Differential privacy for learning vector quantization

Johannes Brinkrolf, Christina Göpfert, and Barbara Hammer

CITEC Center of Excellence, Bielefeld University*

This is a preprint of a publication Brinkrolf et al. [17], as provided by the authors.

Abstract

Prototype-based machine learning methods such as learning vector quantization (LVQ) offer flexible classification tools, which represent a classification in terms of typical prototypes. This representation leads to a particularly intuitive classification scheme, since prototypes can be inspected by a human partner in the same way as data points. Yet, it bears the risk of revealing private information included in the training data, since individual information of a single training data point can significantly influence the location of a prototype. In this contribution, we investigate the question how to algorithmically extend LVQ such that it provably obeys privacy constraints as offered by the notion of so-called differential privacy. More precisely, we demonstrate the sensitivity of LVQ to single data points and hence the need of its extension to private variants in case of possibly sensitive training data. We investigate three technologies which have been proposed in the context of differential privacy, and we extend these technologies to LVQ schemes. We investigate the effectiveness and efficiency of these schemes for various data sets, and we evaluate their scalability and robustness as regards the choice of meta-parameters and characteristics of training sets. Interestingly, one algorithm, which has been proposed in the literature due to its beneficial mathematical properties, does not scale well with data dimensionality, while two alternative techniques, which are based on simpler principles, display good results in practical settings.

1 Introduction

Machine learning has revolutionized automated data processing in various domains including medical image processing, decision support, or biomedical data analysis [2, 42, 44]. Common technologies range from highly nonlinear deep networks, which are particularly successful in the context of big data sets and complex functions, to simpler technologies such as support vector machines, random forests, or kernel regression, which are excellently suited if a limited amount of training data is present or the decision boundary is not too complex. Such models aim for an excellent classification accuracy as the most crucial ingredient. Yet, in particular in the case of an embedding into more complex or interactive systems, this objective is accompanied by the desire to generate interpretable, 'white box' models, which enable a human inspection and hence explanation of the system's classification or decision [38]. For the latter, often, generative rather than discriminative models are used [13, 14].

Learning vector quantization (LVQ) constitutes a very popular and intuitive machine learning technology, which represents data in terms of prototypical examples; this way, it

*Funding by the CITEC center of excellence (EXC 277) is gratefully acknowledged.

combines a generative nature of the model in the form of class representatives with a discriminative objective typically used for supervised training [12, 47]. This fact enables its application to domains where model interpretability is aimed for such as biomedical data analysis [7]. Its function can naturally be combined with metric learning schemes such as offered, e.g. by the generalized matrix learning vector quantization (GMLVQ) [46], which allow a direct inspection also of feature relevances [10]. Further, the prototype-based representation opens the way for extensions to intuitive life-long learning strategies [39] as well as classification with reject option, which, in the case of LVQ, provably mitigates the existence of adversarial examples in a dedicated region of the data space [16].

A model representation in the form of exemplars or prototypes, such as present in LVQ, carries a high risk of revealing sensitive information of the used training data: prototypes display typical feature values of the data, since they are constructed as some form of average of a part of the given training data; for simplified settings, the form of the prototypes in stationary states of the learning algorithm can be analyzed exactly. It can be shown that prototypes result as centers of some data points assigned to their receptive field, and metric parameters result as directions which are similar to principal components [11]. Hence it seems likely that, at least in some (possibly extremal) settings of the training data, information about a single individual can be inferred from the models provided suitable auxiliary information becomes available. While such settings can also occur for more complex models such as deep networks [30], a leakage of private information seems almost inevitable for interpretable models such as LVQ due to the fact that training data explicitly impact the location of prototypes to a significant amount and in a very direct way. In particular in the context of highly sensitive domains such as biomedical applications, this risk is clearly not acceptable.

The necessity to preserve a person's privacy in databases has already been debated for more than twenty years [18]. While encryption technologies enable a secure storage of data and hence privacy because data are not available for unauthorized users [34], the situation becomes more problematic whenever important information of the database is offered to the public. Settings, where this happens, include the demand to train a machine learning model based on possibly private data in a distributed network, or the demand to release a machine learning model to the public.

Interestingly, there do exist various frameworks, which go beyond classical encryption and enable public access to private information for specific settings without mitigating any relevant information. One example is offered by homomorphic encryption [5, 40]. Here, encryption schemes are designed in such a way that they commute with arithmetic operations on the data. As a consequence, learning based on these arithmetic operations becomes possible directly from encrypted data without the necessity of a prior decryption of the individual examples. The result of such a learning scheme is an encrypted model. Similarly, in so-called secure multiparty computation, mechanisms to generate a (plain, unencrypted) output out of private data are designed, which ensure that the single (possibly adversarial) user does not get access to any individual information other than those belonging to his-/herself [19, 26].

Yet, while these frameworks enable learning from private information without a direct access to the single unencrypted individual, they cannot prevent the risk that the resulting summary model might reveal critical information if released to the public in unencrypted form. Such leakage can become possible as soon as the model itself is coupled with auxiliary data as available in the internet or dedicated attacks [24]. The question whether such attacks are possible depends on the form of the resulting model and its relation to the individual data. There do exist approaches, which derive bounds on the possible leakage of such models and which suggest to substitute unfavorite ones with privacy preserving surrogates [3].

One particularly popular formal approach, which focuses on the question how to design

functions for model inference such that they do not reveal information about an individual even if coupled with auxiliary information, comes under the notion of *differential privacy (DP)*. This formalism provably limits the possibility to retrieve private information from published models no matter which auxiliary information or attacks are used [23]. Basically, DP formalizes the intuition that the amount of individual information, which is contained in the models, is strictly limited per query. This way, formal guarantees can be given about the immunity of the formalism to auxiliary information and privacy of individual information.

Machine learning (ML) and DP, at a first glance, seem widely incompatible, since ML reveals information from data while DP hides information. Yet, quite some technologies have been developed, which enable an extension of popular ML tools to differentially private counterparts [32]. Often, these rely on mathematical properties of DP such as composition schemes for privacy preserving operations and explicit relations of the sensitivity of ML mechanisms working on data and the resulting degree of privacy [32]. Interestingly, such DP variants have been proposed for specific models including, for example, naive Bayes, linear regression, decision trees, k-means or SVM, and for general training mechanisms such as evolutionary algorithms or gradient schemes for optimization [48, 20, 41, 1, 6]. So far, however, no effort has been done to investigate differentially private LVQ schemes, despite the popularity of LVQ in sensitive application domains [10, 7, 1, 39].

In this contribution, we investigate LVQ schemes, more precisely plain LVQ and counterparts derived from a cost function, Generalized Learning Vector Quantization (GLVQ) and its extension to relevance learning, Generalized Matrix LVQ (GMLVQ), as regards their preservation of privacy. We show in examples that the schemes are locality sensitive and hence run the risk of revealing private information. Due to this fact, we investigate three schemes which provably lead to DP: an extension of gradient-based training to a DP variant together with a DP initialization of the model [1, 6], a general scheme which is based on subsampling and aggregation [20], and a geometric variation thereof which has been proposed due to its better provable mathematical characteristics [43]. We demonstrate that these schemes can be transferred to LVQ. We investigate the behavior as regards the robustness to the choice of meta-parameters and the competitiveness of the resulting model accuracy compared to the standard (non DP) versions. Thereby, data dimensionality will play a crucial role, and it will turn out that the approach as proposed in [43] yields good results for small dimensions but it does not scale with increasing data dimensionality for LVQ. In contrast, the two alternative schemes provide reasonable results provided enough training data are given.

The roadmap of the following sections is as follows: first, we introduce the LVQ schemes we are going to use. In particular, we motivate their cost function and training mechanisms, since these will constitute the base for DP variants of these methods. Then, we introduce the formal notion of differential privacy and the mathematical properties we are going to use in this article. The latter includes general DP schemes based on the addition of Gaussian noise, composition of DP mechanisms, and DP initialization schemes for the models. Before addressing DP variants of LVQ, we will first elucidate the question whether LVQ models are vulnerable to reveal private information given auxiliary data. One condition which yields to vulnerability is the fact that a model is sensitive to single data points in an easily predictable way. From a mathematical point of view, this fact is captured by the so-called local-sensitivity of a learning algorithm, and we will indeed demonstrate that LVQ schemes are locally sensitive. This fact motivates the necessity for DP variants of LVQ if the model should be released to the public. We introduce three different DP schemes for LVQ, which are based on three different generic mechanisms which can be transferred to the specific setting of LVQ. We evaluate these approaches as regards their sensitivity to meta-parameters, to have some clue about which parameters in particular as regards differential privacy to use, and we evaluate their

performance, i.e. accuracy and obtained privacy in benchmarks.

2 Learning Vector Quantization

In this section, we explain the machine learning algorithms we are interested in, variants of LVQ, which are derived from a cost function. These are powerful classification schemes with strong theoretical guarantees as regards their generalization ability [46], for which the dynamics can partially be derived from cost functions, i.e. general convergence guarantees of stochastic gradient schemes apply, or which dynamics has been investigated, e.g. in the framework of statistical physics of online learning [8].

