ERASMUS UNIVERSITY ROTTERDAM
Erasmus School of Economics

Master Thesis Econometrics and Management Science

Title of thesis: Solving Linear Programs to Train Support Vector Machines Under Differential Privacy

Name of student: Patrick van der Reijden
Student ID number: 430530

Name of Supervisor: Ş. İ. Birbil
Name of Second assessor: F. Frasincar

Date final version: 12-12-2019

Contents

Abstract

Differential privacy is a framework for sensitive data protection which can be used to obtain privately produced machine learning models. The challenge is to make the privacy-accuracy trade-off while still getting acceptable modelling results from private machine learning algorithms. Support Vector Machines are a class of machine learning methods for solving classification problems. Algorithms for training Support Vector Machines under differential privacy exist and have been shown to produce models which perform almost as well as non-private models. Unlike these existing private solutions, which utilize quadratic programming formulations, Support Vector Machines can be trained by solving accompanying linear programming formulations. Solving these privately is an interesting application for both machine learning and linear optimization fields. This research explores the application of privately solving linear programs for training Support Vector Machines by putting a theoretical framework to the test. Although the resulting algorithms do not outperform the state of the art in private machine learning modelling, this research paves a way towards solving different kinds of problems adequately while maintaining differential privacy.

Keywords: support vector machines; differential privacy; machine learning; linear programming; dense multiplicative weights; linear classification

1 Introduction

The privacy framework of differential privacy originates from the area of database querying. It provides a mathematical privacy guarantee for sensitive data which would be queried (Dwork, 2006). The framework has since been successfully applied to machine learning techniques. The resulting models hide sensitive information about the underlying data at the cost of model accuracy. Specifically, algorithms have been produced which implement Support Vector Machine based classification models privately, while sporting excellent out-of-sample prediction results (Chaudhuri et al., 2011). These algorithms are based on the quadratic programming formulation of Support Vector Machines. There are, however, other ways of solving Support Vector Machines (Mangasarian, 1998). One of which is using a linear programming formulation. In this research, the linear programming formulation is considered. Solving quadratic programs privately can be made relatively easy, because, unlike linear programs, solving them can be reduced to minimizing a single cost function using, for instance, the Hinge loss function. However, we want to solve linear programs differently. In differential privacy we use the notion of neighbouring data sets, which differ in only a single record. Since, here, a linear program is largely based on the sensitive

data, we get the notion of neighbouring linear programs. For Support Vector Machines, each record in the data will be represented by a restriction in the resulting linear program. Therefore, neighbouring data sets lead to neighbouring linear programs, which complicates the process of solving the entire system privately. Linear programs have, however, been solved privately in the literature using a type of boosting algorithm together with the exponential mechanism of differential privacy (Hsu et al., 2014). Therefore, this research applies said theory to solving Support Vector Machines using linear programming formulations and aims to find out whether this is an interesting alternative solution.

This work implements and tests the Dense Multiplicative Weights algorithm for solving linear programs privately (Hsu et al., 2014). Originally, this algorithm is accompanied by interesting theoretical properties and linear program-solving accuracy guarantees. In this research, because of the problem setting, the interest lies with model performance and not accurate linear program solving. Therefore, not only is this research presenting a working version of the Dense Multiplicative Weights algorithm for solving linear programs privately, but it is also providing an alternative to the quadratic programming approach of training Support Vector Machines. Hopefully, this will give insight to other people researching differential privacy applications for machine learning and linear programming. At first, this research was focused on privately solving linear programs for training Support Vector Machines by developing a differentially private Simplex Method algorithm. However, the Simplex Method algorithm constantly utilizes all sensitive data, complicating a per-iteration based privacy solution. Additionally, applying output perturbation to the algorithm does not yield interesting results with the output sensitivity calculated using properties of the Simplex Method algorithm. These complications make the work of Hsu et al. (2014) a very interesting alternative. They provide a neat theoretical solution to the problem we are trying to solve. Implementing this solution successfully in practice opens up possibilities for producing alternative ways of solving machine learning problems under differential privacy. Thus, we investigate the alternative of solving Support Vector Machines privately with a linear programming formulation by comparing the results with the state of the art results presented by the Differentially Private Empirical Risk Management framework (Chaudhuri et al., 2011). In order to do so accurately, we use the same data used as for developing this framework. This includes two public machine learning data sets, namely the KDDCup99 and Adult data sets.

In order to test the usability and performance of the algorithm we put forward for approximately solving linear programs to train Support Vector Machines, 10-fold-cross-validated classification test results will be compared across algorithms and settings. The goal is to see

how the new algorithm would perform non-privately, for reference. Solving a linear program this way could be interesting, because for not too extremely large data sets, a non-private model based on a linear programming formulation solved with, for instance, the simplex method, can outperform a quadratic programming model. In addition, for soft-margin Support Vector Machines, we are already allowing restrictions to be violated. The new algorithm can be seen as a boosting controlled violation reduction, and in the case of Support Vector Machines, this could be sufficient to produce well-performing models. Based on the non-private performance, the value of the more interesting private version can be determined in comparison with state of the art results.

We find that the new algorithm works. Depending on parameter settings and data set, one can approximately solve linear programs privately, albeit with relatively poor model performance when applied to training Support Vector Machines. As far as classification using our private linear program solving algorithm goes, the results are never better than from models produced privately with a quadratic programming formulation. However, the proposed algorithm shows promise in its modifiability by the results obtained with a simple task-focused modification.

With these results, this research provides an insight to the possibilities of how to privately produce machine learning models. In addition, we demonstrate a working version of an interesting algorithm for solving linear programs privately. Hopefully, this research sheds some light for the future production of better methods and applications in the field of differential privacy.

1.1 Contributions

The main contributions of this research are twofold. Firstly, we put an interesting theoretical framework to practice in the form of a working Dense Multiplicative Weights algorithm for solving linear programs based on the ideas of Hsu et al. (2014). They provide general solutions for linear programs. We apply their algorithm idea to the working example of training Support Vector Machines in order to validate and test their ideas. In doing so, we demonstrate how to implement their work and what to pay attention to in terms of maintaining privacy guarantees when applying it to a specific problem specification. We find the theoretical solutions provided by Hsu et al. (2014) extremely interesting, but difficult to fully understand and implement from their work. Therefore we hope to provide some insight or clarification on that part.

Secondly, through the above, we provide a differentially private algorithm for different research fields to draw inspiration from. The algorithm can have interesting implications for differentially private linear optimization by providing a working approximate linear program solver for cases where the optimal objective value is known or can be obtained privately. The algorithm

could also still be interesting for machine learning research. This is because we demonstrate how a simple modified version can quickly yield better results. Possibly providing interesting alternatives to output perturbation based privacy solutions.

