DISSERTATION

LEARNING VECTOR QUANTIZATION FOR THE REAL-WORLD: PRIVACY, ROBUSTNESS, AND SPARSITY

JOHANNES BRINKROLF

Bielefeld University,
Faculty of Technology,
Machine Learning Research Group

SUPERVISED BY
PROF. DR. BARBARA HAMMER

REVIEWED BY
PROF. DR. BARBARA HAMMER
PROF. DR. KAI QIN

DECEMBER 2023

Printed on non-aging paper according to ISO 9706.

ACKNOWLEDGEMENTS

This work is the result of a long journey and would not have been possible without help from many people. First and foremost, I would like to thank my supervisor, Barbara Hammer, who gave me the opportunity to stay at Bielefeld University and do my Ph.D. within her work group. Her dedication made this time especially pleasant. Additionally, I would like to thank my second reviewer, Kai Qin, for his careful and in-depth reading as well as his helpful feedback. Thanks to both further committee members, Tim Nattkemper and David Johnson, for taking their time and making the disputation happen within two weeks.

Second, I would like to thank all my colleagues and co-authors for all fruitful discussions and coffee breaks. Specifically, Alexander Schulz, Benjamin Paaßen, and Christina Göpfert supported me, especially in the first years of my Ph.D. I am so grateful to Fabian Hinder and Valerie Vaquet for motivating and distracting me when necessary, providing me with work for procrastination, proofreading, and commenting on my first drafts of this thesis. They became close friends in the last few years.

I developed many ideas together with all my other academic friends I met many times at conferences and workshops. In particular, at the familial workshops held by Thomas Villmann, Michael Biehl, and Frank-Michael Schleif, new perspectives were discussed.

Third, I would like to thank all my friends who supported me during that time and made sure that I got enough breaks and left my office in the evenings. Some of them ensured that I got enough exercise not only at all the dance festivals and events but also on vacations.

Finally, I am grateful to my family, Anne and Christoph. I can come home any time to take a break and clear my mind. They have trusted and believed in me the whole time.

Thank you all!

ABSTRACT

Machine Learning (ML) methods are increasingly used and outperform humans in many specified and well-defined tasks. Considerable research focuses on optimizing the performance of such methodologies. However, the nature of application areas poses further challenges. For example, in critical domains, a false model behavior poses the risk of fatal mistakes. This is particularly relevant in traffic or medicine. In the latter, the data frequently contains sensitive information which should be preserved. Further, much data are recorded on distributed devices with limited computational power, like smartphones and peripheral devices. Hence, models of low complexity are required. As due to technical, legal, or strategic constraints, data transfer is limited employing intelligent mechanisms is crucial. This requires the consideration of further aspects beyond mere accuracy, namely privacy, robustness, efficiency, and distribution of the data itself.

In this thesis, I address these additional aspects, namely privacy, robustness, efficiency, and distribution of the data for prototype-based classifiers. In particular, I focus on Generalized Learning Vector Quantization (GLVQ) models and their variation to metric adaptations.

I show that the original GLVQ model bears the risk of revealing private information of samples present during the training. I propose three versions of training schemes provably obeying privacy. Further, I propose a novel reject option scheme for GLVQ models. Thereby increasing the robustness of the model is achieved. To reduce the complexity of a model and obtain a sparse representation of feature vectors, I apply regularization to the GLVQ scheme. Finally, I propose a methodology fusing model parameters of several models trained on distributed data sets.

Progress has not followed a straight ascending line, but a spiral with rhythms of progress and retrogression, of evolution and dissolution.

— JOHANN WOLFGANG VON GOETHE

CONTENTS

1 INTRODUCTION

Nowadays, *Machine Learning (ML)* is used in many systems in everyday life, for industry, and beyond. For example, industrial machines are monitored automatically, and processes are stopped if faulty items are produced or a component of the machine breaks (Bertolini et al. 2021; Xu et al. 2022). Numerous methodologies exist to detect abnormal behavior or to forecast failures such that the perfect time for maintenance of the machine can be scheduled (Çınar et al. 2020). By detecting any abnormality as soon as possible, production is more efficient as well a lot of money and resources can be saved.

Different methods which are used in these settings are vast in number. They differ enormously in their characteristics, ranging from data-heavy ones, which are capable of modeling high degrees of non-linearity, to methods that deal with comparably few data. The latter often use representations of typical samples or class means that are commonly called prototypes. Some examples are *Neural Gas (NG)*, *Self-Organizing Maps (SOM)*, and *Learning Vector Quantization (LVQ)* whereby the last one will be investigated in this thesis.