Generally speaking, we are interested in classification scenarios in $\mathcal{D} \subset \mathbb{R}^d$ with k classes which are enumerated as $\{1, \ldots, k\}$. Prototype-based classifiers are defined as follows: labeled prototypes $\mathbf{w}_1, \ldots, \mathbf{w}_w$ with labels $c(\mathbf{w}_j)$ are specified such that a good classification and representation of the data is achieved. A new sample $\mathbf{x}$ is classified by the winner takes all scheme

$$\mathbf{x} \mapsto c(\mathbf{w}_{J(\mathbf{x})}) \text{ where } J(\mathbf{x}) := \arg\min_j d(\mathbf{x}, \mathbf{w}_j).$$

Standard LVQ schemes use the squared Euclidean metric

$$d(\mathbf{x}, \mathbf{w}_j) = (\mathbf{x} - \mathbf{w}_j)^T (\mathbf{x} - \mathbf{w}_j).$$

Given labeled training data $\{(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \{1, \ldots, k\} \mid i = 1, \ldots, n\}$, prototypes $\mathbf{w}_j$ are adapted such that the classification error for the given training set is as small as possible. Since this is in general an NP-hard problem [15, 31], heuristics or approximations of the 0-1-loss are used.

Standard LVQ (referred to as LVQ1) relies on the heuristics of Hebbian learning: given a training point $(\mathbf{x}_i, y_i)$, the winner, i.e. closest prototype $\mathbf{w}_{J(\mathbf{x}_i)}$ is determined and adapted by the rule

$$\mathbf{w}_{J(\mathbf{x}_i)} = \begin{cases} \mathbf{w}_{J(\mathbf{x}_i)} + \eta \cdot (\mathbf{x}_i - \mathbf{w}_{J(\mathbf{x}_i)}) & \text{if } y_i = c(\mathbf{w}_{J(\mathbf{x}_i)}) \\ \mathbf{w}_{J(\mathbf{x}_i)} - \eta \cdot (\mathbf{x}_i - \mathbf{w}_{J(\mathbf{x}_i)}) & \text{otherwise} \end{cases}$$

LVQ does not possess a valid cost function, yet it shows surprisingly accurate behavior in typical model situations [9, 8]. Provided data are separable, its limit positions converge to the centers of gravity of the classes, as has been shown in the articles [27, 11].

There do exist different extensions of LVQ to a cost function, including probabilistic frameworks such as proposed in the approach [47], or deterministic approximations of the classification loss, which also relate to the objective of (hypothesis) margin maximization [45, 46]. For the latter, generalized LVQ (GLVQ), the cost function becomes

$$E = \sum_i \Phi \left(\frac{d^+(\mathbf{x}_i) - d^-(\mathbf{x}_i)}{d^+(\mathbf{x}_i) + d^-(\mathbf{x}_i)} \right)$$

where Φ is a monotonic increasing function, $d_+(\mathbf{x}_i)$ the squared distance of $\mathbf{x}_i$ to the closest prototype of the correct class and $d^-(\mathbf{x}_i)$ the smallest squared distance to another prototype of a different class than $\mathbf{x}_i$. Training takes place based on a given training set, by initializing the prototypes within the class centers and minimizing the cost term E by a simple gradient descent or second order techniques such as LBFGS.

Since the choice of the distance is crucial for the performance of the model, GLVQ has been generalized to metric learning schemes dubbed generalized matrix LVQ (GMLVQ)[46].

Essentially, a positive semi-definite quadratic form $\Lambda = \Omega^T\Omega$ is used to define a generalized squared distance function

$$d_\Lambda(\mathbf{x}, \mathbf{w}_j) = (\mathbf{x} - \mathbf{w}_j)^T\Lambda(\mathbf{x} - \mathbf{w}_j).$$

This distance is then used in the winner takes all function of the classifier and the cost function E. Adaptation takes place with respect to prototypes and matrix parameters Ω via gradient schemes. Thereby, matrix elements are initialized as standard unit matrix corresponding to the standard Euclidean distance. In the following we use the identity $\Phi(\mathbf{x}) = \mathbf{x}$ as activation function, since it has historically proven excellent performance, the sigmoidal nonlinearity which is embedded in the summands due to the fraction being strong enough. Some approaches rely on the choice of Φ as sigmoidal function, which enables a more fine graned tuning of the region of interest along the decision boundary within so-called border-sensitive schemes [33].

3 Differential Privacy

In the following, we briefly introduce the concept of differential privacy (DP). We shortly recapitulate the notion of DP as well as a few popular DP strategies, which will be of relevance for our approach.

Differential privacy The notion of differential privacy [23, 21, 22] constitutes a strong standard for privacy guarantees for algorithms on aggregated databases. Thereby, it addresses the question, which additional information about an individual data point which has been used for training a model can be extracted from the model and any given auxiliary information. One example of such settings occurs, for example, if a model reveals a previously unknown correlation of features for a specific data point which enables an adversary to retrieve one specific feature if he/she knows the other. To avoid such problems, the idea is to limit the amount of individual information which is encoded within a given model.

Informally, DP requires that the output of a data analysis mechanism, such a mechanism to learning a model, remains approximately the same if any sample in the input database is added or removed. This guarantees that a single entry cannot substantially affect the revealed outcome, hence it is impossible to retrieve sensitive individual information from the latter. Now, we define DP first and introduce specific differentially private mechanisms later.

Definition 1 (Adjacent data sets) *Assume two data sets D, D' of data points (e.g. training samples for LVQ) are given. The Hamming distance $d_H(D, D')$ between two datasets is the number of entries on which D and D' differ, i.e.,*

$$d_H(D, D') = |D \backslash D' \cup D' \backslash D|.$$

Two datasets are adjacent if they differ in a single individual: $d_H(D, D') = 1$. We denote adjacency of D and D' as $\operatorname{adj} D, D'$.

Differential privacy limits the way in which the output of an operation such a machine learning algorithm can change if it is subject to adjacent data sets.

Definition 2 (Differential Privacy [23]) *Assume $\varepsilon, \delta > 0$ are given. We are interested in the privacy of an operation $\mathcal{A}$ such as a machine learning algorithm, which maps a given set of training data D to a model or summary statistics revealed to the user. We assume that $\mathcal{A}$ is random variable and its outputs are characterized by the probability measure* P. *$\mathcal{A}$ gives*

(ε, δ)-differential privacy *if and only if for all pairs of adjacent datasets D and D', and all measurable events S in the space of possible models, it holds*

$$\mathrm{P}[\mathcal{A}(D) \in S] \leq e^{\varepsilon} \cdot \mathrm{P}[\mathcal{A}(D') \in S] + \delta.$$

Note that the notion of adjacency is symmetric. Hence privacy with the choice $\epsilon = \delta = 0$ would imply that a single example has no influence on the output of the operation $\mathcal{A}$. Since this is obviously useless (by induction, such an algorithm would not be able to learn anything), small values ϵ and δ are usually aimed for. This notion of DP ensures the privacy of any single sample which can be used for training, because adding or removing this sample results in a very small change of the distribution of possible algorithmic outcomes. Hence it is not possible to observe a significant difference of the output of $\mathcal{A}$ if an adversary is allowed only a small number of observations.

We would like to point out that the notion DP is compositional in the following sense: assume m independent mechanisms $\mathcal{A}_1, \ldots, \mathcal{A}_m$, that satisfy DP for $\varepsilon_1, \ldots, \varepsilon_m$, are given. Then, performing these algorithms consecutively results in a mechanism that satisfies ε-differential privacy for $\varepsilon = \sum_i \varepsilon_i$ [22]. We will call ε the privacy loss of the algorithm.

There are several approaches which satisfy ε-differential privacy, including the *Laplace Mechanism* [23]. The latter deals with algorithms or functions $f : \mathcal{D} \mapsto \mathbb{R}^k$ from the domain of all datasets to vectorial outputs. It adds symmetric and scaled noise to each dimension of the output. The magnitude of the required noise depends on the so-called *global sensitivity* (GS) of f. It refers to the maximum difference between the outputs of two adjacent datasets, or, more formally:

Definition 3 (Global sensitivity) *The global sensitivity of f is defined as*

$$\Delta_{\mathrm{GS}} f = \max_{D, D' : \mathrm{adj}\, D,\, D'} \parallel f(D) - f(D') \parallel$$

measured in any norm $\| \cdot \|$.

Similar to the global sensitivity the *local sensitivity* (LS) can be defined. Here, the maximum difference between one fixed dataset and all adjacent ones is considered. Formally:

Definition 4 (Local sensitivity) *The local sensitivity of f for one dataset D is defined as*

$$\Delta_{\mathrm{LS}} f = \max_{D' : \mathrm{adj}\, D,\, D'} \parallel f(D) - f(D') \parallel .$$

Note, that the GS is the maximum LS taken over all possible datasets. Yet, it can be shown that releasing $f(D)$ with noise magnitude proportional to $\Delta_{\mathrm{LS}}(f)$ is not DP because the noise magnitude itself reveals information about the database [20].

Definition 5 (Laplace mechanism) *Given a function f the Laplace mechanism is defined as*

$$\mathcal{A}_f(D) = f(D) + (Y_1, \ldots, Y_k)^T$$

for a given database D, where Y_i are i.i.d. random variables drawn from the Laplace distribution $\mathrm{Lap}\left(\Delta_{\mathrm{GS}} f / \varepsilon\right)$, whereby the global sensitivity is measured based on the L_1-norm. This distribution is defined by the probability density function $\mathrm{P}[\mathrm{Lap}(\beta) = x] = \frac{1}{2\beta} e^{-|x|/\beta}$. It can be shown that the resulting mechanism $\mathcal{A}_f$ is (ε, 0)-differentially private.

The Laplace mechanism constitutes a very convenient way to turn a given database query into a differentially private one. However, it has only limited applicability if f is given by a learning algorithm since its sensitivity might be complicated to bound. Therefore, more methods which directly rely on typical machine learning mechanisms have been proposed. A very popular one adds differential privacy to gradient techniques.

