators. This is because a larger sample size is needed because of the privacy constraints.

The research is relevant for both scientific and practical applications. There is still much research going on about the topic. Also in practice many big companies have adopted differential privacy. Examples of this are the US Census Bureau for showing commutation patterns (see [12]) and Google for sharing historical traffic statistics (see [13]). The corresponding research question can be formulated as: can we include differential privacy into algorithms and how does it affect the performance of the algorithm?

In this paper we will look at a specific method, which is really popular in this framework: Stochastic Gradient Descent (SGD). SGD algorithms have gained significant attention recently because they are relatively simple and guarantee the same asymptotic properties as more computationally difficult algorithms.

In our research we will look at the so-called mini-batch updates of the SGD algorithm with and without differential privacy as in Song et al. (2013). To extent the research, we will apply the method again on different datasets and use a different objective function instead of the normal objective function for linear classification. The research is necessary as the datasets used in Song et al. (2013) do not seem to be appropriate to test the hypothesis. Moreover, using a different objective function shows that mini-batch updates work for all kind of classification problems.

The paper is organized as follows: in the next section, we will give an overview of the existing literature about the topic. In section 3, we will give explanation about the idea of linear classification, a term in the field of Machine Learning, where we are going to work with. Furthermore we will explain the application of SGD on the problem and the way how we are going to include differential privacy, which is based on the way Song et al. (2013) did that. In section 4, we will describe the datasets we are going to use, in section 5 we will show and interpret the results and finally in section 6 we will give conclusions, mention the limitations of the research and give suggestions for future research.

2 Literature Review

The concept of differential privacy is first introduced by Dwork et al. (2006). Their work is based on Nissim and Dinur (2003). They have shown that it is impossible to publish information from a private statistical database without revealing some amount of private information. Because of that they have realized that new techniques should be developed to decrease the privacy risk. Dwork et al. (2006) have made up this technique with differential privacy. They have tried to

protect the privacy of an individual by adding a random noise generated from a specific distribution. The extent of privacy of the specific algorithm is determined by the privacy parameter α. The higher α is, the lower the extent of differential privacy. We will explain this idea more thoroughly in the Methodology section. There has been research about including differential privacy in the perceptron algorithm of Rosenblatt, one of the oldest algorithms in the field of Machine Learning. This research was done by Blum et al. (2005). Moreover they have shown that a huge amount of privacy can be obtained by using a small amount of noise.

There is also literature available about SGD, applied on classification problems. Rakhlin et al. (2011) have looked at the performance of SGD on strongly convex optimization problems, where we are going to work with now. They have shown that for smooth problems SGD attains the optimal convergence rate. Cotter et al. (2011) have first introduced the idea of grouping the data into mini-batches to improve the performance of SGD in a finite sample. Song et al. (2013) have extended this research by including differential privacy in the mini-batch algorithm. They have compared the performance of normal SGD and differential private SGD with and without mini-batches. At the end they have concluded that there is almost no difference in the performance when the mini-batch updates for SGD are used.

3 Methodology

In this section, we will discuss the methodology of the paper. It consists of three subsections. First, we will discuss the linear classification problem. Second, we will look at the Stochastic Gradient Descent Method, with and without mini-batch updates. Finally we will discuss differential privacy for the algorithm.

3.1 Linear classification

As in Song et al. (2013), we will describe the problem as a linear classification problem. Linear classification is a wildly used method in the field of Machine Learning. The definition is to use an object characteristic to identify in which class it belongs to. For example, an e-mail can be spam or non-spam. More generally, it means the data is grouped into $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$. x_i is called the training dataset and every observation of this training dataset is labeled by y_i. Each observation x_i contains d features. To make a prediction of which class x_i belongs to, we need a vector of weights. This vector gives weight to every feature and predicts whether x_i is above or under the hyperplane, which separates the 1 and -1 points. Because of this property, the weight vector is the normal vector to the hyperplane. The goal of linear classification is to find the normal vector which divides the -1 (spam) points and 1 (non-spam) points with the biggest margin (distance). A popular method for finding this vector is solving the following problem:

$$w* = \mathbf{argmin} \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} l(w, x_i, y_i) \tag{1}$$