Learning Vector Quantization (LVQ) provides an intuitive representation of the data by optimizing the positions of prototypes. This representation is sparse in the sense that just a few representatives have to be stored describing the data and determining the class for a new sample. This way, LVQ combines the generative nature of the model with a discriminative objective that is typically used for supervised training (Biehl, Hammer, and Villmann 2016; Seo and Obermayer 2003). Additionally, it provides an explanation of the classification by inspecting the prototypes. Hence, it enables its utilization in domains where model interpretability is crucial such as in biomedical data analysis (Biehl 2017). These benefits encourage the increasing impact in few-shot and life-long learning (Losing, Hammer, and Wersing 2018; Snell, Swersky, and Zemel 2017; H.-M. Yang et al. 2018).

Classically, ML models are evaluated utilizing their generalization ability. Thereby, accuracy or any other loss function averaged over novel samples gathered according to the underlying probability is used. Yet, besides the mere accuracy, other objectives and characteristics become more and more popular as ML models are increasingly used in real-life scenarios. Such criteria are *privacy*, *robustness*, *efficiency*, and *distributed learning*.

- One concern is the *privacy* enforced, for example, by policymakers like the European Commission (European Commission 2020). As a lot of data contains sensitive, personal information, it needs to be ensured that the model does not leak any information when publishing it (Gong et al. 2020).
- Applications in critical areas like traffic, health, or medicine require more consideration of the model's *robustness*. Among others, this can be achieved by looking at the confidence of the predictions and introducing reject options.

- The amount of *computational power*, which is used to apply the model, can be reduced by increasing the model's *sparsity*. Such models can effortlessly run on smart devices and on the edge as in *edge computing*. Thus, the data can be processed in real-time on the same device where it is recorded without any data transfer.
- The data might be recorded on *distributed devices* in many real-world applications. This setup contradicts the classical assumption that the entire data set is available during the training. The straightforward solution would be gathering all distributed data sets and training the model on a single server. However, this is not always possible due to the considerable amount of data that needs to be transmitted. This problem is the key focus of the research area of *distributed learning* (Kairouz et al. 2021).

This thesis will center on technologies taking all these additional objectives into account for LVQ models. In the following, I will briefly motivate the main research questions addressing these objectives and the main technological methods used throughout the thesis to achieve these goals.

RQ1: What is required to preserve personal privacy in LVQ models?

Since data often consists of private data or records belonging to persons, protecting personal privacy gains significance. Using such data in ML, it is possible to uncover sensitive information from a given model. An extreme case is storing the training data in an explicit form which is executed by neighbor-based classifiers, for example. It has been shown that models which compress information might reveal sensitive information from individuals in the training data, too (Dwork and Roth 2014). This can even happen if the data is anonymized or sensitive attributes are discarded in the training set since auxiliary information might enable a reconstruction of data based on feature correlations (Narayanan and Shmatikov 2008).

In this thesis, I will address the question of to which extent LVQ models can preserve privacy. First, I will analyze whether LVQ models are vulnerable to privacy attacks. In Chapter 3, I show that LVQ models can leak information from used samples in training and hence, contradict the definition of privacy by Dwork (2006).

To eliminate this vulnerability, I propose three training schemes that satisfy this definition. For the first one, I adapt the training by replacing the initialization and optimization functions with privacy-preserving ones. In contrast to the other schemes, the training scheme remains unchanged. Here, I fuse the model parameters using those obtained from optimized models trained on subsets of the training set. I show that the fusing itself is privacy-preserving. Since privacy cannot be guaranteed without a loss of performance, I compare the classification performance of the novel schemes with those of non-private versions.

RQ2: How can the predictions of LVQ models be more robust?

ML is used increasingly in critical settings where an accurate model's prediction is demanded to prevent fatal errors (Gawlikowski et al. 2021; Hendrickx

et al. 2021). For this purpose and to increase the robustness, it is crucial to consider the model's certainty in addition to the prediction. In this work, I propose a novel approach for LVQ that provides class probabilities for all classes in a classification scenario. Based on these probabilities, I further suggest a time integration scheme that increases the certainty of predictions in a sequential manner.

Further, I establish strict boundaries for rejected samples extending a reject method by Fischer, Hammer, and Wersing (2015). Given these boundaries and a trained model, the data space of rejected points is precisely characterized. In particular, it also holds for the region close to the decision boundaries where distortions of the input sample can lead to different outputs of the model. Hence, the robustness of the model is scalable.

Finally, I consider the efficiency of LVQ models from two angles which will be elaborated on in the next two research questions:

RQ3: How can the model's complexity be reduced by enforcing sparsity?

I investigate the computational cost of predicting samples for LVQ models. In the case of high dimensional data, a subset containing only valuable features is sufficient for the classification task, e. g. features might be irrelevant, redundant, or correlated. The aim is to obtain a less complex model which only uses some input features. In Chapter 5, I present an adapted cost function for training LVQ models based on the idea of a shrinking operator. The additional penalty term inside the cost function pushes model parameters to zero while the classification performance is optimized simultaneously. After training, a sparse model is obtained where model parameters of as many features as possible are close to zero. By omitting those, I receive a less complex model.

RQ4: Which adaptions to the LVQ training scheme are necessary to train a model using distributed data sets?