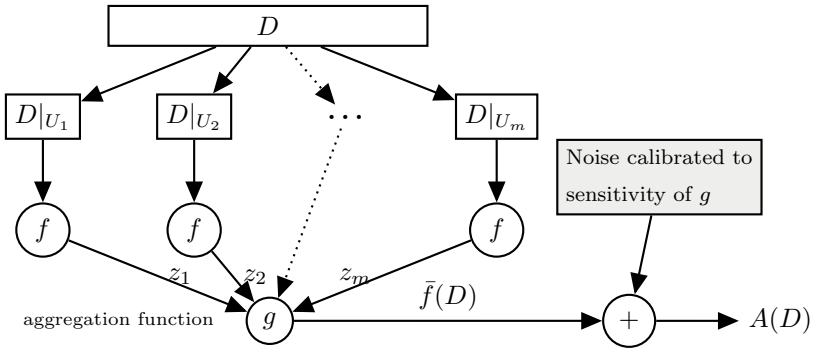


Figure 1: Sample-Aggregate Framework. U_i are random subsets from $\{1, \ldots, |D|\}$ of size $|D|/m$ and $D|_U \subset D$ with indices in U

Differentially Private Stochastic Gradient Descent This method has been introduced by Abadi et al. [1]. Essentially, the mechanism proposes a variant of a gradient descent with private operations only. It assumes that an objective loss function $\mathcal{L}(\theta)$ with parameters θ is given which is optimized to reveal the model parameters θ. The proposed formalism computes the gradient $\nabla_\theta \mathcal{L}(\theta, \mathbf{x}_i)$ of the loss function for each sample $\mathbf{x}_i$ in a random subset of size L, which is taken from the training set of size n with sample probability $q = L/n$. Then, each gradient is clipped whenever its L_2 norm is greater than a threshold C. Adding Gaussian noise drawn from a normal distribution $\mathcal{N}(0, \sigma^2 C^2)$ for each dimension for a specific σ guarantees DP. The results are averaged and a noisy gradient descent according to these directions is taken.

This algorithm reflects mini-batch optimization techniques as are popular for the optimization of non-convex cost functions in machine learning. It has been shown that the resulting algorithm is (ε, δ)-differentially private for any $\delta > 0$, provided $\sigma \in \Omega(q\sqrt{T \log(1/\delta)})/\varepsilon$, where T is the number of steps of the gradient descent method. Pseudocode can be found in [1].

Subsampling and Aggregate The subsample and aggregate framework [22] can deal with functions which posses high global sensitivities or which sensitivities are hard to compute. The basic idea is to randomly divide the dataset into m small batches and apply the function on each subset. Then a so-called aggregation function g is used. This takes the output z_i of all results for $i \in \{1, \ldots, m\}$ and aggregates those with a suitable mechanism. For geometric frameworks such as prototypes, aggregation can be based on geometric averages, for example. To guaranty differential privacy, at this stage, noise, which is calibrated to the aggregation function, is added. This framework is depicted in Fig. 1.

If disjoint subsets are used, one sample can only impact the output of one subset. Hence, the amount of noise is small as compared to the Laplacian mechanism for the full data. Pseudocode can be found in [22].

Center of Attention One variant of the Subsample and Aggregate approach is introduced by Nissim et al. [43]. It is called *Center of Attention*. Essentially, it proposes a specific aggregation function which yields favorable mathematical properties. This aggregation function is computable in any metric space $\mathcal{M}$. The sensitivity of the aggregation function is defined by the so-called *t-radius*. Let Z be set of points $\{z_1, \ldots, z_m\}$ and $z_i \in \mathcal{M}$, e.g., the different outputs for each subset. For a point $c \in \mathcal{M}$ the t-radius $r(c, t)$ is defined by the distance to the t-th nearest neighbor of c in Z. The center of attention $g(Z)$ of Z is then the point

in Z with its smallest t_0-radius, where $t_0 = \lceil (m + \sqrt{m})/2 \rceil$. It can be shown that adding Gaussian noise proportional to $\max_{l \geq 0} r^{(Z)}(t_0 + (l+1))$ to the center of attention is sufficient to guarantee privacy, where $r^{(Z)}(t)$ is the minimum t-radius of any point in Z. This can be calculated easily and efficiently by all pairwise distances and sorting these distances for each point in Z.

4 How vanilla LVQ violates privacy

Before introducing DP variants of LVQ, we want to demonstrate the necessity to do so. Essentially, a learning algorithm is not private if adding or removing one example significantly changes the output distribution of the algorithm: provided an adversary can gain information from a typical distribution (e.g. by inference on publicly available data), information of the added data point can leak. We will demonstrate, that LVQ is prone to at least two qualitatively different changes of the outcome which can be caused by adding a single data point, namely significant changes of the distribution of the prototype position, and significant changes of the data assignment to prototypes.

Leakage due to prototype positions for LVQ LVQ and its variants have been shown to provide high quality predictions. Unfortunately, trained models carry the risk of exposing private data used for training. To see why and in which settings this is the case, let us first consider data sampled from a well separated mixture of two Gaussians and LVQ1. As has been shown in [27, 11], in this case, costs are optimal when the prototypes lie at the mean of their respective classes. This fact enables us to argue purely analytically, since the stationary states of the LVQ1 are given explicitly: LVQ prototypes leak information in the same way that mean value statistics leak information [20]. In particular, if an adversary knows the coordinates of all points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ from one class except for one point $\mathbf{x}'$ as well as the corresponding prototype $\mathbf{w}$, he/she is able to perfectly reconstruct $\mathbf{x}'$ via the formula

$$\mathbf{x}' = (n+1)\mathbf{w} - \sum_{i=1}^{n} \mathbf{x}_i.$$

Even if the adversary does not have access to the points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ themselves, if he/she has access to prototypes $\mathbf{w}_1$ and $\mathbf{w}_2$ trained on data sets that differ only on one record $\mathbf{x}'$, the adversary can recover $\mathbf{x}'$ via the formula

$$\mathbf{x}' = (n+1)\mathbf{w}_2 - n\mathbf{w}_1.$$

This means that there do exist settings where auxiliary information leads to the leakage of details about a single data point.

Leakage due to prototype positions for GLVQ Later versions of LVQ, such as GLVQ adapt the cost function such that prototypes are not only attracted by points from their own class, but also repelled by points from other classes. In addition, GLVQ gives more influence to points that are close to a decision boundary. These changes make the cost function less tractable, in particular analytic solutions of stationary states are not possible, which prevents obvious exploits such as the shown above. However, single outliers still have a significant effect on the resulting prototypes, as can be demonstrated experimentally. This means, that the algorithm is highly sensitive, and there exist situations where the behavior of the algorithm can reveal insights into the characteristics of one data point.

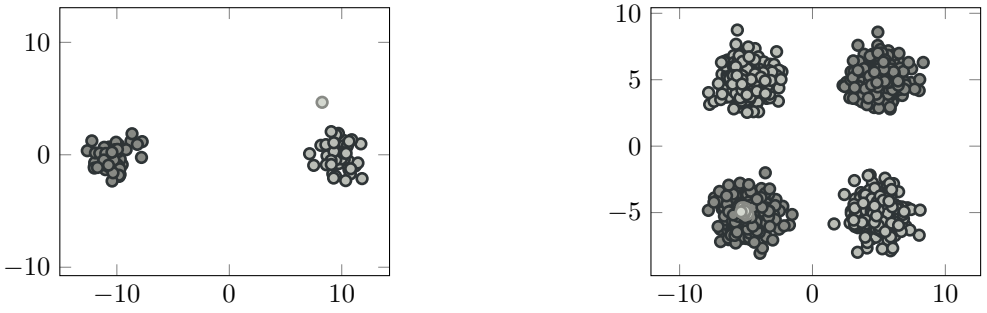
To illustrate this point, we have performed statistical test according to the following setups (see Fig. 2a):

- *Extreme outlier:* we generate one data set with 100 points from two classes, 50 per class, with points sampled from Gaussian distributions with means $(-10, 0)$ and $(10, 0)$, respectively, and variance 1. As second data set, we create a copy of the data set and add one outlier sampled at mean $(10, 10)$ with variance 1. We sample these dataset pairs 400 times and observe the position of the prototype from the outlier class on the y-axis. On average, the prototype trained with the outlier moves upwards by 0.1119. A statistical test shows that the prototypes have different positions on the y-axis with p-value of 5.6776×10^{-25}. Hence the prototype distribution significantly changes in this setting, thus revealing information about a single point by an observation of the output distribution of the learning algorithm.
- *Medium outlier:* This extremely pronounced effect is due to the big distance of the outlier and the class mean. We repeat the same test but reduce the distance of the outlier and set its mean to $(10, 5)$. Then we still observed a shift of the prototype on the y-axis for the second class by 0.0843 with p-value 2.2345×10^{-16}.
- *Weak outlier:* setting the outlier mean to $(10, 2)$ results in a shift by 0.0368 and a p-value of 4.4007×10^{-4}.

Naturally, there exists a phase transition as soon as the outlier approximates the Gaussian cluster. Yet in all three cases described above, a significant shift of the output distribution is observed. Thus, an adversary can use the resulting model to learn about the presence of an outlier, its direction and possibly even its magnitude.