This method is called a regularized convex minimization problem. In this formula, w is the normal vector to the hyperplane. The term $\|w\|^2$ is called the regularization function and prevents the weights for getting too large. The coefficient λ determines the importance of the regularization function. l is the convex loss function. It measures the difference between the prediction and the true value of the classifier y_i. In other words: it measures our unhappiness with the results. The higher the value of the loss function, the more unhappy we are with our prediction. There are many different variations for the loss function. The most popular one is probably the Hinge loss function, which is used for Support Vector Machines. However, in our research this function is

not appropriate since it is not differentiable. Therefore we will look at two other loss functions. The first one is called the Logistic loss function, which is defined as follows:

$$l = \log(1 + e^{-yw^T x}) \tag{2}$$

The Logistic loss function leads to a logistic regression. For the other loss function, we first have to define the so-called softmax classifier. The softmax classifier transforms the weights into probabilities. That means, the output of $w^T x_i$ will be the probability for being labeled with 1 or -1. This is done in the following way:

$$\text{Softmax} = \frac{e^{w^T x_i}}{1 + e^{w^T x_i}} \tag{3}$$

The corresponding loss function of the softmax classifier is the Cross Entropy loss function. This function is defined as follows:

$$l = -t \times \log(\text{softmax}) - (1 - t) \times \log(1 - \text{softmax}) \tag{4}$$

In this case t is equal to $\frac{1+y}{2}$. As we can see here, the linear classifier y is transformed into a $\{0, 1\}$ variable. So, in case $y_i = -1$, $t = 0$ and in case $y_i = 1$, $t = 1$. In this way the loss will be high if the probability of y_i being a certain value (for example 1) is low, while the real value of y_i is in fact this certain value (so 1 in the example).

3.2 Stochastic Gradient Descent

To find w^* in (1) several algorithms can be used. One of the most popular used methods is Stochastic Gradient Descent (SGD). SGD is an iterative algorithm and iterations are updated in the following way:

$$w_{t+1} = w_t - \eta_t(\lambda w_t + \nabla l(w_t, x_i, y_i)) \tag{5}$$

The algorithm has to start with an initialization point, *w0*. The idea is that you move from this initial point to a (local) optimum by updating the iterate in the direction of the gradient. The closer the algorithm approaches the optimum, the smaller the update will be. The parameter η_t is called the step size or learning rate. The learning rate controls to what extent we adjust the weights. It is important that the learning rate is not too large or too small. In case the learning rate is too large, the algorithm can jump over (local) optima. In case the rate is too small, the algorithm converges very slow and it will take a long time before we reach the (local) optimum.

To execute SGD, the gradient of the loss function has to be computed. The gradient can be composed by using a single point, (x_i, y_i). It is also possible to make a batch of data points of size b and base the gradient on that particular set. The formula of the update at step t is then as follows:

$$w_{t+1} = w_t - \eta_t(\lambda w_t + \frac{1}{b} \sum_{x_i, y_i \in B_t} \nabla l(w_t, x_i, y_i)) \tag{6}$$

These mini-batches of size b make sure that the variance is lower compared to SGD with the gradient based on a single point.

3.3 Differential Privacy

To make the algorithm α-differentially private, we have to take the following into account, as mentioned by Dwork et al. (2006); let α be the privacy parameter, a real positive number and $\mathbf{A}$ a randomized algorithm that takes datasets $\mathbf{D}$ as input. Furthermore, let S be a subset of the

image of $\mathbf{A}$, where the image is a subset of the algorithm's codomain. For differential privacy the following has to hold:

$$\Pr(\mathbf{A}(D_1) \in S) \leq e^{\alpha} \times \Pr(\mathbf{A}(D_2) \in S) \tag{7}$$

Datasets D_1 and D_2 are the same except for one element, so except for the data of one individual. As we can see, a lower privacy parameter α guarantees higher privacy.

To add differential privacy in the SGD algorithm, Song et al. (2013) uses the following equation:

$$w_{t+1} = w_t - \eta_t(\lambda w_t + \frac{1}{b} \sum_{x_i, y_i \in B_t} \nabla l(w_t, x_i, y_i) + \frac{1}{b} Z_t) \tag{8}$$

Here Z_t is a random noise vector drawn independently from the Laplace density function:

$$\rho(z) \propto e^{-(\alpha/2)\|z\|} \tag{9}$$

More thoroughly, the values z are sampled in the following way, according to Wu et al. (2017); first we have to sample a vector $\mathbf{v}$ and project it to the unit ball. Second we have to draw a magnitude l from the Gamma distribution $\Gamma(d, 2/\alpha)$. Finally we can obtain the values z by l$\mathbf{v}$. In the paper, we will use a value of 1 for the privacy parameter α.