Nowadays, data sets from similar distributions are collected on an increasing number of physically distributed devices. One option is to train individual models on each device using only the local data. However, in many cases combining information from more data sets is beneficial. Due to technical, private, legal, or strategic concerns, sometimes it is unfeasible transmitting all data to one server. Instead, an alternative approach is to rely on *Federated Learning (FL)* and transfer only some information extracted from the data to the server (Kairouz et al. 2021).

In this thesis, I propose the first federated LVQ. I exploit the property of LVQ, which already delivers strong results on small training sets. Further, I provide a merging scheme that combines the model parameters. On real-world benchmarks, the approach shows that it can also deal with missing classes in some subsets.

The structure of this thesis is as follows: First, Chapter 2 covers the notation, background knowledge, and related work for the remaining chapters. It introduces the different variants of LVQ models, their extensions, and the

benchmarks used to evaluate all new approaches. Chapter 3 shows how simple prototype-based classifiers can leak information from the training set. Therefore, three different methods are proposed which fulfill the definition of privacy and hence, preserve privacy. This chapter answers **RQ1**. Next, Chapter 4 addresses the robustness of LVQ and deals with **RQ2**. It describes post-processing steps for LVQ models that will lead to a better insight into the classification of new samples. They give the possibility to reject a prediction and to gain a certainty score for each class. Further, new bounds on minimum changes required for a change in the prediction will be given. Chapter 5 details an approach to obtain sparser models within the training procedure. Here, the prediction step is more efficient by discarding features from the input vectors that do not have any information to discriminant the classes. By removing such dimensions, the models become less complex and the classification becomes more efficient. **RQ3** will be answered in this chapter. The last research question, **RQ4**, is addressed in Chapter 6. It also deals with efficiency in the sense that the data is distributed on several servers and does not need to be transferred to one to train a model. A novel approach to obtain a general model via a fusion scheme is proposed. Chapter 7 concludes my thesis and provides an outlook.

This thesis covers work that I presented in journals as well as at international conferences. In particular, it contains work presented in the following conference and journal publications.

Conference Publications:

- Brinkrolf, Johannes, Tobias Mittag, et al. (2016). "Virtual optimisation for improved production planning". In: *Proceedings of the Workshop New Challenges in Neural Computation (NC^2 2016)*. Ed. by Barbara Hammer, Thomas Martinetz, and Thomas Villmann, pp. 67–74. URL: https://www.techfak.uni-bielefeld.de/~fschleif/mlr/mlr_04_2016.pdf#page=67.

- Brinkrolf, Johannes, Kolja Berger, and Barbara Hammer (2017). "Differential Privacy for Learning Vector Quantization". In: *Proceedings of the Workshop New Challenges in Neural Computation (NC^2 2017)*. Ed. by Barbara Hammer, Thomas Martinetz, and Thomas Villmann, pp. 17–25. URL: https://www.techfak.uni-bielefeld.de/~fschleif/mlr/mlr_03_2017.pdf#page=17.

- Brinkrolf, Johannes and Barbara Hammer (2017). "Probabilistic extension and reject options for pairwise LVQ". In: *12th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization, WSOM 2017, Nancy, France, June 28-30, 2017*, pp. 205–212. DOI: 10.1109/WSOM.2017.8020028. URL: https://doi.org/10.1109/WSOM.2017.8020028.

- Brinkrolf, Johannes, Kolja Berger, and Barbara Hammer (2018). "Differential private relevance learning". In: *26th European Symposium on Artificial Neural Networks, ESANN 2018, Bruges, Belgium, April 25-27, 2018*. URL: https://www.esann.org/sites/default/files/proceedings/legacy/es2018-119.pdf.

- Brinkrolf, Johannes and Barbara Hammer (2020a). "Sparse Metric Learning in Prototype-based Classification". In: *28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2020, Bruges, Belgium, October 2-4, 2020*. Ed. by Michel Verleysen, pp. 375–380. URL: https://www.esann.org/sites/default/files/proceedings/2020/ES2020-138.pdf.

- Brinkrolf, Johannes and Barbara Hammer (2021). "Federated Learning Vector Quantization". In: *29th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2021, Online event (Bruges, Belgium), October 6-8, 2021*. DOI: 10.14428/esann/2021.ES2021-141. URL: https://doi.org/10.14428/esann/2021.ES2021-141.

Journal Publications:

- Schulz, Alexander, Johannes Brinkrolf, and Barbara Hammer (2017). "Efficient kernelisation of discriminative dimensionality reduction". In: *Neurocomputing* 268, pp. 34–41. DOI: 10.1016/j.neucom.2017.01.104. URL: https://doi.org/10.1016/j.neucom.2017.01.104.

- Brinkrolf, Johannes and Barbara Hammer (2018). "Interpretable machine learning with reject option". In: *at - Automatisierungstechnik* 66.4, pp. 283–290. DOI: 10.1515/auto-2017-0123. URL: https://doi.org/10.1515/auto-2017-0123.