1.2 Outline

In the following sections of this paper, we will aim to answer the research question. First, we give an explanation and implementation of the tools required for this research in the methodology. We will, most importantly, take a look at Support Vector Machine problem formulations, the concept of differential privacy and explain how we aim to privately solve linear programs. Second, we will perform a computational study and briefly discuss the results in the experimental study section. Finally, we take a more broad look at the results and implications of the research and consider what we have learned and still could learn in the discussion.

2 Methodology

This research is dedicated to the development of differentially private machine learning methods. In this section, we present an explanation and implementation of these concepts, in addition to other concepts which are important to this research. First we take a look at Support Vector Machine models and their different possible problem formulations. Secondly, we introduce the concept of Differential Privacy. Then we introduce the Dense Multiplicative Weights algorithm idea, followed by our problem specific implementation of this idea. Finally, we consider the method of testing our new algorithm.

2.1 Support Vector Machine Models

Support Vector Machines (SVMs) are a class of machine learning models originating from the field of statistical learning, typically used for solving classification problems (Vapnik, 1982). SVM algorithms use labelled training data to construct hyperplanes which best separate the different classes in the data. The algorithms produce model weights w, which represent the hyperplane, such that new incoming data can be labelled according to which side of the hyperplane they fall. In this research, we concern ourselves with linear classification tasks, even though SVMs algorithms are capable of more complex classification tasks through kernel specification. For now, we focus on the linear kernel for solving binary classification problems.

Solving the classification problem by training the SVM means we are constructing the best class separating hyperplane. Assuming the data is linearly separable, we do this by maximizing the distance between two parallel hyperplanes which have the two classes on either side. The

resulting model weights will represent the hyperplane which lies exactly in-between the two supporting hyperplanes. An SVM algorithm finds this result by maximizing the distance between the support hyperplanes while keeping all data points with the same class on the same side of the supporting hyperplane. The distance between two hyperplanes can be represented by the norm of the supporting vectors and as such, SVM algorithms seek to minimize the norm of w. The most common implementation of this is minimizing the Euclidean norm of w, while keeping all data on the correct sides of the supporting hyperplanes, leading to the quadratic programming (QP) formulation for SVMs.

Of course, in practice, training data for binary classification will not be linearly separable. Then, we allow data points to fall on the wrong side of their supporting hyperplane. We do, however, penalize these mistakes. We do so by minimizing the so-called soft-margin error on top of minimizing the norm of w (Cortes & Vapnik, 1995). In this research, we will mostly consider soft-margin SVMs.

Before we continue, it is important to specify that the algorithms in this paper take training data $\mathcal{D} = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y} : i = 1, ..., n\}$ with $\mathcal{Y} = \{-1, 1\}$ and assuming $\mathcal{X} = \{x : x \in \mathbb{R}^p, \|x\|_2 \leq 1, \|x\|_\infty \leq 1\}$ for the classification problems. Throughout this research, the infinity norm represents $\|x\|_\infty = \max(|x_j| : j = 1, 2, ..., p)$.

Now, in order to train SVMs, we solve the problem presented by the QP-SVM formulation, which is of the form:

$$\begin{aligned} \text{minimize} \quad & \|w\|_2^2 + C \textstyle\sum_{i=1}^n \xi_i \\ \\ \text{subject to} \quad & y_i x_i^\intercal w && \geq 1 - \xi_i, \quad i = 1, ..., n, \\ & \xi_i && \geq 0, \qquad i = 1, ..., n, \\ & w && \in \mathbb{R}^p, \end{aligned} \tag{1}$$

where w are, once again, the model coefficients, ξ_i are the soft-margin coefficients and $C > 0$ is the soft-margin error trade-off parameter.

We can solve this QP by incorporating the restrictions in the objective function by means of the Hinge loss function. The Hinge loss function simply counts 1 if the restriction is violated by the model and 0 if not. This loss function is convex, but not differentiable. Therefore, in line with the work presented by Chaudhuri et al. (2011), we solve the QP by minimizing the following loss function:

$$L_{QP}(w, D) = \frac{\Lambda}{2}\|w\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} \ell_{Huber}(y_i x_i^\intercal w), \tag{2}$$

where Λ is a regularizer, $D \in \mathcal{D}$, and the soft-margin variables ξ are represented by a loss function for constraint violation. Instead of the Hinge loss function, we use the Huber loss function, which is a differentiable approximation of the Hinge loss function:

$$\ell_{Huber}(z) = \begin{cases} 0 & \text{if } z > 1 - h, \\ \frac{1}{4h}(1 + h - z)^2 & \text{if } |1 - z| \leq h, \\ 1 - z & \text{if } z < 1 - h. \end{cases} \tag{3}$$

For the implementation in this research, we use $h = 0.5$, again in line with Chaudhuri et al. (2011). The above function says that if $y_i x_i^\intercal w$ is sufficiently positive, meaning the predicted class using coefficients w is correct, the loss is 0.

Now, in this research, for the purpose of exploring alternative solutions, we consider a different SVM problem formulation. Namely, the linear programming (LP) formulation (Mangasarian, 1998). Instead of the Euclidean norm, the objective minimizes the absolute norm of the model weights. Additionally, to adhere to the standard form of LP, the formulation requires the problem coefficients to be non-negative. To achieve this, we rewrite the variable weights term using $w = w^+ - w^-$, such that $\|w\|_1 = \sum_{j=1}^{p}(w_j^+ + w_j^-)$. Basically, we introduce decision variables $w^+, w^- \in \mathbb{R}^p$ instead of w. The idea is that in the optimal solution, because we want to minimize $\|w\|_1$, only one of $w_j^+$and w_j^- can be non-zero for all j. Meaning that an optimal solution will have non-negative decision variables, but w_j can be negative if w_j^- is non-zero. This transformation results in the LP

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{j=1}^{p}(w_j^+ + w_j^-) + C\sum_{i=1}^{n}\xi_i \\ \\ \text{subject to} \quad & y_i x_i^\intercal(w^+ - w^-) && \geq 1 - \xi_i, \quad i = 1, ..., n, \\ & \xi_i && \geq 0, \qquad i = 1, ..., n, \\ & w_j^+, w_j^- && \geq 0, \qquad j = 1, ..., p. \end{aligned} \tag{4}$$