4 Data

Song et al. (2013) uses both a synthetic and two real datasets. The synthetic dataset contains $n = 10.000$ observations drawn from a uniform distribution. For the real data, the KDDCup99 and the MNIST datasets are being used. The first one is an intrusion detection dataset on network connections with size $n = 50.000$. The second dataset consists of 60.000 training examples and 10.000 test examples with images of handwritten digits from 0 to 9. Tavallaee et al. (2009) argues that the KDDCup99 dataset has some important shortcomings and therefore they propose a new dataset, called the NSL-KDD. We will also use this dataset to evaluate the SGD method with private differential updates. Finally we will use a dataset, called the advertisement dataset, designed by Kushmerick (1999). This dataset labels an observation as an advertisement or not an advertisement. In the following subsections, we will explain the data more thoroughly.

4.1 Synthetic dataset

For the synthetic dataset, we are gonna draw a random vector from the 5-dimensional sphere. This means that the dimension d is 5. To create the labels y_i, we first have to create a vector $\mathbf{a}$. To create y_i the following must hold:

$$y_i = \begin{cases} 1, & \text{if } \mathbf{a}^{\mathrm{T}}\ \mathrm{x_i} > 0\ ; \\ -1, & \text{if } \mathbf{a}^{\mathrm{T}}\ \mathrm{x_i} < 0\ . \end{cases} \tag{10}$$

We are free to choose the value of $\mathbf{a}$. It can be seen as the perfect hyperplane to divide the labels. After defining all the x_i and y_i, we apply SGD to see if we can draw this perfect hyperplane again.

4.2 KDDCup99 dataset

The KDDCup99 dataset is the most wildly used dataset when it comes to intrusion detection on network connections. The training dataset contains 4,900,000 connections vectors, which have 41 elements each. All the vectors are labeled as normal or attack. The attacks can have the following categories:

Denial of Service Attack is an attack where the attacker makes a memory resource too busy or too full to handle specific requests, or denies user access to a machine.

User to Root Attack is an attack where the attacker gains access to a normal user account, for example by obtaining passwords, and with that information gets into a specific system.

Remote to Local Attack is an attack where the attacker can send packets to a machine over a network. However, the attacker does not have an account on that particular machine but can gain local access.

Probing Attack is an attempt to gather information about a network.

The test data of the KDDCup99 comes from a different probability distribution as the training dataset. Moreover it contains several attack types which are not included in the training set. This is because intrusion experts believe that most attacks are variants of already known attacks. For this paper we will use a subsample of 50.000, randomly selected for the mini-batches, just like in Song et al. (2013).

4.2.1 Problems KDDCup99

Tavallaee et al. (2009) noted that there are some inherent problems in the KDDCup99 dataset. These problems are mainly caused because KDDCup99 is based on another dataset called DARPA98. The structural problems DARPA98 has, are still present in KDDCup99. McHugh (2000) already noted the following issues:

- The data is claimed to be similar to data observed from Air Force bases after several months of sampling. However, there is no validation of it. Neither analytical nor experimental.
- Traffic collectors, also known as packet sniffers, of DARPA98 such as TCPdump, which are also used in KDDCup99, are very likely to become overloaded. In that case they can drop packets in heavy data traffic. There was, however, no examination of that possibility.
- There is no exact definition of the attacks in DARPA98.

Moreover Tavallaee et al. (2009) encountered some statistical deficiencies themselves. In the first place they say that there are a huge number of redundant records, which causes a bias for frequent records. Secondly, they claim that the accuracy rate of several Machine Learning Methods is way too high. They measured that in the following way: each record has 21 classifiers (classifiers can be normal or malicious). The accuracy of each record is determined by how many of the classifiers are correctly predicted. In both the training and test set, all the 21 classifiers were correctly predicted for 97.97% and 86.64% respectively, for all the records. This says that KDDCup99 is not challenging enough for the different methods and therefore not appropriate.