- Brinkrolf, Johannes, Christina Göpfert, and Barbara Hammer (2019). "Differential privacy for learning vector quantization". In: *Neurocomputing* 342, pp. 125–136. DOI: 10.1016/j.neucom.2018.11.095. URL: https://doi.org/10.1016/j.neucom.2018.11.095.

- Brinkrolf, Johannes and Barbara Hammer (2020b). "Time integration and reject options for probabilistic output of pairwise LVQ". In: *Neural Computing and Applications* 32.24, pp. 18009–18022. DOI: 10.1007/s00521-018-03966-0. URL: https://doi.org/10.1007/s00521-018-03966-0.

2 BACKGROUND

Overview: This chapter will introduce the notation used in this thesis and the notion of distances and dissimilarities. Additionally, we will explain all methods applied throughout this thesis, i. e. Learning Vector Quantization (LVQ), Generalized Learning Vector Quantization (GLVQ), and its variations. At the end of this chapter, the public benchmarks which will be used to illustrate the ability of all proposed and novel methods are introduced.

2.1 NOTATION

This section describes the main conventions and provides a description summarizing the notation. Small letters, e. g. c, d, and w are used for scalars or numbers referring to quantities of a set or the number of dimensions. Bold-faced letters, e. g. $\boldsymbol{a}$ or $\boldsymbol{x}$, indicate a vector in a higher dimension, often in $\mathbb{R}^d$, if it is not differently defined on the spot. A set of points is written as $\mathcal{X}$. In the case of training samples, the ith sample in a data set is notated as $\boldsymbol{x}_i$ and its label as y_i. In case one element of a vector is indexed, square brackets are used for distinction, e. g. the element in the jth dimension of a vector $\boldsymbol{a}$ is written as $[\boldsymbol{a}]_j$. Matrices are notated in bold-faced capital letters, like $\boldsymbol{A}$, and its transpose is written as $\boldsymbol{A}^\top$. If one element of a matrix is specified then two indices r and s are used, e. g. $[\boldsymbol{A}]_{rs}$ points to the element in the rth row and sth column. The following list contains the representation of all used mathematical notations:

$\mathbb{N}$, $\mathbb{R}$	set of natural and real numbers, respectively
$\mathbb{R}^d$	set of d-dimensional vectors over $\mathbb{R}$
$\mathcal{O}$	asymptotic notation
c, d, w	simple scalar or number, e. g. used for quantities: c refers to the number of classes, d to the dimension of the samples, and w to the number of prototypes.
$\boldsymbol{a}$, $\boldsymbol{x}$, $\boldsymbol{w}$	vectors typically in $\mathbb{R}^d$, $\boldsymbol{x}$ is a sample out of the data set and $\boldsymbol{w}$ is one prototype
$\mathbf{1}$	all-ones vector typically in $\mathbb{R}^d$, i. e. $\mathbf{1} = (1, \ldots, 1)^\top$
$[\boldsymbol{a}]_j$	element in the jth dimension of vector $\boldsymbol{a}$
$\|\boldsymbol{x}\|_2$ **or** $\|\boldsymbol{x}\|$	$= \sqrt{\boldsymbol{x}^\top \boldsymbol{x}} = \sqrt{\sum_{j=1}^{d} [\boldsymbol{x}]_j^2}$ (ℓ_2 norm, or Euclidean norm of $\boldsymbol{x}$)
$\|\boldsymbol{x}\|_1$	$= \sum_{j=1}^{d} \left\lvert [\boldsymbol{x}]_j \right\rvert$ (ℓ_1 norm, or absolute value norm of $\boldsymbol{x}$)
$\|\boldsymbol{x}\|_0$	number of non-zero elements of $\boldsymbol{x}$ (ℓ_0 norm)

$\boldsymbol{A} \in \mathbb{R}^{m \times d}$	$m \times d$ matrix over $\mathbb{R}$
$\mathbb{I}$, $\mathbb{I}_d$	identity matrix $\left(\text{if not specified differently then in } \mathbb{R}^{d \times d}\right)$
$[\boldsymbol{A}]_{rs}$	element of $\boldsymbol{A}$ in the rth row and the sth column
$\boldsymbol{A}^\top$	transpose of $\boldsymbol{A}$
Φ	a monotonic increasing function, e. g. sigmoid function, logistic function, or identity
$\mathcal{D}$	instances domain (of a set or data set), typically a subset of $\mathbb{R}^d$
$\boldsymbol{x}_i$	ith sample of $\mathcal{D}$
y_i	label or output of $\boldsymbol{x}_i$
$\mathrm{c}(\boldsymbol{x})$	function which gives the label or output of $\boldsymbol{x}$
$\mathbb{P}$, $\mathbb{E}$	probability and expectation of a random variable
$\mathcal{N}(\mu, \sigma)$	Gaussian distribution with expectation μ, standard deviation σ, and PDF $\mathbb{P}[\mathcal{N}(\mu, \sigma) = x] = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$
$\mathrm{Lap}(\mu, \beta))$	Laplace distribution with location μ, scaling factor β, and PDF $\mathbb{P}[\mathrm{Lap}(\mu, \beta) = x] = \frac{1}{2\beta} e^{-\lvert x-\mu \rvert/\beta}$
$f'(x)$	derivative of function $f\colon \mathbb{R} \to \mathbb{R}$ at x
$\frac{\partial}{\partial [w]_i} f(\boldsymbol{w})$	partial derivative of function $f\colon \mathbb{R}^d \to \mathbb{R}$ at $\boldsymbol{w}$ w. r. t. $[w]_i$
$\min_{x \in C} f(x)$	$= \min\{f(x) \mid x \in C\}$ (minimal value of f over C)
$\max_{x \in C} f(x)$	$= \max\{f(x) \mid x \in C\}$ (maximal value of f over C)
$\arg\min_{x \in C} f(x)$	$= \text{set } \{x \in C \mid f(x) = \min_{z \in C} f(z)\}$
$\arg\max_{x \in C} f(x)$	$= \text{set } \{x \in C \mid f(x) = \max_{z \in C} f(z)\}$
log	natural logarithm