LP (4) can in turn be reduced to the standard form

$$\begin{aligned} \text{minimize} \quad & c^\intercal \bar{x} \\ \\ \text{subject to} \quad & A\bar{x} && \leq b, \\ & \bar{x} && \geq 0 \end{aligned} \tag{5}$$

which is equivalent to 4 if we use

$$
\begin{aligned}
\underset{1\times(2p+n)}{c^\intercal} &= [\ \underset{1\times 2p}{\bar{1}_{2p}^\intercal}\ ,\ \underset{1\times n}{C\bar{1}_n^\intercal}\ ], \\
\underset{1\times(2p+n)}{\bar{x}^\intercal} &= [\ \underset{1\times p}{(w^+)^\intercal}\ ,\ \underset{1\times p}{(w^-)^\intercal}\ ,\ \underset{1\times n}{\xi^\intercal}\ ], \\
\underset{n\times(2p+n)}{A} &= -[\ \underset{n\times n}{\mathrm{diag}(y)}\underset{n\times p}{X^\intercal}\ ,\ -\underset{n\times n}{\mathrm{diag}(y)}\underset{n\times p}{X^\intercal}\ ,\ \underset{n\times n}{I_n}\ ], \\
\underset{n\times 1}{\bar{b}} &= -\underset{n\times 1}{\bar{1}_n},
\end{aligned}
$$

where $[a, b]$ represents a concatenation of vectors a and b, I_m is a size m identity matrix, $\bar{1}_m$ a size m vector of ones, and y, X the vector and matrix constructed by all $y_i, x_i \in D$. In reference to the standard form LP, $\bar{x}$ is used to represent the vector of decision variables and is not related to the explanatory variable vectors x_i.

This final formulation can be used to obtain an SVM model with LP-solving tools. In this research, we consider an approximate LP-solving tool utilized by Hsu et al. (2014), which will be explained in section (2.4).

2.2 Differential Privacy

Differential privacy (DP) is a framework based on individual data security. Originally motivated by seeking privacy for records in statistical database querying, it was first introduced by Dwork (2006). The main idea of the framework is that query output should not be able to reveal anything about any specific record. If we, for instance, have a database with three records containing the salary of three people and we obtain the average of their salaries, then we know nothing about the individuals. However, if we were knowledgeable on this average before removing a single record (or omitting one from the query) and obtained the new average, we could deduce the salary of the individual who's record was omitted. Under DP, the output of these respective queries would be adjusted in such a way that the outputs cannot be used to reliably draw any conclusion about the information of the omitted record. For instance through adding noise. Formally, DP is defined as follows, using the concept of mechanism, which can refer to anything along the lines of a database query, function or algorithm.

Definition 2.1. Mechanism $\mathcal{M}$ is (ϵ, δ)-differentially private if for all datasets D and D' differing in at most one record, and all $\mathcal{S} \subset range(\mathcal{M})$,

$$Pr[\mathcal{M}(D) \in \mathcal{S}] \leq exp(\epsilon) \times Pr[\mathcal{M}(D') \in \mathcal{S}] + \delta. \tag{6}$$

Put differently, adding or removing a record from an input data set does not significantly

affect the output probability of mechanism $\mathcal{M}$. Here, ϵ represents the available privacy budget. The smaller this budget, the more private the mechanism needs to be made. The approximation parameter δ allows the mechanism to fail to guarantee privacy with a probability of δ. This means that there is a probability of δ that the output of a mechanism is unique in the sense that it cannot be obtained by a neighbouring database, even under DP. If we have $\delta = 0$, the mechanism is ϵ-DP, which is referred to as pure DP, meaning the mechanism will not fail. Throughout this research, when applicable, we will set $\delta = \frac{1}{n}$. This is because when the mechanism is allowed to fail, it is allowed to do so for each possible neighbouring database, meaning each individual's information is at risk with probability δ. Therefore, to reduce the expected rate of failure, one should normally set $\delta < \frac{1}{n}$. However, since there is no other applicable rule of thumb and our focus lies elsewhere, we will stick to the setting $\delta = \frac{1}{n}$.

We guarantee differential privacy by adding noise to the data or to the mechanism output. Dwork et al. (2006) demonstrate a more general application of the framework. They introduce the concept of the sensitivity method. Assuming mechanism $\mathcal{M}$ can be anything along the lines of a function or algorithm with a vector output, we define the sensitivity of this mechanism $S \in \mathbb{R}_+$ as follows.

Definition 2.2. A mechanism $\mathcal{M} : \mathcal{D} \to \mathbb{R}^p$ is S-sensitive if for all datasets D and D' differing in at most one record,

$$S(\mathcal{M}) \geq \max_{D,D'} \|\mathcal{M}(D) - \mathcal{M}(D')\|_1. \tag{7}$$

This sensitivity is used to calibrate a level of noise to be added to the mechanism output, in order to mask the input data. The noise can be generated using a Gaussian or Laplacian distribution, but we opt for Laplacian noise to stay in line with the related research. Laplacian noise follows the density

$$f(\mathbf{v}) \propto \exp(-\nu\|\mathbf{v}\|) \tag{8}$$

with mean 0 and standard deviation $\frac{1}{\nu}$. In practice, ϵ-DP is achieved by perturbing the mechanism output with Laplacian noise with $\nu = \frac{\epsilon}{S}$. This method is also referred to as output perturbation.

In the work of Chaudhuri and Monteleoni (2009), they build on this framework to propose a privacy-preserving machine learning algorithm design. They introduce the idea of objective perturbation, where the objective function is perturbed before optimization. This work is later applied to the Empirical Risk Minimization framework (Chaudhuri et al., 2011), which we will go into in the next section.

Compositions of ϵ-DP mechanisms have relatively predictable behavior. If outputs of differ-

ent mechanisms are based on the same data, the composition of those different outputs maintains a cumulative privacy budget. If they do not overlap in underlying data, the composition is as private as the least private part. In the case of cumulative privacy budgeting for (ϵ, δ)-DP mechanisms, Dwork et al. (2010) provide a more efficient result in the form of composition theorem (1).

Theorem 1. *For any $\delta \in (0, 1)$, the composition of T ϵ'-DP mechanisms is (ϵ, δ)-DP for*

$$\epsilon' = \frac{\epsilon}{\sqrt{8T \log(1/\delta)}}.$$

Beside the sensitivity method, another method for achieving differential privacy is the exponential method (McSherry & Talwar, 2007). Instead of adding noise to the mechanism output, one selects the output randomly from the mechanism's output space $\mathcal{R}$. The idea is that the solutions in the output space are assigned a quality score, which is calculated using a quality score function, and the probability of choosing one such a solution is proportional to their quality and the level of privacy required.

Definition 2.3. Let $\epsilon > 0$ be given and suppose that each possible mechanism output $r \in \mathcal{R}$ is assigned a quality score $Q : \mathcal{R} \times \mathcal{D} \to \mathbb{R}$, which has a maximum sensitivity of $S(Q)$. Then, for database D, the ϵ-DP exponential mechanism with quality score $Q(r, D)$ outputs a solution $r \in \mathcal{R}$ with probability proportional to

$$\exp\Big(\frac{\epsilon}{2S} Q(r, D)\Big). \tag{9}$$

A highly sensitive quality score could render different solutions nearly identical in probability of selection, making the selection nearly uniform. Even if S is sufficiently small and the solution probabilities are properly representative of their quality, mechanisms with a large solution space will suffer greatly from this privacy preserving method, since the probability of choosing the best solution will still diminish. If, however, the solution space is small enough and the quality score aptly chosen, this method can be very interesting for producing differentially private algorithms.