4.3 NSL-KDD dataset

In Section 4.2 it was mentioned that the KDDCup99 copes with some deficiencies. Tavallaee et al. (2009) comes with an alternative dataset to solve these inefficiencies, called the NSL-KDD dataset. In the first place they removed all the redundant records from KDDCup99. This caused the biggest deficiency and is now gone. Secondly, they made the dataset more challenging for all

the different Machine Learning methods they mentioned in the following way: if in 90% of the records all 21 classifiers are correctly predicted, the other 10%, so the records where less than 21 classifiers are correctly predicted, will be included in the dataset. In this way both a new training set and a new test set are constructed. Obviously, this dataset is more challenging than the original KDDCup99.

4.4 MNIST dataset

The MNIST dataset was firstly introduced by LeCun et al. (1998). Since then it is wildly used in the field of Machine Learning. Basically, it is a dataset with images from the digits 0 to 9. The training set consists of 60,000 pictures and the test set of 10,000. It is actually a subset from the larger dataset NIST with the difference that the size of the digits are normalized and centered in a fixed size image. For their research Song et al. (2013) used the so called 1 vs all classification, which means that the digit 1 is considered as an attack and all the other digits as normal.

4.5 Advertisement dataset

The advertisement dataset was created by Nicholas Kushmerick in 1999. It is a type of dataset where the training set consists of 1560 features and 3279 observations. Every observation is labeled as an advertisement or not an advertisement.

The observations are actually images on an internet page. As many internet pages obtain income by showing advertisements on their websites, these pages are full of them. Not every internet user appreciates that and therefore they want to have the possibility to remove the images. This can be done by Machine Learning and the advertisement dataset is an appropriate way to train the computer to recognize whether an image is an advertisement or not.

5 Results

In this section we will discuss the main results. In the first subsection we will show the results of the replication part for the datasets used in Song et al. (2013). In the second subsection, we will show the results of the two other datasets and results for the Cross Entropy loss function.

5.1 Replication part

In the replication part, we used the same datasets and methods as in Song et al. (2013). This means that the results are shown for the KDDCup99 dataset, the MNIST dataset and a synthetic dataset. For every graph the objective value, so the value after putting w_t in (1), is drawn against the number of iterations t both for the differentially private SGD algorithm (always the blue line) and the normal SGD algorithm (always the red line). As in Song et al. (2013), we used a learning rate of $1/\sqrt{t}$. After that we checked the final objective value, so after n iterations, for batch sizes 1, 5, 10, 20 and 50. This was done for learning rates $1/\sqrt{t}$ and $10/\sqrt{T}$. As initial value *w0*, we used a vector of ones. For the KDDCup dataset we used a reduced dimension d of 9 and for the MNIST dataset of 15. The loss function used is always the Logistic loss function as in (2).