Leakage due to insensitivity of GLVQ output The fact that the vanilla GLVQ algorithm leaks information is not necessarily surprising since a useful algorithm always exhibits some degree of sensitivity to its input. A natural cure in the differential privacy framework, as proposed by the Laplacian mechanism, is to add noise proportional to the global sensitivity of the learning algorithm. As we will see, however, this is not a feasible solution in the case of GLVQ since GLVQ can exhibit an extremely high sensitivity even on large data sets, due to another effect of the learning algorithm in settings where there exists a mismatch of the prototypes and the underlying modality of the data distribution, the algorithm needs to distribute the prototypes among the data. For perfectly balanced data distributions, this results in a symmetry breaking of the algorithm. This symmetry, i.e. two different prototype locations which are regarded as equally good by the algorithm, can be disturbed by adding few additional points. We will show that this is the case using the noisy XOR problem (see Fig. 2b):

- *XOR:* the data is generated by four Gaussians, one in each quadrant of the coordinate plane, where the top left and bottom right cluster belongs to class number 1 while the bottom left and top right clusters belong to class number 2. We generate data sets according to the following experimental conditions: 1200 points in total (300 in each cluster) sampled randomly at means $(10, 10)$, $(-10, 10)$, $(10, -10)$ and $(-10, -10)$ with standard deviation 1. This is the "balanced" condition, the first data set. As second data set, we create a copy of this data set and add 10 points in the bottom left cluster. This is the "unbalanced" condition. We train an LVQ algorithm with two prototypes for class 1 and a single prototype for class 2. 400 test runs on instances of each condition show that running GLVQ on the "balanced" condition, results in one prototype in each cluster of class 1 and a random assignment of the prototype of class 2 to one of the two clusters with probability 50% each over a random initialization of the setting. In contrast, running GLVQ on the "unbalanced" condition always results in the prototype for class 2 being assigned to the bottom left cluster.



(a) Two clusters with a single outlier.

(b) Asymmetrical XOR.

Figure 2: Two settings with potential privacy leaks: for the toy settings pictured in 2a and 2b, we compare GLVQ results on the data sets comprised of only the dark points to results on the data sets augmented by the light gray points.

Hence, symmetry breaking of optima of the algorithm might depend on very few samples added to the classifier, independent of the size of the overall training set. This carries the risk of revealing significant information about the position of these points provided the adversarial has knowledge about the existence of such symmetries in the solution of the model.

This analysis demonstrates two aspects: on the one hand, the result of LVQ is sensitive to single data points and there do exist settings where a leakage of private information is possible, provided an adversary has access to auxiliary information. On the other hand, it seems that GLVQ cannot easily be combined with differential privacy mechanisms that rely on the global sensitivity of the algorithm itself, since, as shown in the last example, the global sensitivity can not easily be limited with non-trivial bounds. One trivial bound would be the domain size, but adding noise of this scale would render a resulting model essentially useless. Due to these observations, we will now evaluate mechanisms for differential privacy that do not rely on privatizing a single, already trained model. Instead, we will look at privatizing optimization as well as using a sample and aggregate mechanism to exploit well-structured data sets that have low local sensitivity.

5 Ensuring privacy for LVQ

We can transfer both methods, a DP stochastic gradient descent [1, 6] as well as subsampling and aggregate to GLVQ [20, 43]. For the latter, we need to define a suitable way to aggregate sets of prototypes as delivered by LVQ for subsets. We will use two different aggregation methods. For the first, we just need to investigate a suitable level of noise for GLVQ, the second relies on geometric considerations.

In the following, we will restrict to LVQ models with one prototype per class only. This is clearly a limitation of the setting, albeit there do exist quite a couple of applications where one prototype per class is sufficient [46, 7]. The restriction of our analysis to one prototype per class is due to the circumstance that a few operations such as initialization and merge of prototypes are much easier for this setting, hence easier to exchange by DP variants. We will remark on this fact wherever it is relevant and give a hint how DP variants could be extended to multiple prototype scenarios, but we leave the details to future work since it would go beyond the scope of this article.

5.1 Ensuring privacy via differential private SGD

In the following we describe how we change the gradient training of a GLVQ model to obtain a DP variant. Since this approach addresses the training scheme rather than the output only, i.e. the prototypes, this mechanism could also be used to extend GMLVQ to a DP variant. Essentially, the scaled stochastic gradient descent used for GLVQ is substituted by a DP variant as already introduced in Section 3. We refer to this method as **SGD**.

First step – Initialization of prototypes: As indicated above, we assume that we use one prototype per class, which are initialized by the class centers in vanilla LVQ. For vanilla LVQ, class centers are calculated based on the sum of all samples of each class and the number of class members: $\mathbf{w}_j = 1/N_j \sum_{i:c(\mathbf{x}_i)=j} \mathbf{x}_i$ for each class j. These operations can directly be enhanced to DP versions based on the Laplace mechanism as follows:

- *Cardinalities of classes:* The cardinalities of the classes are given by the function $f : \mathcal{D} \mapsto \mathbb{N}^k$, $f(D) = (N_1, N_2, \ldots, N_k)$. This function has a sensitivity $\Delta f = 1$ because adding or removing one data point in the dataset changes only the output of one N_i by one. In the literature, these functions are also known as histogram queries [22]. Hence the Laplace mechanism according to the sensitivity 1 can render this computation DP.

- *Sum of points:* The sum of all points in each class is given by the function $g : \mathcal{D} \mapsto \mathbb{R}^{k \cdot d}$, $g(D) = \left(\sum_{i:c(\mathbf{x}_i)=1} \mathbf{x}_i, \ldots, \sum_{i:c(\mathbf{x}_i)=k} \mathbf{x}_i\right)$. Without loss of generality, we assume that the samples are normalized such that $\mathcal{D} \subset [-1, 1]^d$. Then, the sensitivity of the function is $\Delta g = d$. One adjacent dataset can change the output at least by one in each dimension in the L_1 norm because the classes are disjoint sets.

- *Composition of the two functions:* For a given privacy loss ε_1 we obtain all N_i and all sums with the Laplace Mechanism in a differentially private way. We use the noise scales $\beta_f = 2/\varepsilon_1$ for the function f and $\beta_g = 2d/\varepsilon_1$ for g. Then we achieve a ε_1-differentially private mechanism altogether due to standard arguments for composition.

Note, that the noise which needs to be added in this Laplacian mechanism does not depend on the number of samples in the dataset. Hence, it has a smaller impact on big ones and a higher if it is getting smaller.

This DP initialization is restricted to one prototype per class, since explicit analytic formulas exist in this case. For more prototypes per class, different initialization schemes are popular, such as an initialization by neural gas, for example [29]. Since an analytic solution of the stationary stated of neural gas does not exist, DP variants of neural gas could be obtained, e.g. by DP variants of its gradient scheme.

Gradient descent: For the gradient descent, we rely on the algorithm as described in chapter 3 by Abadi et al. [1]. Essentially, a batch gradient with stochastic noise is done in the following way:

- choose a random subset of size L
- compute the gradient of this mini-batch, and clip each single gradient to L_2 norm at most C
- add sufficiently large Gaussian noise per dimension (where the variance can be computed based on C and desired degree of privacy according to [1]).

Let L be the batch size, C a bound for the norm of the gradient, $q = L/n$ the sample probability for one sample, E the number of epochs and $T = E/q$ the runs of the gradient

descent and the number of updates. For GLVQ we just have the gradients of the prototypes which we have to clip. In the case of GMLVQ, the parameters of the projection matrix Ω would also be clipped together with the parameters for the prototypes in the L_2 norm. For a given ε_2 and δ we can calculate the noise scale by $\sigma = 2q\sqrt{T\log(1/\delta)}/\varepsilon_2$.

DP bounds for the full scheme: Hence, the total privacy loss of the whole training, i.e. initialization of prototypes followed by gradient-based optimization, is $\varepsilon = \varepsilon_1 + \varepsilon_2$ due to the composition theorem of differential private mechanisms. We obtain an (ε, δ)-differential private algorithm.

5.2 Smooth sensitivity – mean value

As an alternative, we investigate the possibility to guarantee DP by a suitable subsampling and aggregation as introduced before [22]. The overall procedure is as follows:

- decompose the training data randomly into m disjoint subsets (bins)
- perform GLVQ on each bin
- aggregate the prototypes as follows: since there exists only one prototype per class, we can simply rely on the class-wise prototype mean value. If more than one prototype per class would be used, we would need to add a matching step beforehand, minimizing their pairwise distance.
- guarantee that the result is DP by using the Gaussian Mechanism

We need to ague how to choose the variance for the Gaussian Mechanism: averaging is the composition of the sum of prototypes and dividing by the number of bins. Since the latter is fixed, we need to make only the first operation DP by adding Gaussian noise to each dimension. The size of the Gaussian noise can be chosen based on the following estimate of the global sensitivity of this operation: The bins are disjoint subsets and therefore, adding or deleting one sample to the data set affects the output of at most one bin. Hence the GS of computing the sum can be bounded by $\sqrt{d}$ when measured in the L_2 norm. We use the Gaussian mechanism as proposed by Balle and Wang to compute the required variance [6].[1] This method is referred as **SA**. Note that the aggregation scheme can also be used to derive DP variants of algorithms without explicit cost functions such as LVQ1.

5.3 Smooth sensitivity – center of attention

For the method SA as introduced above, binning and aggregation are done randomly and the GS is taken. As an alternative, Nissim et al. propose a mechanism, dubbed center of attention (**COA**), which benefits from the local geometric setting and the LS [43]. In contrast to the algorithm SA, the type of binning and aggregation is changed as follows:

Binning: We use m subsets with size n/m, which are chosen randomly from the n training data, each one chosen uniformly without replacement. Note that with probability at least $1 - 2^{-\sqrt{m}+\log n}$ no points occurs in more than $\sqrt{m}$ subsets [43], but the bins are not disjoint. Hence with a high probability, any sample from the training data affects the result of at most $\sqrt{m}$ bins. Otherwise, we re-sample the bins as it is suggested by Nissim et al. [43].