2.1.1 Distances and Dissimilarities

The entire work is centered around classifiers that use the notion of *distances* or *dissimilarities*. This is common for many ML methods (Pekalska and Duin 2005). An intuitive view of a distance is the length of the shortest path between two points. In a physical environment, a straight line is often considered. This section will define distances first and then clarify the difference between distances and dissimilarities.

Definition 1 (Distance). *Let $\mathcal{D}$ be an arbitrary set. A function $\bar{d}\colon \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ is called a* distance *if and only if for all $\boldsymbol{p}$, $\boldsymbol{q}$, and $\boldsymbol{z} \in \mathcal{D}$ it holds:*

$$\bar{d}(\boldsymbol{p}, \boldsymbol{q}) \geq 0 \qquad \text{non-negativity,} \tag{2.1}$$
$$\bar{d}(\boldsymbol{p}, \boldsymbol{q}) = 0 \Leftrightarrow \boldsymbol{p} = \boldsymbol{q}, \qquad \text{identity of indiscernibles,} \tag{2.2}$$
$$\bar{d}(\boldsymbol{p}, \boldsymbol{q}) = \bar{d}(\boldsymbol{q}, \boldsymbol{p}), \qquad \text{symmetry, and} \tag{2.3}$$
$$\bar{d}(\boldsymbol{p}, \boldsymbol{q}) \leq \bar{d}(\boldsymbol{p}, \boldsymbol{z}) + \bar{d}(\boldsymbol{z}, \boldsymbol{q}) \qquad \text{triangle inequality.} \tag{2.4}$$

Note that from Inequality (2.1) and Equation (2.2) it follows that $\bar{d}(\boldsymbol{p}, \boldsymbol{q}) > 0$ if $\boldsymbol{p} \neq \boldsymbol{q}$. Instead of the distance, a less strict form is considered, in this thesis. We will denote the dissimilarity as $d(\boldsymbol{p}, \boldsymbol{q})$ and define it in the next definition.

Definition 2 (Dissimilarity). *Let $\mathcal{D}$ be an arbitrary set. A function $d\colon \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ is called a* dissimilarity *if and only if for all $\boldsymbol{p}$ and $\boldsymbol{q} \in \mathcal{D}$ the following two properties hold:*

$$d(\boldsymbol{p}, \boldsymbol{q}) \geq 0 \qquad \text{non-negativity and} \tag{2.5}$$
$$d(\boldsymbol{p}, \boldsymbol{q}) = d(\boldsymbol{q}, \boldsymbol{p}) \qquad \text{symmetry.} \tag{2.6}$$

Due to the missing identity of indiscernibles (Equation (2.2)) the triangle inequality does not hold, either. This phenomenon for dissimilarities is demonstrated in Example 1.

Example 1. *A simple example of dissimilarity is the squared Euclidean distance. Let $\bar{d}(A, B) = 2$, $\bar{d}(A, C) = 4$, and $\bar{d}(B, C) = 3$ then the triangle Inequality (2.4) holds for the distances, hence $\bar{d}(A, C) = 4 < 2 + 3 = \bar{d}(A, B) + \bar{d}(B, C)$. However, the inequality does not hold for the squared distance since $\bar{d}(A, C)^2 = 4^2 = 16 > 13 = 4 + 9 = 2^2 + 3^2 = \bar{d}(A, B)^2 + \bar{d}(B, C)^2$.*

The next section will introduce classifiers that use squared distances as measurements of dissimilarities for distinguishing two points.

2.2 LEARNING VECTOR QUANTIZATION

In this section, the central ML algorithms used in this thesis are introduced: Learning Vector Quantization and variants of GLVQ. Versions of these and an overview of the taxonomy for LVQ classifiers are given later in Section 2.2.6. Before introducing the original classifier, a brief overview of ML is given.