5.1.1 Results KDDCup

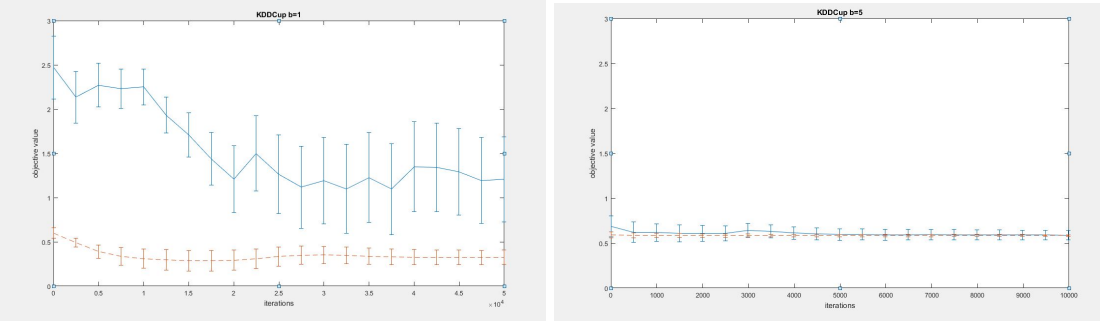

Figure 1: Results KDDCup99 dataset for batch sizes 1 and 5

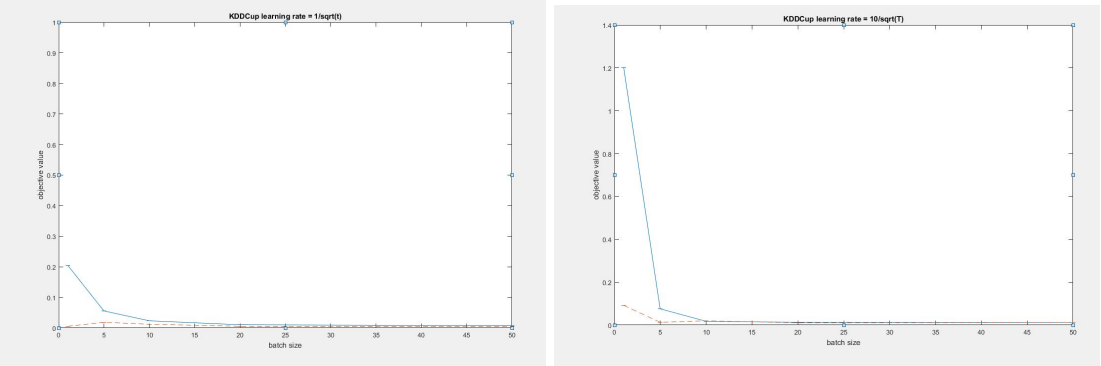

Figure 2: Final objective values for batch sizes 1, 5, 10, 20 and 50

5.1.2 Results MNIST

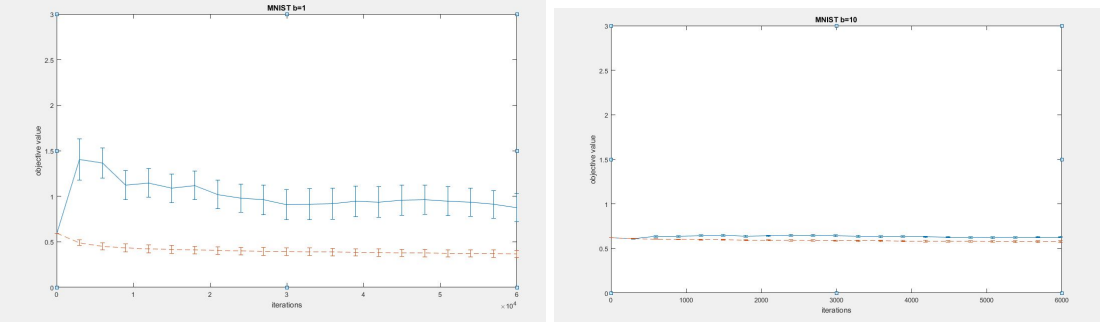

Figure 3: Results MNIST dataset for batch sizes 1 and 10

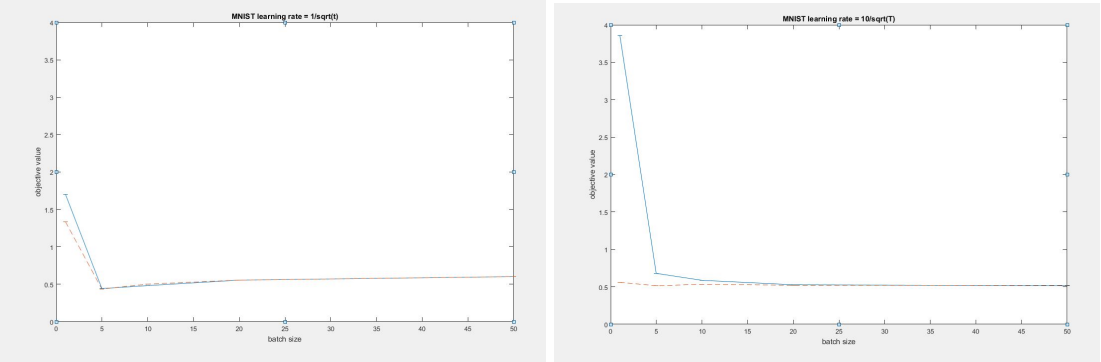

Figure 4: Final objective values for batch sizes 1, 5, 10, 20 and 50

5.1.3 Results Synthetic dataset

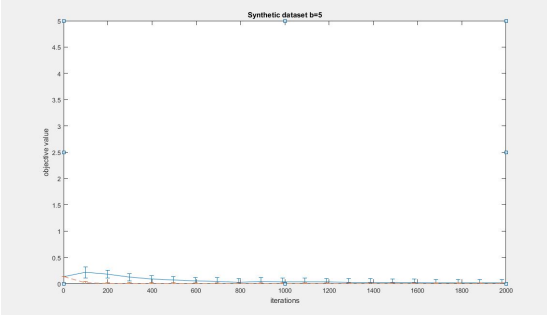

Figure 5: Results Synthetic dataset for batch size 5 and learning rate $1/\sqrt{t}$