2.3 Output perturbation and objective perturbation

The first two algorithms we consider for reference are the output and objective perturbation algorithms for training SVMs privately put forward by Chaudhuri et al. (2011). Both algorithms use the Laplace mechanism for differential privacy by adding Laplacian noise somewhere in the algorithm such that the input of the algorithm remains private. The size of this noise is based on the desired level of privacy ϵ and the sensitivity of the mechanism to be made private.

Output perturbation refers to the method of adding noise to real valued output of an algorithm in order to mask private input. Algorithm 1 gives the steps for output perturbation as implemented in this research for privately training QP-SVMs:

Algorithm 1: Training SVMs with output perturbation

Input: Data D, privacy parameter ϵ, Λ

Output: Model coefficients w

Draw vector $\mathbf{v}$ from density (8) with $\nu = \frac{n\Lambda\epsilon}{2}$

Compute $w = \text{argmin}(\ L_{QP}(w, D)\ ) + \mathbf{v}$.

Objective perturbation implies minimizing a noisy objective function. Chaudhuri and Monteleoni (2009) first proposed this algorithm for logistic regression, but it was later modified such that it is also applicable to SVMs. The algorithm, as this research implements it, is shown in Algorithm 2.

Algorithm 2: Training SVMs with objective perturbation

Input: Data D, privacy parameter ϵ, Λ, q

Output: Model coefficients w

Set $\epsilon' = \epsilon - \log(1 + \frac{2q}{n\Lambda} + \frac{q^2}{n^2\Lambda^2})$

If $\epsilon' > 0$

$\Delta = 0$

Else

$\Delta = \frac{q}{n(\exp(\frac{\epsilon}{4})-1)} - \Lambda$, $\epsilon' = \frac{\epsilon}{2}$

Draw vector $\mathbf{v}$ from density (8) with $\nu = \frac{\epsilon'}{2}$

Compute $w = \text{argmin}(\ L_{QP}(w, D) + \frac{1}{n}\mathbf{v}^\intercal w + \frac{\Delta}{2}\|w\|_2^2\ )$.

Proofs for privacy guarantees of the two algorithms can be found in the paper of Chaudhuri et al. (2011). Throughout this research, the above algorithms are implemented by minimizing the specified loss function using the `fminfunc` function in MATLAB.

2.4 Privately solving LPs using Dense Multiplicative Weights

The solution we consider for solving the LP-formulated SVM problem is the Dense Multiplicative Weights (DMW) Algorithm for solving Linear Programs privately, adapted by Hsu et al. (2014). Specifically, the algorithm specialized for privately solving LPs with highly sensitive constraints. This refers to neighbouring LPs that differ only in at most a single entire constraint. This applies to LP-SVMs, because each row in A in (5) represents a single data point. Hsu et al. (2014) provide the ideas and proofs for the algorithm for solving such LPs privately. Our

implementation of the algorithm, specific to our problem setting, is based on their Algorithm 2. The main idea of the algorithm is to start from the optimal solution OPT $= c^\intercal \bar{x}_{opt}$, and then privately and repeatedly solve a feasibility problem that returns a solution that at least satisfies OPT $= c^\intercal \bar{x}$ and tries to violate the least amount of constraints possible. In the fashion of a boosting algorithm, this problem takes weights $z \in \mathbb{R}^n_+$ which in this case are assigned to the n constraints in the LP. These weights update iteratively in order to promote the importance of previously broken constraints. However, in order to maintain privacy for the constraints, i.e. in order to hide whether a data point is well presented by the model or not, we find solutions only covering most of the constraints. This is done by solving the modified (weighted) feasibility problem

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{i=1}^{n} z_i(A_i \bar{x}) \\ \text{subject to} \quad & c^\intercal \bar{x} = \text{OPT}, \\ & \bar{x} \geq 0. \end{aligned} \tag{10}$$

Here, A_i represents a single row in matrix A. The solution to the above problem has only one active variable in vector $\bar{x}$ due to having only a single equality constraint. Therefore, the problem would have $2p + n$ possible solutions in the case of soft-margin SVM LP's. We will call these solutions $\mathcal{K}$ such that $\bar{x} \in \mathcal{K}$ if $\bar{x}$ is a possible solution to (10). Due to the single equality constraint, if c^T were just a vector of ones, we would know that all solutions in $\mathcal{K}$ are of the form OPT$\cdot e_j$, where e_j signifies a zero vector with a 1 at the jth index. Privacy is achieved by means of the so-called exponential mechanism (McSherry & Talwar, 2007). Each iteration, a solution is randomly selected. Better solutions are given a higher probability of selection through quality scores, which in this case are based on the objective function of (10). In order to obtain a bound on the sensitivity of these quality scores, the weights z are projected into a set of dense distributions through means of Bregman projections. For the exact usage in the privacy guarantees and other proofs, we refer to (Hsu et al., 2014).

Definition 2.4. Let density parameter $s > 0$. Given a vector z such that $|z| \leq s$ and $z \in [0, 1]^p$, let $\Gamma_s z$ be the (Bregman) projection of z into the set of $1/s$-dense distributions, defined by $\Gamma_s z_j = \frac{1}{s} \cdot \min(1, \zeta z_j)$ for $j = 1, ..., p$, where $\zeta \geq 0$ such that $s = \sum_{j=1}^{p} \min(1, \zeta z_j)$

Of course, the solutions with a single active variable do not provide a good model, but the algorithm averages all solutions after repeating this process in many boosting iterations in order to finally output a solution model that still satisfies $c^\intercal \bar{x} =$ OPT, and hopes to satisfy most constraints.

2.5 Privately training SVMs using the DMW algorithm

Originally, the algorithm puts a lot of focus on the accuracy with which it approximates the LP. In this paper, we are largely disregarding this side of the algorithm, because we are focusing on the privacy guarantees and the out-of-sample model results. Therefore, input parameter α, as used by Hsu et al. (2014), is not interesting to us. Instead, we only really still implement it to stay somewhat faithful to the original shape of the algorithm, while using it to set the amount of boosting iterations T. Additionally, β, which we see as an output parameter based on the input parameter settings, is left completely out of view.