ML approaches are typically divided into three categories: supervised learning, unsupervised learning, and reinforcement learning (Bishop 2007). In the former, the algorithms build a mathematical model of a data set containing pairs of inputs and corresponding outputs. If the target output is an element of a finite set, it is a *classification* problem. Otherwise, if the output is continuous, it is a *regression* task. The goal of both tasks is to learn a general rule or function that maps inputs to outputs. In contrast, in unsupervised learning, no outputs are given to the learning algorithm. Considered tasks are finding some structures in the data, e. g. finding groupings, clusters, or

outliers. Typically, reinforcement learning aims to find the optimal actions for an agent interacting within a dynamic environment.

As already stated, we will only look at classification scenarios in $\mathcal{D} \subset \mathbb{R}^d$ with multiple classes, also known as a multi-class problem, in this thesis. For simplicity, we enumerate the classes from 1 to c. For a finite set with c elements, it is always achievable to map each element to exactly one number between 1 and c by a bijective function. A *classifier* maps any input $\boldsymbol{x}$ to one of the classes. More formal:

$$\mathrm{h}\colon \mathcal{D} \to \{1, \ldots, c\}. \tag{2.7}$$

Note that there also exists the task that a subset of all possible classes is the desired output. In the literature, this problem is called multi-label classification but will not be further discussed here.

In this thesis, we focus on prototype-based classifiers, particularly, on LVQ, as intuitive and distance-based classifiers. They are defined as follows: labeled prototypes $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_w$ with labels $\mathrm{c}(\boldsymbol{w}_j) \in \{1, \ldots, c\}\ \forall j \in \{1, \ldots, w\}$ are specified such that a good classification and representation of the data is achieved. A new sample $\boldsymbol{x}$ is classified by the *winner-takes-all* scheme:

$$\boldsymbol{x} \mapsto \mathrm{c}(\boldsymbol{w}_{J(\boldsymbol{x})}) \quad \text{where} \quad J(\boldsymbol{x}) := \underset{j \in \{1, \ldots, w\}}{\arg\min}\, d(\boldsymbol{x}, \boldsymbol{w}_j). \tag{2.8}$$

Note that the number of prototypes (w) does not have to be the same number of classes (c). Having more than one prototype per class or different numbers of prototypes for each class might also be reasonable. Standard LVQ schemes use the squared Euclidean metric

$$d(\boldsymbol{x}, \boldsymbol{w}_j) = (\boldsymbol{x} - \boldsymbol{w}_j)^\top (\boldsymbol{x} - \boldsymbol{w}_j). \tag{2.9}$$

Given a set of labeled training samples $\{(\boldsymbol{x}_i, y_i) \in \mathcal{D} \times \{1, \ldots, c\} \mid i = 1, \ldots, n\}$, prototypes $\boldsymbol{w}_j$ are adapted such that the classification error for the given training set is as small as possible. Since in general, this problem is NP-hard (Blum and Rivest 1993; Höffgen, H. U. Simon, and Horn 1995), heuristics or approximations of the 0-1-loss are used.

Standard LVQ (referred to as LVQ1) by Kohonen (1997) relies on the heuristics of Hebbian Learning: given a training point $(\boldsymbol{x}_i, y_i)$, the winner, i. e. closest prototype $\boldsymbol{w}_{J(\boldsymbol{x}_i)}$, is determined and adapted by the rule

$$\boldsymbol{w}_{J(\boldsymbol{x}_i)} = \begin{cases} \boldsymbol{w}_{J(\boldsymbol{x}_i)} + \eta \cdot (\boldsymbol{x}_i - \boldsymbol{w}_{J(\boldsymbol{x}_i)}) & \text{if } y_i = \mathrm{c}(\boldsymbol{w}_{J(\boldsymbol{x}_i)}) \\ \boldsymbol{w}_{J(\boldsymbol{x}_i)} - \eta \cdot (\boldsymbol{x}_i - \boldsymbol{w}_{J(\boldsymbol{x}_i)}) & \text{otherwise.} \end{cases} \tag{2.10}$$

The prototype is shifted into the direction of the training point if it belongs to the same class as the sample $\boldsymbol{x}_i$ and pushed to the opposite direction otherwise. LVQ does not possess a valid cost function, yet it shows surprisingly accurate behavior in typical model situations (Biehl, Ghosh, and Hammer 2007; Biehl, Hammer, Schneider, et al. 2009). Additionally, general convergence guarantees of stochastic gradient schemes apply, and dynamics have been investigated, e. g. in the framework of statistical physics of online learning (Biehl, Ghosh, and Hammer 2007). Further, the positions of the prototypes converge to the centers of gravity of the classes provided data is separable, as has been shown in the articles by Biehl, Hammer, Schleif, et al. (2015) and Hammer and Villmann (2002a).