5.1.4 Findings

What we can see here is that the batch size reduces the variance. In figures 1 and 5, there is a huge difference between the private and non-private algorithms with batch size 1. However, when the number of iterations increases the objective value comes closer to the non-private algorithm, both for the MNIST and for the KDDCup dataset. When we increase the batch size to 5 and 10 for the KDDCup and the MNIST dataset respectively, we see almost no difference anymore in the objective values and the variance is way lower than with a batch size of 1.

In figures 2 and 4, we can see that a larger batch size gives a smaller objective value but only if we compare a batch size of 1 with a larger batch size. Between a batch size of 5 and a higher batch size, there is almost no difference in the objective value. The same can be said about the learning rates used. Only in batch size 1, there is a huge difference in objective values as learning rate $10/\sqrt{T}$ gives very high values. But again, when we increase the batch size the big difference is gone and both learning rates give more or less the same objective values.

5.2 Extension part

For the extension part, I first used the same methods as in the replication part but with the different datasets NSLKDD and the advertisement dataset. Furthermore, I looked at the results for the Cross Entropy loss function as in (4) for the KDDCup and MNIST dataset. For the

NSLKDD, we used a reduced dimension of 9, just like for the KDDCup dataset and for the advertisement dataset we used a reduced dimension of 100.

5.2.1 Results NSLKDD

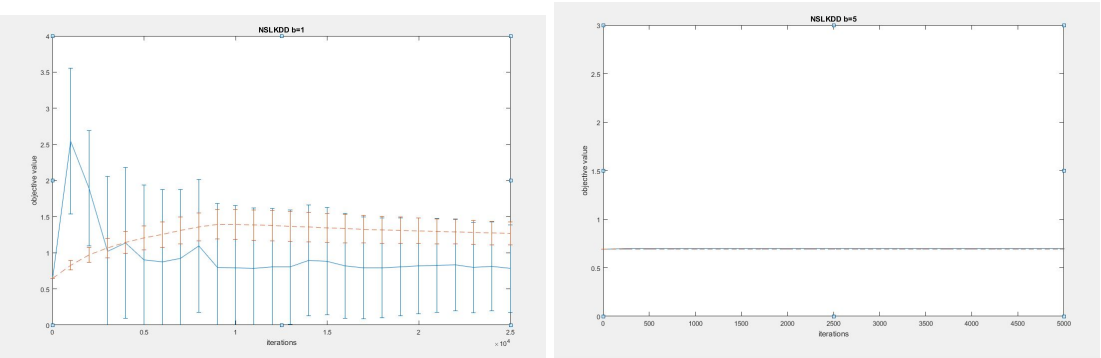

Figure 6: Results MNIST dataset for batch sizes 1 and 5

In figure 6 we can see that the variance of the non-private algorithm is very high all the time and because of that even performs better at some point than the non-private algorithm. Again when we increase the batch size to 5, almost no difference is found.