Aggregation: The resulting prototypes per subset are matched by their class label. Aggregation then takes place by means of the center of attention as described in Section 3. It has been shown in [43] that adding noise calibrated by the magnitude of the *smooth sensitivity* of

[1] Interestingly, their mechanism extends applicability also to the region ($\varepsilon > 1$).

dataset	DP $\varepsilon = 0.75$	DP $\varepsilon = 1.5$	DP $\varepsilon = 2.5$	non priv. SGD	non priv. BFGS
MNIST	0.1893 (0.0042)	0.1871 (0.0020)	0.1871 (0.0020)	0.1857 (0.0022)	0.1853 (0.0018)
	0.2188 (0.0162)	0.1721 (0.0067)	0.1673 (0.0033)	0.1583 (0.0031)	0.1484 (0.0021)
Motion	0.1121 (0.0061)	0.1123 (0.0063)	0.1121 (0.0058)	0.1112 (0.0062)	0.1111 (0.0062)
	0.1116 (0.0074)	0.1048 (0.0080)	0.1038 (0.0057)	0.0914 (0.0068)	0.0897 (0.0066)
Segment	0.4793 (0.0779)	0.1792 (0.0152)	0.1635 (0.0124)	0.1458 (0.0133)	0.1458 (0.0132)
	0.2642 (0.0432)	0.1745 (0.0205)	0.1696 (0.0233)	0.0932 (0.0108)	0.0870 (0.0109)

Table 1: Mean and std. dev. in brackets for test error rates. As a baseline, the results of a non-private training with SGD and a BFGS optimizer are given. The first rows for each dataset are results for GLVQ the second for GMLVQ.

the aggregation function g gives differential privacy. Here, the smooth sensitivity is an upper bound defined in [43] of the local sensitivity, which does not share its drawback of possibly revealing private information due the size of the noise. It can be computed based on the pairwise distances of the outputs from each subset. In particular, if the outputs are very similar, a small variance for the Gaussian distribution can be used even if the global sensitivity is big. We will refer to this approach as COA.

6 Experiments

We provide experiments for the three methods, SGD, SA, and COA. For the first, SGD, experiments for GLVQ as well as GMLVQ are possible, for the latter two, only GLVQ can be extended since it is not clear how to sensibly accumulate metric parameters. We will evaluate all setups for the same real world data sets. In addition, we will also evaluate the latter two frameworks for theoretical data sets, since this will enable us to explain the unexpected behavior of COA for the real data.

6.1 Differential privacy for gradient based GLVQ

Data sets: We test our approach with three real world datasets, MNIST [35], Motion Tracking [4] and Image Segmentation [36]. The first has 70.000 instances with pictures of handwritten digits. The second one consists of 10.299 samples of recorded accelerator data by a mobile phone, classes are different motion categories. The last one consists of 2.310 image patches of small landscapes which are characterized by has real-valued image descriptors.

Experimental setup: For all settings, evaluation is done based on a 5-fold repeated cross-validation with five repeats. The total privacy loss is split into $\varepsilon_1 = 0.2\varepsilon$ for the initialization step and $\varepsilon_2 = 0.8\varepsilon$ for the parameter optimization. The other parameters are chosen as $\delta = 10^{-5}$, $q = 0.01$, $C = 0.5$ and $E = 50$. These parameters are chosen in such a way that reasonable results arise. We will see in experiments, that the results are robust within a region around these metaparameters.

We compare the error rates of the results to the results of vanilla GLVQ and GMLVQ. Since our aim is to provide DP variants of LVQ rather than arbitrary classifiers, we do not compare to alternative DP classification schemes other than the one introduced above. Note that the performance of vanilla LVQ itself in comparison to alternatives has been subject of experiments in the literature [28]. For vanilla GMLVQ or GLVQ, the optimum is found by a standard stochastic gradient descent (SGD) and, in comparison, by the LBFGS algorithm, a quasi-Newton method for solving nonlinear optimization problems [25]. For MNIST, due to its size, we always use one fold for training and four as the test set. For Image Segmentation and Motion Tracking, we use four folds for training and one fold as test set.

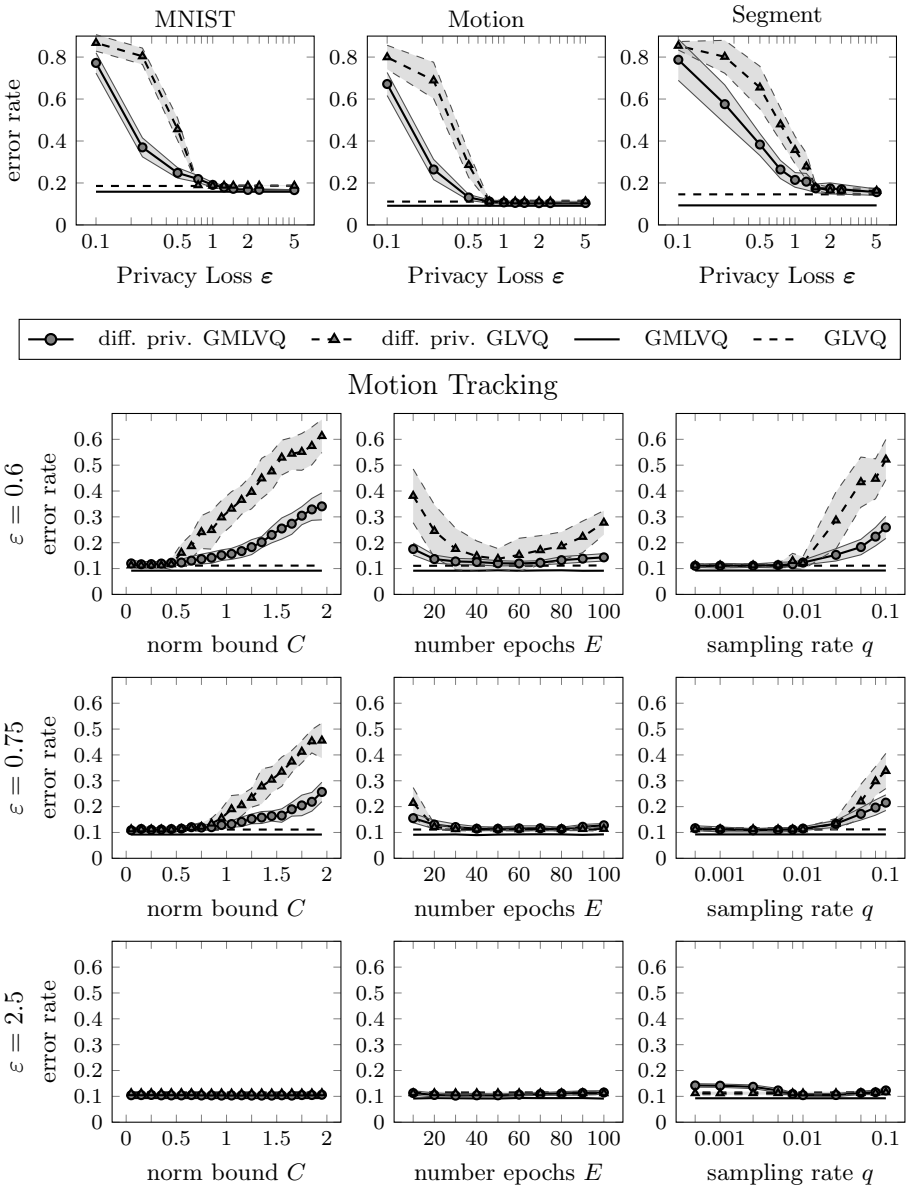


Figure 3: Averaged GLVQ and GMLVQ test error rates for SGD and in vanilla form on three benchmarks. Only the privacy loss varies and all other parameters are fixed ($C = 0.5$, $E = 50$ and $q = 0.01$).
For the Motion Tracking dataset the averaged test error rates are also plotted with different hyperparameters for different privacy losses. $\varepsilon = 0.6$ in the second, $\varepsilon = 0.75$ in the third and $\varepsilon = 2.5$ in the fourth row.

Evaluated quantities: In Fig. 3, results for different privacy losses are shown for all datasets. Furthermore, the error curves of the hyperparameters (C, E and q) for different values of $\varepsilon \in \{0.6, 0.75, 2.5\}$ are shown in the second, third and forth row in the same plot, respectively. The solid lines are the results for GMLVQ and the dashed for GLVQ. Note that our objective is to obtain settings where the priorly chosen privacy loss is as small as possible to ensure higher privacy. Likewise, we still want to attain a classifier with similar error rates as the vanilla LVQ variants.

Lower accuracy for higher privacy: One can see that the privacy loss affects the classification strongly in some regions and the curves fall sharply at a certain point. For the datasets the points vary between $\varepsilon = 0.75$ for Motion Tracking and $\varepsilon = 2.5$ for Segmentation. This is due to the higher effects of the noise on smaller datasets or smaller lot sizes.

Sensitivity with respect to hyperparameters: The choice of the other hyperparameters has a strong influence in critical regions for epsilon only, being rather robust for ϵ larger 2. As an example, for Motion data we test ten different values for $C \in [0.05, 2]$ and $E \in [10, 100]$ and 20 for $q \in [0.0005, 0.1]$. Very similar curves could be also observed for the other two datasets. If one compares the second and the third row (i.e. $\varepsilon = 0.6$ and $\varepsilon = 0.75$), where the privacy loss is changed only a bit, the impact is clearly less. In the last row ($\varepsilon = 2.5$) one can not see any variances.