2.2.1 Generalized Learning Vector Quantization

There exist different extensions of LVQ to a cost function, including probabilistic frameworks such as those proposed in the approach by Seo and Obermayer (2003), or deterministic approximations of the classification loss, that also relate to the objective of (hypothesis) margin maximization (Sato and Yamada 1995; Schneider, Biehl, and Hammer 2009a). For the latter, GLVQ, the cost function becomes

$$E = \sum_{i=1}^{n} \Phi(\mu(\boldsymbol{x}_i)) \quad \text{where } \mu(\boldsymbol{x}_i) = \frac{d(\boldsymbol{x}_i, \boldsymbol{w}^+) - d(\boldsymbol{x}_i, \boldsymbol{w}^-)}{d(\boldsymbol{x}_i, \boldsymbol{w}^+) + d(\boldsymbol{x}_i, \boldsymbol{w}^-)} \tag{2.11}$$

and Φ is a monotonic increasing function, e. g. the logistic function or the identity $\Phi(\boldsymbol{x}) = \boldsymbol{x}$. The $d(\boldsymbol{x}_i, \boldsymbol{w}^{\pm})$ are the squared distances of the sample $\boldsymbol{x}_i$ to the closest prototype $\boldsymbol{w}^+$ belonging to the same class and to the closest prototype $\boldsymbol{w}^-$ of a different class than $\boldsymbol{x}_i$. More formally, for a sample $\boldsymbol{x}_i$

$$\boldsymbol{w}^+ = \underset{\substack{j \in \{1, \ldots, w\} \\ c(\boldsymbol{w}_j) = c(\boldsymbol{x}_i)}}{\arg\min} \; d(\boldsymbol{x}_i, \boldsymbol{w}_j) \quad \text{and} \quad \boldsymbol{w}^- = \underset{\substack{j \in \{1, \ldots, w\} \\ c(\boldsymbol{w}_j) \neq c(\boldsymbol{x}_i)}}{\arg\min} \; d(\boldsymbol{x}_i, \boldsymbol{w}_j).$$

Training takes place based on a given training set as above and consists of two steps. First, the prototypes are initialized within the class centers. If there is more than one prototype in one class, a small noise is typically added such that finding the closest prototype becomes unique. Second, the cost function E is minimized by simple gradient descent or second-order techniques such as limited-memory BFGS (LBFGS). For that purpose, the derivatives for both prototypes $\boldsymbol{w}^+$ and $\boldsymbol{w}^-$ are required. Using the chain rule, the derivative for one point $\boldsymbol{x}$ and the prototype $\boldsymbol{w}^+$ yields

$$\frac{\partial}{\partial \boldsymbol{w}^+} E = \frac{\partial}{\partial \mu(\boldsymbol{x})} \Phi\big(\mu(\boldsymbol{x})\big) \cdot \frac{2 d(\boldsymbol{x}, \boldsymbol{w}^-)}{\big(d(\boldsymbol{x}, \boldsymbol{w}^+) + d(\boldsymbol{x}, \boldsymbol{w}^-)\big)^2} \cdot \frac{\partial}{\partial \boldsymbol{w}^+} d(\boldsymbol{x}, \boldsymbol{w}^+) \tag{2.12}$$

and for $\boldsymbol{w}^-$

$$\frac{\partial}{\partial \boldsymbol{w}^-} E = \frac{\partial}{\partial \mu(\boldsymbol{x})} \Phi\big(\mu(\boldsymbol{x})\big) \cdot \frac{-2 d(\boldsymbol{x}, \boldsymbol{w}^+)}{\big(d(\boldsymbol{x}, \boldsymbol{w}^+) + d(\boldsymbol{x}, \boldsymbol{w}^-)\big)^2} \cdot \frac{\partial}{\partial \boldsymbol{w}^-} d(\boldsymbol{x}, \boldsymbol{w}^-) \tag{2.13}$$

with

$$\frac{\partial}{\partial \boldsymbol{w}^{\pm}} d(\boldsymbol{x}, \boldsymbol{w}^{\pm}) = -2(\boldsymbol{x} - \boldsymbol{w}^{\pm}). \tag{2.14}$$

Comparing the derivative with the update rules for LVQ1 in Equation (2.10) yields that also the prototype belonging to the same class ($\boldsymbol{w}^+$) is shifted into the direction of the sample $\boldsymbol{x}_i$. The closest prototype of another class ($\boldsymbol{w}^-$) is pushed away. Additionally, the step-size is multiplied by a factor consisting of the partial derivatives of Φ and μ. Later, in the other chapters, we will adapt the cost function, and therefore, the derivatives are changed.

Some approaches rely on the choice of Φ as a sigmoid function, which enables a more fine-grained tuning of the region of interest along the decision boundary within so-called border-sensitive schemes (Kaden et al. 2015). In this thesis, the identity $\Phi(\boldsymbol{x}) = \boldsymbol{x}$ is utilized as an activation function, since it has historically proven excellent performance.