5.2.2 Results advertisement dataset

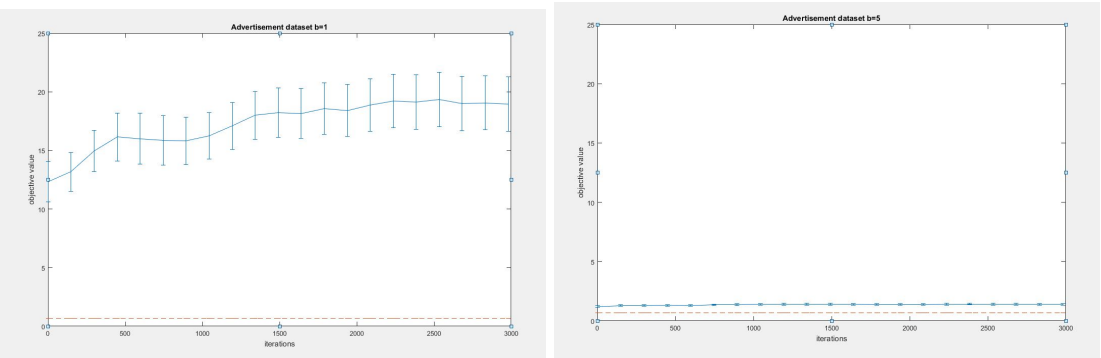

Figure 7: Results advertisement dataset for batch sizes 1 and 5

In figure 7 we can see that the non-private algorithm performs very bad, way worse than the private algorithm. It almost seems to be increasing when more iterations are being used, until some point. With a batch size of 5, the private algorithm still performs worse but the difference is not big anymore.

5.2.3 Results Cross Entropy loss function

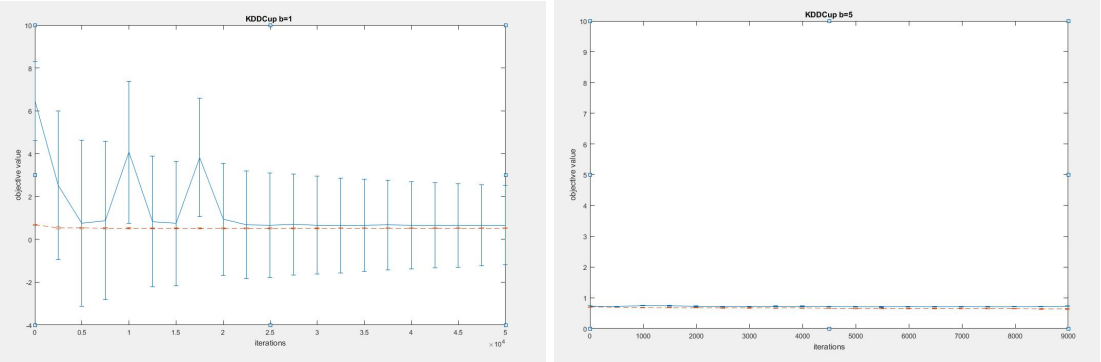

Figure 8: Results KDDCup99 dataset for batch sizes 1 and 5 with CE loss function

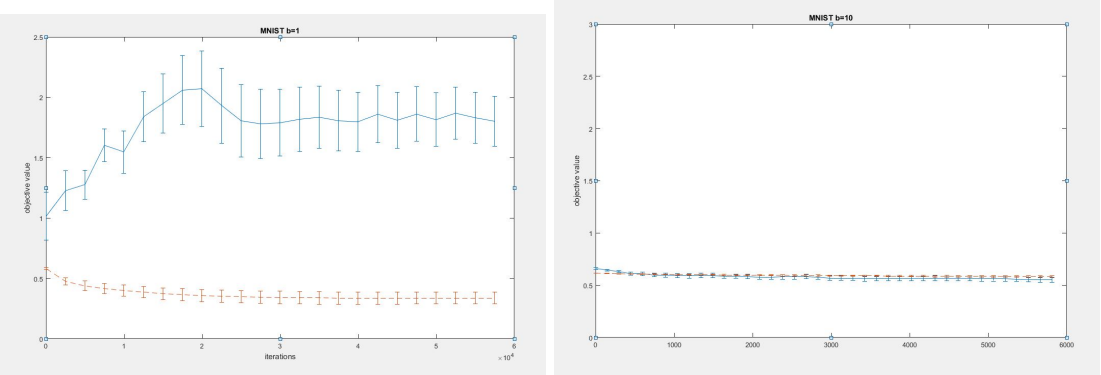

Figure 9: Results MNIST dataset for batch sizes 1 and 10 with CE loss function

In figure 8 we can see that the objective value for the KDDCup dataset is very volatile for batch size 1. However, at some point the value becomes the same as the non-private value. When the batch size is increased to 5, we can see that the difference is gone from the start.

In figure 9 we can see that the objective values are always way higher for batch size 1 but here again they become the same when batch size is increased to 5.

6 Conclusion

In this paper, we tried to find an answer on the research question: can we include differential privacy into information algorithms and how does it affect the performance of the algorithm? To find an answer on that question we made use of Machine Learning methods and applied the SGD iterative algorithm to obtain objective values. We compared the performance of normal (non-private) SGD with private SGD and also looked at the effect of batch sizes, just like in Song et al. (2013). As extensions, we used different datasets and a whole different objective function.