Rationale for hyper-parameter choice: Recall that the noise is drawn from a normal distribution with $\sigma \sim Cq\sqrt{T \log 1/\delta}/\varepsilon$. Hence, it is clear that the model accuracy becomes worse if this variance is set to a higher value than necessary. At the same time, if the clipping parameter C is too small, the averaged gradient may point in a different direction compared to the true gradient. Here a choice up to $C = 0.5$ seems reasonable. Regarding the number of epochs, too few iterations can lead to a premature stopping before convergence, while too many steps lead to an accumulation of the noise in each step, which changes the overall gradient too much. Hence, a medium number of epochs such as 50 seems suitable. The sampling rate directly influences the required amount of noise. Here, we achieve good results for small values $q \leq 0.01$. Note, that a lower bound for q occurs naturally due to the fact that more than one sample is selected for random subsets. Hence, all hyperparameters are set accordingly to the sweet spots as observed in these experiments.

Size of privacy loss: Naturally, the privacy loss ε should be as small as possible. Yet, it cannot reach zero less since this would render learning impossible. Which privacy loss is acceptable in practice? This depends very much on the application at hand, and it can be observed that a larger data set enables better privacy in general. The privacy loss which we are able to obtain compares favorably to other approaches in the literature such as presented e.g. in the work [43], since we need less iterations for convergence and therefore a much smaller ε. In the work [43], a neural network with 1000 hidden units is presented with privacy loss up to 8 for the MNIST dataset and 750 epochs. Due to the choice of the model as a comparably complex one, however, the accuracy as presented in [43] is higher (up to 98%).

Overall performance of gradient based DP schemes for LVQ: In Tab. 1 the means and the standard deviations of the error rates for all three benchmark sets are listed. For GLVQ we often get trained models which are almost as good as the non-private ones. For GMLVQ the BFGS optimization finds better parameters than SGD. Here, the private versions face difficulties due to the noise in the relevance matrix. The result is very sensitive to matrix parameters and even small changes in the values of the matrix can cause a worse classification. To experimentally test the matrix sensitivity in this case, we add normally distributed random numbers with variance $\sigma = 0.025$ on each element of the relevance matrix. We observe an increase of 0.0277 ± 0.001 (from 0.1484 to 0.1761) of the error rate for the original GMLVQ approach and the MNIST dataset. For the Motion dataset the error increases by 0.0234 ± 0.0085 using

the same settings. Hence, alternative schemes would be beneficial which particularly tackle stable DP variants of matrix adaptation.

6.2 Sample and Aggregate

For the sample and aggregate mechanisms, we evaluate the behavior of GLVQ only. We test both aggregation functions on artificial data sets with well separable data first to get some insight into their behavior. Afterwards, we evaluate the methods on the same three benchmark datasets as above.

Artificial data The artificial datasets are generated by three multidimensional Gaussians with the identity as covariance matrix and (5, 0), (−5, 0), and (0, 5). For higher dimensions, we choose accordingly scaled unit vectors as centers for the Gaussians. All clusters have 3000 samples, so the dataset consists of 9000 samples. Obviously, classes are well separable.

In Fig. 4 the GLVQ test error rates are plotted for different values of the privacy loss and different values of the number of bins in the first two rows. Again, the curves fall sharply at a certain value of privacy loss. For SA a privacy loss equal to 1.25 is sufficient for all three datasets if m is set to 50. For COA a smaller value of m gives better results but it is getting worse if the dimension rises due to the bigger pairwise distances of the prototypes for the bins. As a consequence, the required level of noise increases, making the method unsuitable for higher dimensionality.

This effect is surprising, since COA comes with quite strong formal guarantees. So let us look more closely into this different behavior of COA and SA: If the number of bins is bigger the output of the models trained on the bins varies more. SA simply calculates the mean of all prototypes which is a good approximation because of the robust geometric properties of LVQ. Since the variance is proportional to $1/m$ the added noise is smaller for bigger m. For COA the noise is scaled by twice the minimal t-radius. Here, the noise which is added to the center of attention becomes larger if the dimensionality increases. We observed that, de facto, the noise shifts prototypes outside the cluster, hence the results get worse.

Real-Word data Fig. 5 shows the error rates for the three real-world datasets. The number of bins is set to 15 and 50 for COA and SA, respectively. For the tested privacy losses, COA does not provide any useful solutions, which can be attributed to the problems of higher dimensionalities, as just discussed. SA provides solutions with privacy loss larger than 2, whereby the required noise is proportional to the data dimensionality. Albeit the result of COA are reasonable, SGD yields better performance, hence the SGD technology seems better suited to make LVQ private in realistic settings.

7 Method comparison

In the previous sections, we have adapted, implemented and tested three different methods for ensuring differential privacy of GLVQ. Now we provide a structured comparison, which is somewhat higher level, to shed some light onto important aspects of the proposed models. We will compare the methods according to the categories

1. effectiveness,
2. ease of use,
3. computational issues and

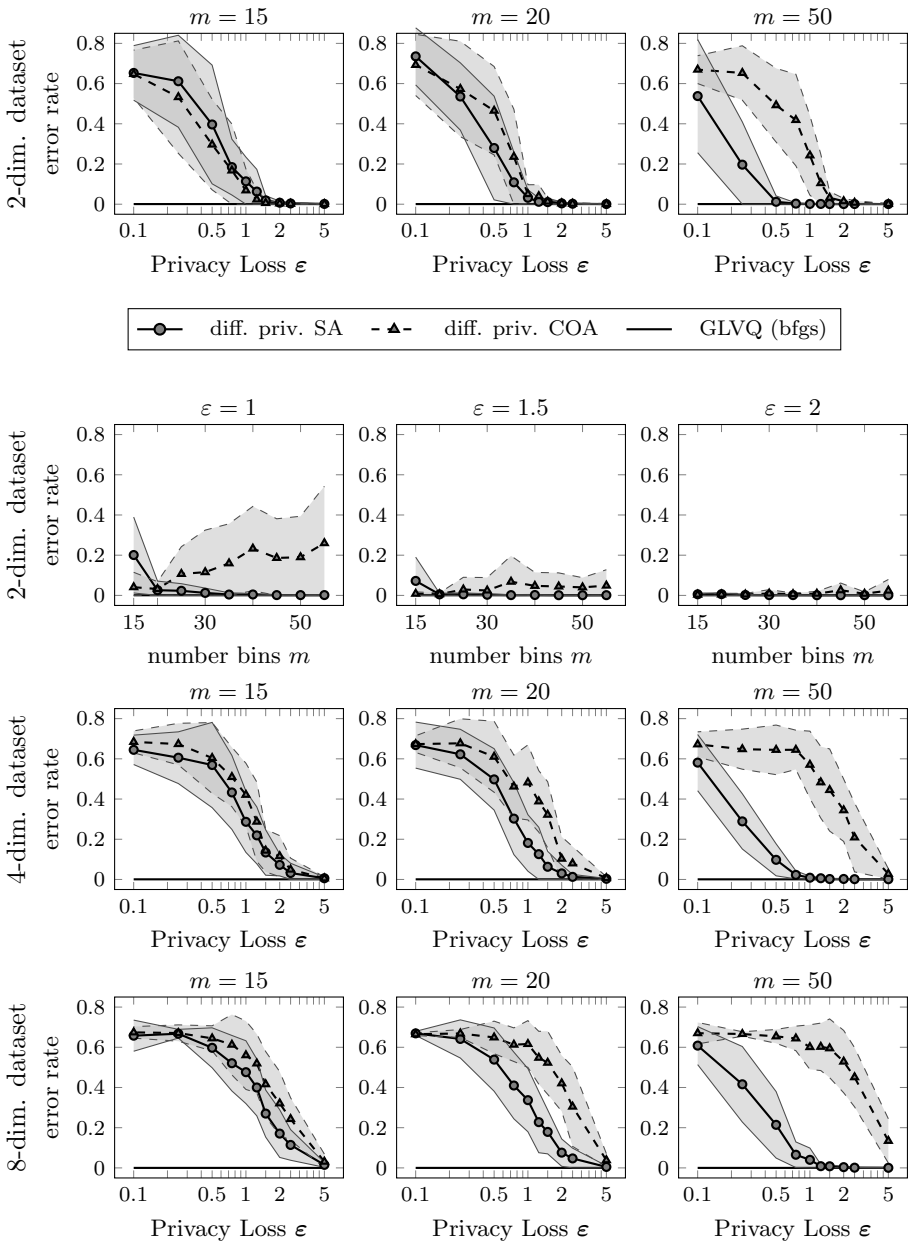


Figure 4: Averaged GLVQ test error rates for artificial datasets with different dimensions (2 in the first, 4 in the third and 8 in the forth row) and different values of the numbers of bins ($m \in \{15, 20, 50\}$). In the second row the curves for the parameter m for three privacy losses is shown.

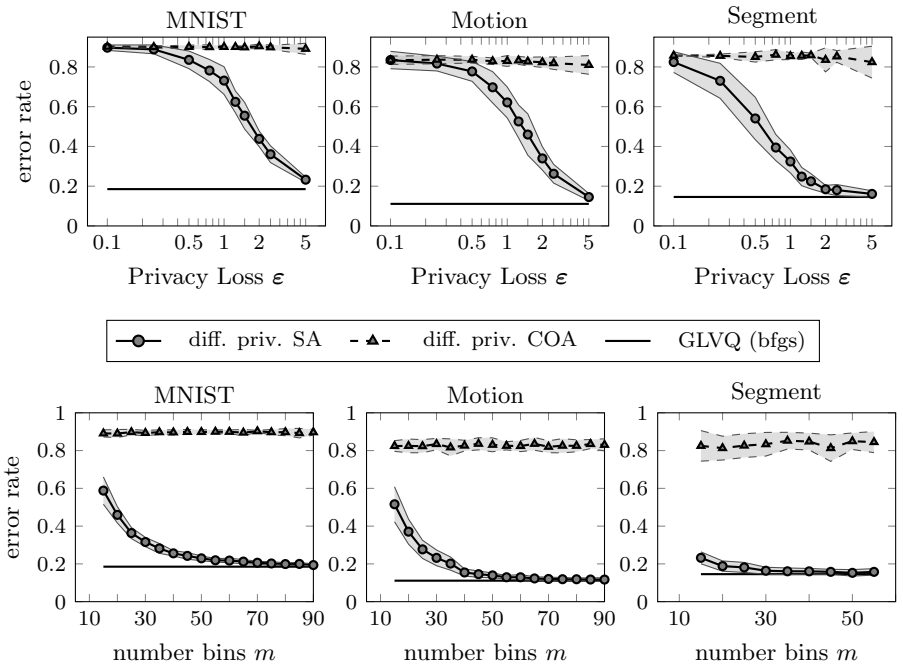


Figure 5: Averaged GLVQ test error rates for different privacy losses for our approach and the non-private version with BFGS optimization on the three benchmark datasets. In the second row the impact for the parameter m is shown for a privacy loss of 5.