In order to implement the private DMW Algorithm for our problem setting, several details must be addressed. The foremost being the non-private calculation and use of OPT in the implementation by Hsu et al. (2014). In their Private Fractional Set Cover example, they assume OPT is known and public, while this does not hold for us. As they rightfully point out, using an OPT that is calculated from private information (and thus changes when the data changes) causes additional overhead in privacy. To combat this, we must use the sensitivity method and use a private OPT. Because obtaining a private solution is the problem we are trying to solve in this paper, we must take a shortcut to obtain a private OPT. This is where algorithms 1 and 2 may come in. Obtaining private model weights and using these to calculate a private OPT does not necessarily require an LP formulation if we assume the QP formulation leads to a good enough value for OPT. In light of this, we should prefer Algorithm 2 for this, because adding noise to output could lead to more extreme values of OPT since it is mostly calculated from the absolute norm of the model weights. Using the DMW Algorithm with a perturbed OPT will hopefully lead to a smaller cost in model accuracy compared to simply using output perturbation on the model weights, since the objective value is of relative little impact on a soft-margin model.

Another issue with the algorithm is the fact that the solution space of the exponential mechanism in the iterations scales with data set size. This can lead to increasing inaccuracy for large data sets. A simple solution is letting the DMW Algorithm solve a hard-margin SVM LP based on the (private) OPT of a soft-margin SVM LP. This would require the OPT to be calculated from the model weights only, therefore disregarding the soft-margin after acquiring an initial private solution using the sensitivity method. In the context of SVM LPs, not satisfying each constraint is not a high price to pay if the model performs well generally. Additionally, this way, the privacy mechanism in the algorithm can even be interpreted as an implicit soft-margin as the algorithm still tries to satisfy as many constraints as possible. This change means that

the standard form variables are changed to

$$
\begin{aligned}
\underset{1\times 2p}{c^\intercal} &= \underset{1\times 2p}{\bar{1}^\intercal_{2p}}\,, \\
\underset{1\times 2p}{\bar{x}^\intercal} &= [\ \underset{1\times p}{(w^+)^\intercal}\ ,\ \underset{1\times p}{(w^-)^\intercal}\ ], \\
\underset{n\times 2p}{A} &= -[\ \underset{n\times n}{\operatorname{diag}(y)}\underset{n\times p}{X^\intercal}\ ,\ -\underset{n\times n}{\operatorname{diag}(y)}\underset{n\times p}{X^\intercal}\ ], \\
\underset{n\times 1}{\bar{b}} &= -\underset{n\times 1}{\bar{1}_n}.
\end{aligned}
$$

Finally, we need to calculate the width of a hard-margin SVM LP and the accompanying quality score sensitivity in order to get the DMW Algorithm to work and be private. Hsu et al. (2014) provide a theoretical example in the form of a Private Fractional Set Covering problem. This problem closely resembles our problem setting, except for that the rows of the constraint matrix A are not just zeros and ones, but have values ranging from -1 to 1. As for the width of the LP, due to the definition by Hsu et al. (2014), we get

$$
\begin{aligned}
\rho &\geq \max_{\bar{x}\in\mathcal{K}} \|A\bar{x} - b\|_\infty \\
&= \max_{\bar{x}\in\mathcal{K}} \max_{i=1,\ldots,n} |A_i\bar{x} + 1| \\
&= \max_{j=1,\ldots,2p} \max_{i=1,\ldots,n} |\mathrm{OPT} A_i e_j + 1| \\
&= \mathrm{OPT} + 1.
\end{aligned}
\tag{11}
$$

For the sensitivity of the quality score, as they consider $Q = \sum_{i=1}^{n} z_i(-A_i\bar{x} - b_i)$ where $\bar{x} \in \mathcal{K}$ and z is a Bregman projection, we get

$$
\begin{aligned}
S(Q) &= \max_{\bar{x}\in\mathcal{K},D,D'} |Q - Q'| \\
&= \max_{\bar{x}\in\mathcal{K},D,D'} \left| \sum_{i=1}^{n} z_i(-A_i\bar{x} - b_i) - \sum_{i=1}^{n'} z'_i(-A'_i\bar{x} - b'_i) \right| \\
&= \max_{j=1,\ldots,2p} \max_{D,D'} \left| \sum_{i=1}^{n} z_i(-\mathrm{OPT}A_i e_j + 1) - \sum_{i=1}^{n'} z'_i(-\mathrm{OPT}A'_i e_j + 1) \right| \\
&= \max_{j=1,\ldots,2p} \max_{D,D'} \left| -\mathrm{OPT}\sum_{i=1}^{n} z_i A_{ij} + 1 + \mathrm{OPT}\sum_{i=1}^{n'} z'_i A'_{ij} - 1 \right| \\
&= \max_{j=1,\ldots,2p} \max_{D,D'} \left| \mathrm{OPT}\left(\sum_{i=1}^{n'} z'_i A'_{ij} - \sum_{i=1}^{n} z_i A_{ij} \right) \right| \\
&= \max_{j=1,\ldots,2p} \max_{D,D'} \left| \mathrm{OPT}\left(\sum_{i=1}^{n} z'_i A_{ij} + z'_{n'} A'_{n'j} - \sum_{i=1}^{n} z_i A_{ij} \right) \right| \\
&= \max_{j=1,\ldots,2p} \max_{D,D'} \left| \mathrm{OPT}\left(\sum_{i=1}^{n} A_{ij}(z'_i - z_i) + z'_{n'} A'_{n'j} \right) \right| \\
&\leq \max_{D,D'} \left| \mathrm{OPT}\left(\|z'_{1:n} - z\|_1 + \frac{1}{s} \right) \right| \\
&\leq \left| \mathrm{OPT}\left(\frac{2}{s} + \frac{1}{s} \right) \right| \\
&= \frac{3\mathrm{OPT}}{s}.
\end{aligned}
\tag{12}
$$

For the final important step in (12), we use the following result related to Bregman projections.

Lemma 2. (Hsu et al., 2014). Let $s > 0$ be given. Suppose z, z' are neighbouring weight vectors. If $\tilde{z}, \tilde{z}'$ are the respective Bregman projections into the set of $1/s$-dense distributions, then

$$\|\tilde{z} - \tilde{z}'\|_1 \leq \frac{2}{s}$$

With the above adjustments to suit our problem specification, we come to the implementation as found in Algorithm 3.