In the replication part, we obtained more or less the same results as in Song et al. (2013). What we have seen for every dataset, was that for a batch size of 1, the differential private

algorithm is very volatile and objective values are far above the values obtained with normal SGD. Especially for the KDDCup99 dataset, the standard deviations are really high. When we increase the batch size, it can be seen that the performance of the private SGD algorithm is way better. Moreover, there is almost no difference with the normal SGD objective values anymore. In figures 2 and 4 it is shown that a batch size larger than 1 has a positive effect on the objective value (i.e. objective value is lower) but from a batch size of 5, increasing does not have a significant effect anymore.

In the extension part, the same conclusions can be drawn. Just like in the datasets of the replication part, the performance of the algorithm improves with a larger batch size. Especially for the advertisement dataset, the objective value is really high (< 15) for a batch size of 1 and is way higher than the performance of the normal SGD algorithm. However, when we increase the batch size to 5, there is again almost no difference anymore. As we can see in figures 8 and 9 the same can be said about the use of a different loss function, the Cross entropy loss.

So, after analyzing the results from both the replication part and the extension part we can say that differential privacy can be included in information algorithms. For small batch sizes ($>$ 5), differential privacy affects the performance of the algorithms in a negative way and objective values are way above the values for normal SGD. However, if we increase the batch size to 5 the private algorithms are almost identical to the non-private ones. This is also the case if we use the Cross entropy loss function instead of the Logistic loss function.

Of course this paper also copes with some limitations. In the first place, the dimension reduction gives a bias to the results. As the features are chosen randomly, the objective values can differ a lot. This results in a very high standard deviation when the batch size is 1, especially for the KDDCup dataset. The same can be said about the observations, which are also selected randomly. For the NSLKDD and advertisement dataset, one could say that the number of observations is a bit too low so the results for these datasets are maybe not 100% trustworthy.

For future research it could be interesting to look at the effect of the dimension size on the objective values. Furthermore the research can be extended by looking at multivariate problems. So, instead of the classifier being drawn out of the set $\{-1, 1\}$, we can look at the performance of the SGD algorithm when the classifier can take more values. Here again, we can apply the problem on different loss functions.

References

[1] Song, S., Chaudhuri K., Sarwate A., 2013 "Stochastic Gradient Descent with Differentially Private Updates"

[2] Tavallaee M., Bagheri E., Lu W., Ghorbani A., 2009 "A Detailed Analysis of the KDDCup99 Data Set"

[3] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory, ACM Transactions on Information and System". *Security, vol. 3, no. 4, pp. 262–294, 2000*

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". *Proceedings of the IEEE, 86(11):2278–2324, 1998.*

[5] X. Wu, F. Li, A. Kumar, K. Chaudhuri, S. Jha, J. Naughton. "Bolt-on Differential Privacy for Scalable Stochastic Gradient Descent-based Analytics"

[6] N. Kushmerick. "Learning to remove internet advertisements"

[7] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis" *Theory of Cryptography, vol. 3876, March 2006, pp. 265–284.*

[8] I. Dinur and K. Nissim. "Revealing information while preserving privacy"

[9] A. Blum, C. Dwork, F. McSherry and K. Nissim, "Practical privacy: the SuLQ framework," *Proc. PODS. New York, NY, USA: ACM, 2005, pp. 128–138.*

[10] A. Rakhlin, O. Shamir and K. Sridharan. "Making gradient descent optimal for strongly convex stochastic optimization," *ArXiV, Tech. Rep. arXiv:1109.5647 [cs.LG], 2012.*

[11] A. Cotter, O. Shamir, N. Srebro, and K. Sridharan, "Better mini-batch algorithms via accelerated gradient methods," *Adv. NIPS 24, 2011, pp. 1647–1655.*

[12] Ashwin Machanavajjhala, Daniel Kifer, John M. Abowd, Johannes Gehrke, and Lars Vilhuber. "Privacy: Theory meets Practice on the Map". *In Proceedings of the 24th International Conference on Data Engineering, (ICDE) 2008.*

[13] A. Eland. "Tackling Urban Mobility with Technology by " *Google Policy Europe Blog, Nov 18, 2015.*