4. guarantees.

Effectiveness

Our experiments show that, for realistic data sets, SGD gives generally good classification accuracy for privacy loss ranging from 0.75 to 2.5. SA yields acceptable results for privacy loss of 2 or 5, respectively, while COA does not yield acceptable results for realistic data for the tested epsilon values, which can be attributed to the high dimensionality of the data. Generally, SA is more suited for large data sets with stable results on the single bins, especially well separated datasets. Since the variance of noise added to the mean in the sample and aggregate framework is inversely proportional to the number of subsamples, the noisiness of the differentially private estimate can be greatly reduced when a data set allows for a large number of subsamples. On top of this, the COA framework adaptively reduces the noise added when the underlying data set exhibits a small smooth sensitivity, meaning that small noise is added to large and good-natured datasets - a setting which is seemingly rare for real life data sets with dimensionality larger than 10. Moreover, when LVQ is not stable on subsets, either because of properties of the data itself or, as in the XOR-Example in Section 4, because of a mismatch between data and model parameters, SA and COA can both become very noisy. In addition, sample and aggregate is not easily applicable to other LVQ variants: GMLVQ, for example, would require an averaging of the relevance matrix, which requires careful thought lest it provide nonsensical results. SGD does not adapt the noise added to the size of the datasets or the stability of GLVQ on subsets of the data. However, through rigorous accounting of accumulated privacy loss, it achieves good results even for small ε and can easily be adapted to any other LVQ method which is based on a gradient descent.

Ease of use

Implementing SA for GLVQ requires only a few lines of code. The only parameter that requires tuning (apart from ε and δ), is the number of subsets m. In contrast, implementing COA requires a more complicated aggregation procedure and involves more parameters, e.g. the choice of radius t_0, the smoothness parameter β, the admissibility parameter α of the noise distribution and the number of subsets s that may contain the same point. While [43] gives recommendations for most of these settings, these did not yield reasonable accuracies for our settings. Implementing SGD also requires only a few lines of code. It requires the choice of the batch size which corresponds to the sample probability, the clipping threshold and the number of epochs.

Computational issues

Let n be the number of points in the data set. Let m be the number of training subsets for SA and COA. Then, calculating sample and aggregate requires m GLVQ runs on n/m data points each, with complexity $n/m \cdot k \cdot$ number of iterations per LVQ. These runs are completely independent and can thus be easily parallelized. SA requires computing an average over m GLVQ models. If there is more than one prototype per class, a matching between the prototypes can easily be implemented (e.g. as maximum flow problem per class). COA also requires m GLVQ runs on n/m data points each, and may also require the finding a matching between prototypes if there is more than one per class. Instead of averaging, COA requires finding the model with minimum t_0-radius, which takes $\mathcal{O}(m^2 \log m)$ time. In our experiments, we did not report actual run times, since, in practice, SGD essentially has the same complexity as a standard minibatch gradient descent. COA and SA are faster than classical batch GLVQ with LBFGS, since they act on smaller minibatches. All experiments are manageable on a standard desktop computer, e.g. training COA or SA for the MNIST data on a Intel® Xeon E5-1620 v3 with 3.5GHz and 16 GB RAM takes about 7s.

Guarantees

Per design, all methods guarantee differential privacy with a certain loss. Yet, to the best of our knowledge, there are no results which allow us to bound the noise which need to be added by SGD or SA. [43, Theorem 4.2] gives bounds for the noise added by COA for data sets that are sufficiently good-natured. It is shown that if the optimum found by GLVQ on D with range of diameter S can be approximated within accuracy r using samples of size n/m, then the difference between the optimum found by GLVQ on D and the differentially private result has expected magnitude

$$\mathcal{O}\left(\frac{r}{\varepsilon}\right) + \frac{S}{\varepsilon} \exp\left(-\Omega\left(\frac{\varepsilon\sqrt{m}}{d}\right)\right)$$

in each coordinate. It is not clear whether there exist conditions on a dataset that ensure that GLVQ can be approximated within accuracy r on subsamples. However, it can be shown that under certain separation conditions on D, LVQ1 with one prototype per class can be approximated within accuracy r on D, so that the cited theorem provides noise bounds in this case. However, if GLVQ is unstable under subsampling of D, the results given by center of attention may differ markedly from the optimal solution, as we have observed in experiments.

8 Conclusions

We have investigated technologies which enable an extension of LVQ schemes to variants which respect differential privacy. As we have shown, LVQ, by its very design, has a high risk of revealing private information if used in its vanilla form. This is due to at least two different effects: prototypes which represent the model are essentially close to centers of data points hence, similar to the privacy risks of the mean, outliers can possibly be detected. Second, the result of LVQ is not stable in particular in settings where the number of prototypes does not match the inherent modality of the data distribution and several symmetric optimal solutions exist. Also in these settings, information about few points can possibly leak since few points can break those symmetries. Turning LVQ schemes into DP variants faces the challenge that, according to the possibly large sensitivity of LVQ schemes, different mechanisms are needed which focus on batches and their suitable aggregation. The latter can be either incorporated into training itself, i.e. a mini-batch gradient descent, or it can be used for subsequent aggregation. In both settings, noise can limit the information which can be uncovered as concerns a single data point from the final result.

Depending on the accepted privacy loss, these technologies yield acceptable results, and the sensitivity as regards meta-parameters can be controlled. Interestingly, a simple aggregation seems better suited in particular for high dimensional data sets in comparison to a more advanced aggregation based on the center of attention. Further, gradient based techniques provide convincing results.

These mechanisms are first steps towards learning schemes for LVQ which enable the release of models also in areas with highly sensitive data such as biomedical models or activity profiles. At present, the proposed mechanisms are yet limited to single prototypes per cluster, but extensions would be possible based on more general initialization or aggregation, respectively. Further, DP metric learning yet constitutes a challenge due to its high sensitivity to noise.

It can be expected that the notion of privacy goes hand in hand with a (small) loss in accuracy of the model, since random noise rather than dedicated information is added, which typically goes beyond the amount of noise used to regularize machine learning models. Hence, typically, comparably big data is required for reasonable results. A challenging question is how to mediate this problem in practical applications, where small data or rare populations are present. Recent approaches propose mechanisms which enable a selection about how much potentially sensitive information can be revealed provided a higher accuracy is required, such as discussed e.g. in the work [37]. It would be an interesting endeavor to extend this technology to LVQ schemes.

References

[1] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318, 2016.

[2] Carlos Affonso, Andr Luis Debiaso Rossi, Fbio Henrique Antunes Vieira, and Andr Carlos Ponce de Leon Ferreira de Carvalho. Deep learning for biological image classification. *Expert Syst. Appl.*, 85(C):114–122, November 2017. ISSN 0957-4174. doi:10.1016/j.eswa.2017.05.039. URL `https://doi.org/10.1016/j.eswa.2017.05.039`.

[3] Patrick Ah-Fat and Michael Huth. Optimal accuracy-privacy trade-off for secure multi-

party computations. *CoRR*, abs/1803.00436, 2018. URL http://arxiv.org/abs/1803.00436.

[4] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *21st European Symposium on Artificial Neural Networks, Bruges, Belgium*, 2013. URL http://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2013-84.pdf.

[5] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela Jäschke, Christian A. Reuter, and Martin Strand. A guide to fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2015:1192, 2015. URL http://dblp.uni-trier.de/db/journals/iacr/iacr2015.html#ArmknechtBCGJRS15.

[6] Borja Balle and Yu-Xiang Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *JMLR Workshop and Conference Proceedings*, pages 403–412. JMLR.org, 2018. URL http://proceedings.mlr.press/v80/balle18a.html.

[7] Michael Biehl. Biomedical applications of prototype based classifiers and relevance learning. In *AlCoB: 4th Int. Conference on Algorithms for Computational Biology*, pages 3–23, 2017.

[8] Michael Biehl, Anarta Ghosh, and Barbara Hammer. Dynamics and generalization ability of LVQ algorithms. *Journal of Machine Learning Research*, 8:323–360, 2007. URL http://dl.acm.org/citation.cfm?id=1314511.

[9] Michael Biehl, Barbara Hammer, Petra Schneider, and Thomas Villmann. Metric learning for prototype-based classification. In *Innovations in Neural Information Paradigms and Applications*, pages 183–199. 2009. doi:10.1007/978-3-642-04003-0_8. URL https://doi.org/10.1007/978-3-642-04003-0_8.

[10] Michael Biehl, Petra Schneider, David Smith, Han Stiekema, Angela Taylor, Beverly Hughes, Cedric Shackleton, Paul Stewart, and Wiebke Arlt. Matrix relevance LVQ in steroid metabolomics based classification of adrenal tumors. In *20th European Symposium on Artificial Neural Networks, ESANN 2012, Bruges, Belgium, April 25-27, 2012*, 2012. URL https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2012-86.pdf.