Algorithm 3: Training SVMs with differentially private Dense Multiplicative Weights

Input: Data D, privacy parameters ϵ, δ, density parameter s, accuracy parameter α

Output: Model coefficients w

Obtain $\text{OPT} = \|w_{obj}\|_1$ where w_{obj} is the output from Algorithm (2)

Set $\rho = \text{OPT} + 1$, $T = \frac{36\rho^2 \log n}{\alpha^2}$ $\eta = \sqrt{\frac{\log(n)}{T}}$, $\epsilon' = \frac{\epsilon}{\sqrt{8T\log(\delta^{-1})}}$, $\Delta = \frac{3\text{OPT}}{s}$

Initialize $z_i^0 = \frac{s}{n},\ i = 1, ..., n$

For $t = 1, ..., T$

Set $\zeta > 0$ such that $s = \|\min(1, \zeta z^t)\|_1$

Obtain projection $\tilde{z}^t = \frac{\min(1,\zeta z^t)}{s}$

For $j = 1, ..., 2p$

Compute j^{th} solution $\tilde{x}^* = \text{OPT}e_j$

Compute quality score $Q_j = \sum_{i=1}^n \tilde{z}_i^t(-A_i\tilde{x}^* - b_i)$

Select j with $\Pr(j = l) \propto \exp(\frac{\epsilon'}{2\Delta}Q_l),\ j = 1, ..., 2p$

Set $\tilde{x}^t = \text{OPT}e_j$

Compute losses $\ell_i^t = \frac{1}{2\rho}(b_i - A_i\tilde{x}^t) + \frac{1}{2},\ i = 1, ...n$

Update weights $z^{t+1} = \exp(-\eta\ell^t)z^t$

Compute $\bar{x} = \frac{1}{T}\sum_{t=1}^T \tilde{x}^t$

Compute $w = \bar{x}_{1:p} - \bar{x}_{p+1:2p}$

This algorithm computes a solution $\bar{x}$ to an LP like (5), such that $A_i\bar{x} \leq b_i + \alpha$, except for at most s constraints, with probability at least $1 - \beta$. The ϵ, δ privacy guarantee of the algorithm follows from Theorem (1), because each iteration t, a solution is chosen with ϵ'-DP. Additional proofs for accuracy and privacy guarantees of the algorithm, and more in-depth explanations can be found in the paper of Hsu et al. (2014). Since we are more interested in out-of-sample prediction results of our model than the accuracy guarantees of LP-solving, we will leave these out.

2.6 Out of sample testing and K-fold cross-validation

Before we continue with the experimental study of the above-mentioned algorithms, we touch on the method we use for out-of-sample model testing, since we are most interested in the out-of-sample classification performance results of our algorithms. The results in the experiments are primarily based on the test set error rates. We split the data into training and test sets. The training data is used to produce a classification model which we test on the test data. The error rate refers to the fraction of erroneous classifications made by the model on the test data. In order to prevent bias from splitting the data, we use K-fold cross-validation. K-fold cross-validation refers to the method of splitting the data into K different training and test sets. If a data set has size n, then we obtain K different, disjoint test sets of size $\frac{n}{K}$. K can be chosen as large as n and as small as 2. With the former, we obtain leave-one-out-validation. This form of result validation is very exhaustive, since it means you have to run the algorithm you are testing n times. Testing it on a single data point each time. Despite its exhaustiveness, since each run uses all but one data points to create a model, there is very little training set bias. If we only have two different training sets which are half the original size, one can expect the resulting models to differ greatly based on which data they are based on. However, on top of being exhaustive, leave-on-out-cross-validation has a higher test error rate variance than if $K < n$. Generally speaking, choosing K comes with making a bias-variance trade-off (James et al., 2013). A general rule of thumb is to pick $K = 10$, because it is non-exhaustive and yields test error rates which suffer from neither high bias nor high variance.

3 Experimental study

In this section we conduct a number of numerical experiments to investigate the classification performance of Algorithm 3. All results produced by the experiments are acquired using 10-fold cross-validation on the data sets. We opt for 10-fold cross-validation because we want to verify consistency of model results without exhaustively running the algorithms on the large data sets. This type of validation is especially important because the differential privacy accuracy-privacy trade-off requires a lot of randomness in the model results. For algorithms 1 and 2, we use the state-of-the-art parameter settings as presented by Chaudhuri et al. (2011).

3.1 Data

In these experiments, the Adult (Dua & Graff, 2017) and KDDCup99 (Hettich & Bay, 1999) data sets are used. Before use, the data is processed such that all values in X are reduced to the

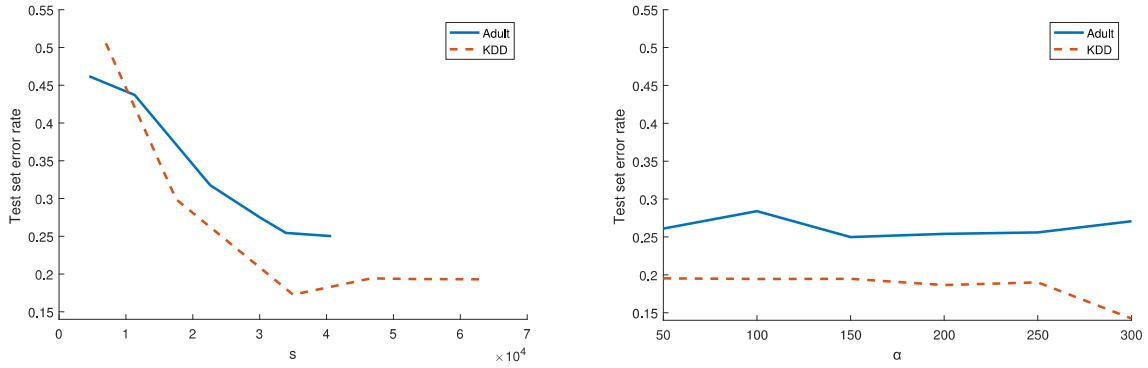
range $[-1, 1]$ and each vector representing a record is normalized such that $\|x_i\| = 1, i = 1, ..., n$.

The Adult data set consists of 45,220 labeled records of individuals. Each individual has an annual income of either below or above \$50,000 and is labeled as such. The data set contains features such age, sex, education, etc. As per construction, all data is normalized.

The KDDCup99 (KDD) data set initially contains millions of records of network connections which are labeled as a DOS attack or not. For the purpose of this research we use a random subsample of 70,000 records from the official 10% corrected subset.

3.2 Parameter testing for the private Dense Multiplicative Weights algorithm

First off, we do some parameter testing for Algorithm 3. From the originating work, there is no clear guideline as to what one should use as values for the density and accuracy parameters. All we know is that they are restricted to $0 < s \leq n$ and $0 < \alpha < 9\rho$. Once again, density parameter s signifies up to how many constraints are not guaranteed to be satisfied. A larger s means potentially more violated constraints. Accuracy parameter α signifies the margin of by how much violation we still classify a constraint as satisfied. Most importantly for us, α translates into how many boosting iterations the algorithm uses. We try different values for one parameter while keeping the other fixed, resulting in Figure (1). The fixed values are based on preliminary testing with a wide range of values. The fixed values for Figures (1a) and (1b) are $s = \frac{3}{4}n$ and $\alpha = 200$ for both data sets respectively.



(a) Test error rate results for different values of s (b) Test error rate results for different values of α

Figure 1: Cross-validated test error rate results for parameter testing of Algorithm 3 with $\epsilon = 0.1$ for different data sets

Figure (1a) shows a simple result where a larger value for s is more desired, with diminishing results as we get closer to n. This is not entirely unexpected, because s signifies how many restrictions the algorithms allows to be violated, and in our SVM setting, restriction violation is not necessarily a bad thing so long as we still get a good output model. Therefore, we want

to choose a large s, without getting too close to n. Hence, for our fixed value, we opt for a value of $s = \frac{3}{4}n$.