[11] Michael Biehl, Barbara Hammer, Frank-Michael Schleif, Petra Schneider, and Thomas Villmann. Stationarity of matrix relevance LVQ. In *2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015*, pages 1–8, 2015. doi:10.1109/IJCNN.2015.7280441. URL https://doi.org/10.1109/IJCNN.2015.7280441.

[12] Michael Biehl, Barbara Hammer, and Thomas Villmann. Prototype-based models in machine learning. *Wiley Interdisciplinary Reviews: Cognitive Science*, 7(2):92–111, 2016.

[13] Christopher Bishop and Julia Lasserre. Generative or discriminative? getting the best of both worlds. 8:3–23, January 2007. URL https://www.microsoft.com/en-us/research/publication/generative-discriminative-getting-best-worlds/.

[14] Patrick Blöbaum, Shohei Shimizu, and Takashi Washio. Discriminative and generative models in causal and anticausal settings. In *Proceedings of the Second International Workshop on Advanced Methodologies for Bayesian Networks - Volume 9505*, AMBN 2015, pages 209–221, Berlin, Heidelberg, 2015. Springer-Verlag. ISBN 978-3-319-28378-4. doi:10.1007/978-3-319-28379-1_15. URL https://doi.org/10.1007/978-3-319-28379-1_15.

[15] Avrim Blum and Ronald L. Rivest. Training a 3-node neural network is np-complete. In Stephen Jose Hanson, Werner Remmele, and Ronald L. Rivest, editors, *Machine Learning: From Theory to Applications - Cooperative Research at Siemens and MIT*, volume 661 of *Lecture Notes in Computer Science*, pages 9–28. Springer, 1993. ISBN 3-540-56483-7. doi:10.1007/3-540-56483-7_20. URL https://doi.org/10.1007/3-540-56483-7_20.

[16] Johannes Brinkrolf and Barbara Hammer. Interpretable machine learning with reject option. *Automatisierungstechnik*, 66(4):283–290, 2018. doi:10.1515/auto-2017-0123. URL https://doi.org/10.1515/auto-2017-0123.

[17] Johannes Brinkrolf, Christina Göpfert, and Barbara Hammer. Differential privacy for learning vector quantization. *Neurocomputing*, 2019. doi:10.1016/j.neucom.2018.11.095.

[18] J. Richard Dowell. An overview of privacy and security requirements for data bases. In *Proceedings of the 15th Annual Southeast Regional Conference*, ACM-SE 15, pages 528–536, New York, NY, USA, 1977. ACM. doi:10.1145/1795396.1795469. URL http://doi.acm.org/10.1145/1795396.1795469.

[19] Wenliang Du and Mikhail J. Atallah. Secure multi-party computation problems and their applications: A review and open problems. In *Proceedings of the 2001 Workshop on New Security Paradigms*, NSPW '01, pages 13–22, New York, NY, USA, 2001. ACM. ISBN 1-58113-457-6. doi:10.1145/508171.508174. URL http://doi.acm.org/10.1145/508171.508174.

[20] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006. ISBN 3-540-35907-9. doi:10.1007/11787006_1. URL https://doi.org/10.1007/11787006_1.

[21] Cynthia Dwork. A firm foundation for private data analysis. *Commun. ACM*, 54(1):86–95, 2011. doi:10.1145/1866739.1866758. URL http://doi.acm.org/10.1145/1866739.1866758.

[22] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014. doi:10.1561/0400000042. URL https://doi.org/10.1561/0400000042.

[23] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006. ISBN 3-540-32731-2. doi:10.1007/11681878_14. URL https://doi.org/10.1007/11681878_14.

[24] Cynthia Dwork, Adam Smith, Thomas Steinke, and Jonathan Ullman. Exposed! a survey of attacks on private data. *Annual Review of Statistics and Its Application (2017)*, 2017.

[25] R. Fletcher. *Practical Methods of Optimization; (2Nd Ed.)*. Wiley-Interscience, New York, NY, USA, 1987. ISBN 0-471-91547-5.

[17] Keith B. Frikken. Algorithms and theory of computation handbook. chapter Secure Multiparty Computation, pages 14–14. Chapman & Hall/CRC, 2010. ISBN 978-1-58488-820-8. URL http://dl.acm.org/citation.cfm?id=1882723.1882737.

[27] Barbara Hammer and Thomas Villmann. Batch-rlvq. In *ESANN 2002, 10th Eurorean Symposium on Artificial Neural Networks, Bruges, Belgium, April 24-26, 2002, Proceedings*, pages 295–300, 2002.

[28] Barbara Hammer, Marc Strickert, and Thomas Villmann. Relevance LVQ versus SVM. In *Artificial Intelligence and Soft Computing - ICAISC 2004, 7th International Conference, Zakopane, Poland, June 7-11, 2004, Proceedings*, pages 592–597, 2004. doi:10.1007/978-3-540-24844-6_89. URL https://doi.org/10.1007/978-3-540-24844-6_89.

[29] Barbara Hammer, Marc Strickert, and Thomas Villmann. Supervised neural gas with general similarity measure. *Neural Processing Letters*, 21(1):21–44, 2005. doi:10.1007/s11063-004-3255-2. URL https://doi.org/10.1007/s11063-004-3255-2.

[30] Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-Cruz. Deep models under the GAN: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 603–618, 2017. doi:10.1145/3133956.3134012. URL http://doi.acm.org/10.1145/3133956.3134012.

[31] Klaus-Uwe Höffgen, Hans Ulrich Simon, and Kevin S. Van Horn. Robust trainability of single neurons. *J. Comput. Syst. Sci.*, 50(1):114–125, 1995. doi:10.1006/jcss.1995.1011. URL https://doi.org/10.1006/jcss.1995.1011.

[32] Zhanglong Ji, Zachary Chase Lipton, and Charles Elkan. Differential privacy and machine learning: a survey and review. *CoRR*, abs/1412.7584, 2014. URL http://arxiv.org/abs/1412.7584.

[33] Marika Kaden, Martin Riedel, Wieland Hermann, and Thomas Villmann. Border-sensitive learning in generalized learning vector quantization: an alternative to support vector machines. *Soft Comput.*, 19(9):2423–2434, 2015. doi:10.1007/s00500-014-1496-1. URL https://doi.org/10.1007/s00500-014-1496-1.

[34] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007. ISBN 1584885513.

[35] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[36] M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

[37] Katrina Ligett, Seth Neel, Aaron Roth, Bo Waggoner, and Zhiwei Steven Wu. Accuracy first: Selecting a differential privacy level for accuracy-constrained ERM. *CoRR*, abs/1705.10829, 2017. URL http://arxiv.org/abs/1705.10829.

[38] Zachary Chase Lipton. The mythos of model interpretability. *CoRR*, abs/1606.03490, 2016. URL http://arxiv.org/abs/1606.03490.

[39] Viktor Losing, Barbara Hammer, and Heiko Wersing. Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing*, 275:1261–1274, 2018. doi:10.1016/j.neucom.2017.06.084. URL `https://doi.org/10.1016/j.neucom.2017.06.084`.

[40] Paulo Martins, Leonel Sousa, and Artur Mariano. A survey on fully homomorphic encryption: An engineering perspective. *ACM Comput. Surv.*, 50(6):83:1–83:33, December 2017. ISSN 0360-0300. doi:10.1145/3124441. URL `http://doi.acm.org/10.1145/3124441`.

[41] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 94–103. IEEE Computer Society, 2007. ISBN 978-0-7695-3010-9. doi:10.1109/FOCS.2007.41. URL `https://doi.org/10.1109/FOCS.2007.41`.

[42] Johannes Merkert, Marcus Mueller, and Marvin Hubl. A survey of the application of machine learning in decision support systems. In *23rd European Conference on Information Systems, ECIS 2015, Münster, Germany, May 26-29, 2015*, 2015. URL `http://aisel.aisnet.org/ecis2015_cr/133`.

[43] Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. Smooth sensitivity and sampling in private data analysis. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 75–84. ACM, 2007. ISBN 978-1-59593-631-8. doi:10.1145/1250790.1250803. URL `http://doi.acm.org/10.1145/1250790.1250803`.

[44] D. Ravì, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, and G. Z. Yang. Deep learning for health informatics. *IEEE Journal of Biomedical and Health Informatics*, 21(1):4–21, Jan 2017. ISSN 2168-2194. doi:10.1109/JBHI.2016.2636665.

[45] Atsushi Sato and Keiji Yamada. Generalized learning vector quantization. In David S. Touretzky, Michael Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 423–429. MIT Press, 1995. ISBN 0-262-20107-0. URL `http://papers.nips.cc/paper/1113-generalized-learning-vector-quantization`.

[46] Petra Schneider, Michael Biehl, and Barbara Hammer. Adaptive relevance matrices in learning vector quantization. *Neural Computation*, 21(12):3532–3561, 2009.

[47] Sambu Seo and Klaus Obermayer. Soft learning vector quantization. *Neural Computation*, 15(7):1589–1604, 2003. doi:10.1162/089976603321891819. URL `https://doi.org/10.1162/089976603321891819`.

[48] Jun Zhang, Xiaokui Xiao, Yin Yang, Zhenjie Zhang, and Marianne Winslett. Privgene: differentially private model fitting using genetic algorithms. In Kenneth A. Ross, Divesh Srivastava, and Dimitris Papadias, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, pages 665–676. ACM, 2013. ISBN 978-1-4503-2037-5. doi:10.1145/2463676.2465330. URL `http://doi.acm.org/10.1145/2463676.2465330`.