Figure (1b) shows a relative indifference to α by the algorithm. This is initially surprising since for our setting, α is basically a proxy for choosing the amount of boosting iterations T. A small α means a larger T. However, the relative indifference to changes in α is perhaps expected due to the fact that if we choose a large s, and thus allow many restrictions to be violated, which we want to do, α has fewer constraints left to have an influence over. Our fixed value of $\alpha = 200$ was chosen to balance the running time and performance of the algorithm. Intuitively, a smaller T will lead to an insufficient amount of iterations for the boosting part of the algorithm to provide a good model. Additionally, a large T will reduce the quality of each iteration's solution, because the privacy budget needs to be spread more thinly.

3.3 Non-private Dense Multiplicative Weights for solving Linear Programs

Before we continue with putting Algorithm 3 to the test, we take a quick look at a non-private setting. This means we want to know how Algorithm 3 would perform if there were no privacy requirements. Intuitively, because it is very much an approximating boosting algorithm, we expect the algorithm to produce a worse model than the model we get from simply minimizing the QP loss function (2). We would, however, like to know if the algorithm at the very least works in producing a model by approximately solving the LP.

The non-private results in Table (1) are obtained using three different non-private models. The Fixed model is one that always predict the largest class in the training set. The QP-SVM model is obtained by minimizing (2). The DMW model is obtained by running Algorithm 3 with the adjustment that the exponential mechanism is guaranteed to always pick the highest scoring solution.

Data	Model		
	Fixed	QP-SVM	DMW
Adult	0.248	0.173	0.243
KDD	0.195	0.025	0.063

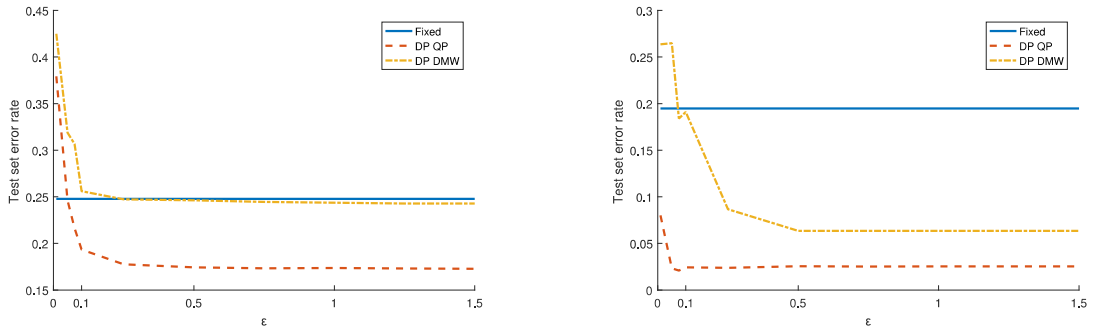
Table 1: Cross-validated test error rates for different non-private models

Table (1) demonstrates that despite its relative poor performance, Algorithm 3 succeeds in producing a working classification model based on the LP-SVM formulation. This is a promising result, as this means that, within the algorithm, we only need the exponential mechanism to pick a good enough solution often enough to output a working model.

3.4 Privately training Support Vector Machines

Now we finally compare the performance of the private algorithms 2 and 3. For Algorithm 2, we use the parameter settings yielding the best results for the respective data sets as presented by Chaudhuri et al. (2011). The parameter settings for Algorithm 3 include, once again, $\alpha = 200$ and $s = \frac{3}{4}n$ for both data sets.

For reference, the Fixed model error rate as explained in Section (3.3) is included in Figure (2).



(a) Test error rate results for the Adult data set (b) Test error rate results for the KDD data set

Figure 2: Cross-validated test error rate results for different values of ϵ for different classification models

Figure (2) demonstrates the effect the available privacy budget has on algorithms 2 and 3. Very much in line with Table (1), we see that from $\epsilon = 0.5$ onwards, the classification results match the non-private setting. It is no surprise that for a restricted budget, both algorithms perform poorly compared to the (factually non-private) Fixed predictor. We see that the algorithms have similar trends, but that in terms of model performance, Algorithm 3 does not provide any kind of better solution than Algorithm 2. This is no surprise, as one can not expect an algorithm privately reconstructing the model output of another algorithm to outputperform said algorithm. Algorithm 3 is built on losing information and Figure (2) very much shows this. It does, however, work in the sense that is does a relative good job of private classification. It is clear that due to the restriction of the private starting point of the algorithm, Algorithm 3 should never perform well in a private SVM setting.

3.5 Adjusting the quality score to better suit Support Vector Machines

With the above results, despite Algorithm 3 being poorly suited for the task at hand, we want to know if we can still improve the algorithm for the SVM classification task. Therefore, we seek to make a task-focused modification to the algorithm. The most obvious point of improvement in the algorithm is the quality score. It is, of course, based on solving problem (10), which is

an excellent solution for approximately solving LPs. For our problem specification, however, we would like to simplify the accompanying quality score. As it stands, the quality score rewards restriction satisfaction as well as punishing restriction violation. This is all well and good, except for the fact that the quality score works with absolute differences. This means that a point that is far away from the separating plane will have great negative or positive effect on the quality score. We want to reduce this effect by reducing the quality score to a simple count. Instead of summing the distances from the separating plane, we count the amount of satisfied restrictions. Additionally, we want to reintroduce another form of soft-margin φ. In Algorithm 3, we discard the proper soft-margin formulation after acquiring OPT. Reintroducing a form of leeway may contribute to the quality of the algorithm adjustment. This particular soft-margin-type modification means also counting restrictions that are only slightly violated:

$$Q(\tilde{x}, D) = \sum_{i=1}^{n} z_i I(A_i \tilde{x} \leq b_i + \varphi), \tag{13}$$

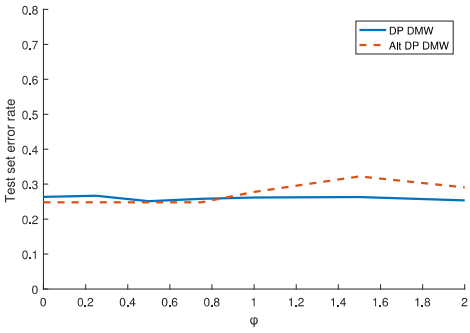
where $I(\cdot)$ is the indicator function.

Before we continue, we need to verify and set the sensitivity of this quality score. We get

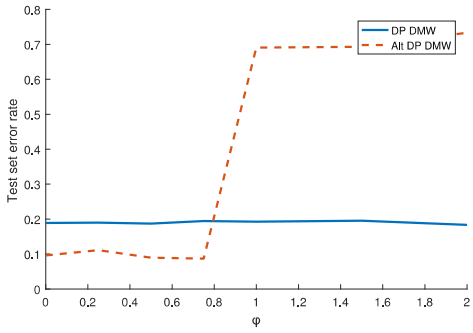
$$\begin{aligned}
S(Q) &= \max_{\bar{x} \in \mathcal{K}, D, D'} |Q - Q'| \\
&= \max_{\bar{x} \in \mathcal{K}, D, D'} \left| \sum_{i=1}^{n} z_i I(A_i \tilde{x} \leq b_i + \varphi) - \sum_{i=1}^{n'} z'_i I(A'_i \tilde{x} \leq b'_i + \varphi) \right| \\
&= \max_{\bar{x} \in \mathcal{K}, D, D'} \left| \sum_{i=1}^{n} (z_i - z'_i) I(A_i \tilde{x} \leq -1 + \varphi) - z'_{n'} I(A'_{n'} \tilde{x} \leq -1 + \varphi) \right| \\
&= \max_{D, D'} \left| \sum_{i=1}^{n} (z_i - z'_i) \cdot 1 - z'_{n'} \cdot 0 \right| \\
&\leq \max_{D, D'} \|z - z'_{1:n}\|_1 \\
&\leq \frac{2}{s}.
\end{aligned} \tag{14}$$

Thereby adjusting the value for Δ in Algorithm 3 to $\frac{2}{s}$.

We now test the performance of this version of the algorithm for different values of φ. In the below results, we refer to this modified version as Alt DP DMW. With “Alt” as in alternative.



(a) Test error rate results for the Adult data set



(b) Test error rate results for the KDD data set

Figure 3: Cross-validated test error rate results for different values of φ for different quality score for Algorithm 3

Figure (3) shows that only a change of quality score can improve the classification results of Algorithm 3. Indeed, the DMW algorithm was designed for approximately solving LPs and not for training SVMs. It is, however, encouraging that with slight tweaks, the algorithm can be set to the task. As for the attempted additional soft-margin implementation, the above figures show that it is best left at $\varphi = 0$: a hard-margin restriction satisfaction count. For both data sets, at some point, the soft-margin seems to cause the training to fail.

In both figures, the performance of the DP DMW model can be seen to vary slightly over the different runs, where you should expect the result to be the same for all φ. This is actually not an unexpected result due to the random nature of differential privacy and the exponential mechanism.

4 Discussion

Using the DMW algorithm for solving LPs privately to train LP-SVMs merits mixed results. On the one hand, it is not a good alternative to existing solutions for solving classification problems. On the other hand, the algorithm demonstrates, in practice, a solution for approximate LP solving. The algorithms shows to be adjustable to better suit the learning tasks it was not originally designed for. This is an interesting result which could lead to more interesting solutions to machine learning or optimization problems using differential privacy. There are, however, some problems with this research that need to be taken into account before concluding on the usefulness of the private DMW algorithm.

The comparisons of the QP and LP based SVM training algorithms is not an entirely fair one. Algorithm 3 works under approximate DP, by incorporation of parameter δ. For the above results we have set this parameter to $\delta = 1/n$, which makes the algorithm tend towards ϵ-DP, at the cost of exponential mechanism accuracy due to a smaller ϵ'. Additionally, by construction,

the algorithm attempts to approximate the result of Algorithm 2 in order to preserve differential privacy. Logically, the algorithm could be more interesting in a setting where this starting point can be determined non-privately.

Although the DMW algorithm is relatively successful at training SVMs, its success at accurately solving LPs under differential privacy remains to be tested. Of course, this research is a promising result in the application of the previously theoretical framework of Hsu et al. (2014), but depending on the problem specification, the algorithm may not perform as desired.

For future research it would be interesting to go back to the original intentions of Hsu et al. (2014) and test the practical validity of their algorithms. In this research we focus on neighbouring LPs that are highly sensitive in their constraints, but neighbouring LPs could for instance be sensitive in cost function as well. Such different settings would require different algorithms and Hsu et al. (2014) provide several, albeit theoretical, solutions. It would be valuable to find problem specifications where these algorithms provide a working solution.

On the other hand, privately training LP-SVM might yet yield interesting results by further tweaking the private DMW algorithm to better suit such a problem specification. This does not necessarily have to concern the quality scoring component of the algorithm. We expect that strides can be made with a different initialization of OPT as well. As it stands, OPT is based on a noisy classification model. If a problem specification allows for a better, perhaps non-private way of initializing OPT, this could lead to better results.

Although the main focus was on producing a working algorithm, this research shows even a small and simple task-specific modification leads to improved results. In that sense, this research merely touched on model optimization, because of its focus on a working algorithm, rather than a state-of-the-art one. It could be interesting to continue this line of thought to see the potential of a specialized algorithm based on this work.

As it stands, the DMW algorithm for privately solving LP-SVMs is not that practically interesting by itself. We do, however, hope that the implications of such a working algorithm help to inspire better working private solutions in the fields of machine learning and linear optimization, as our solution can be applied to problem specifications other than Support Vector Machines.

References

Chaudhuri, K., & Monteleoni, C. (2009). Privacy-preserving logistic regression. *Advances in neural information processing systems*, 289–296.

Chaudhuri, K., Monteleoni, C., & Sarwate, A. (2011). Differentially private empirical risk minimization. *The Journal of Machine Learning Research*, *12*, 1069-1109.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Dua, D., & Graff, C. (2017). *UCI machine learning repository.* Retrieved from `http://archive.ics.uci.edu/ml` (last visited on 19-11-2019)

Dwork, C. (2006). Differential privacy. *ICALP 2006. Proceedings, Part II*, 1-12.

Dwork, C., McSherry, F., Nissim, K., & Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. *Theory of Cryptography Conference*, 265-284.

Dwork, C., Rothblum, G. N., & Vadhan, S. (2010). Boosting and differential privacy. *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, 51–60.

Hettich, S., & Bay, S. D. (1999). *The UCI KDD archive.* Retrieved from `http://kdd.ics.uci.edu` (last visited on 19-11-2019)

Hsu, J., Roth, A., Roughgarden, T., & Ullman, J. (2014). Privately solving linear programs. *International Colloquium on Automata, Languages, and Programming*, 612–624.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning. , *112*.

Mangasarian, O. (1998). *Generalized support vector machines.*

McSherry, F., & Talwar, K. (2007). Mechanism design via differential privacy. In *FOCS* (Vol. 7, pp. 94–103).

Song, S., Chaudhuri, K., & Sarwate, A. (2013). Stochastic gradient descent with differentially private updates. *2013 IEEE Global Conference on Signal and Information Processinge*, 245-248.

Vapnik, V. (1982). *Estimation of dependences based on empirical data berlin.* Springer-Verlag.