2.2.2 Generalized Matrix Learning Vector Quantization

Since the choice of distance is crucial for the performance of the model, GLVQ has been generalized to metric learning schemes dubbed Generalized Matrix Learning Vector Quantization (GMLVQ) (Schneider, Biehl, and Hammer 2009a). Essentially, a positive semi-definite quadratic form $\Lambda = \Omega^\top \Omega \in \mathbb{R}^{d\times d}$ is used to define a generalized squared distance function

$$d_\Lambda(\boldsymbol{x}, \boldsymbol{w}) = (\boldsymbol{x} - \boldsymbol{w})^\top \Lambda (\boldsymbol{x} - \boldsymbol{w}). \tag{2.15}$$

This distance is then used in the winner-takes-all function of the classifier and the cost function E, see Equation (2.11). Adaptation takes place via gradient schemes with respect to prototypes and matrix parameters of Ω. For GMLVQ, the gradients for both prototypes, see Equations (2.12) and (2.13), hold once replacing the distance $d(\boldsymbol{x}, \boldsymbol{w}^\pm)$ with $d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^\pm)$ and the gradient for the latter one yields

$$\frac{\partial}{\partial \boldsymbol{w}^\pm} d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^\pm) = -2\Lambda(\boldsymbol{x} - \boldsymbol{w}^\pm). \tag{2.16}$$

Likewise, the matrix parameters in Ω are updated. The gradient of the cost function with respect to Ω is

$$\frac{\partial}{\partial \Omega} E = \frac{\partial}{\partial \mu(\boldsymbol{x})} \Phi\big(\mu(\boldsymbol{x})\big) \cdot \frac{\partial}{\partial \Omega} \mu(\boldsymbol{x}) \tag{2.17}$$

where for one matrix element in the rth row and sth column

$$\begin{aligned} \frac{\partial}{\partial [\Omega]_{rs}} \mu(\boldsymbol{x}) =& 2 \left(\frac{d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^-)}{\big(d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^+) + d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^-)\big)^2} \cdot \frac{\partial}{\partial [\Omega]_{rs}} d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^+) \right. \\ & \left. - \frac{d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^+)}{\big(d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^+) + d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^-)\big)^2} \cdot \frac{\partial}{\partial [\Omega]_{rs}} d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^-) \right) \end{aligned} \tag{2.18}$$

with

$$\begin{aligned} \frac{\partial}{\partial [\Omega]_{rs}} d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^\pm) &= 2 \sum_{j=1}^{d} \big([\boldsymbol{x}]_s - [\boldsymbol{w}^\pm]_s\big) [\Omega]_{rj} \big([\boldsymbol{x}]_j - [\boldsymbol{w}^\pm]_j\big) \\ &= 2\big([\boldsymbol{x}]_s - [\boldsymbol{w}^\pm]_s\big) \cdot [\Omega(\boldsymbol{x} - \boldsymbol{w}^\pm)]_r. \end{aligned} \tag{2.19}$$

Initialization of the prototypes is typically done via class centers. The relevance matrix is set as the identity corresponding to the standard squared Euclidean distance, or with uniform random numbers, i. e. $[\Omega]_{rs} \in [-1, 1]$. After each update step, the projection matrix is normalized such that

$$\sum_j [\Lambda]_{jj} = \sum_{rs} [\Omega]_{rs}^2 = 1. \tag{2.20}$$

This prevents degeneration of the matrix. Note that this does not change the classification because the distances are multiplied by the same factor for each sample (Schneider, Biehl, and Hammer 2009a).

2.2.3 Local Generalized Matix Learning Vector Quantization

The principle of GMLVQ can be further extended to local schemes, dubbed Local Generalized Matix Learning Vector Quantization (LGMLVQ) by attaching different relevance matrix Λ_j to each prototype $\boldsymbol{w}_j$. For the dissimilarity measurement, the formula becomes

$$d_{\Lambda_j}(\boldsymbol{x}, \boldsymbol{w}_j) = (\boldsymbol{x} - \boldsymbol{w}_j)^\top \Lambda_j (\boldsymbol{x} - \boldsymbol{w}_j). \tag{2.21}$$

The derivative of Equation (2.21) with respect to the corresponding prototype is then

$$\frac{\partial}{\partial \boldsymbol{w}_j} d_{\Lambda_j}(\boldsymbol{x}, \boldsymbol{w}_j) = -2\Lambda_j(\boldsymbol{x} - \boldsymbol{w}_j). \tag{2.22}$$

While training a LGMLVQ model, the projection matrices for both prototypes, the closest belonging to the same class ($\boldsymbol{w}^+$) and to a different class ($\boldsymbol{w}^-$) are updated individually. For both projection matrices Ω^+ and Ω^-, respectively, the derivatives for each matrix element hold

$$\frac{\partial}{\partial [\Omega^+]_{rs}} \mu(\boldsymbol{x}) = \frac{2 d_{\Lambda_-}(\boldsymbol{x}, \boldsymbol{w}^-)}{(d_{\Lambda_+}(\boldsymbol{x}, \boldsymbol{w}^+) + d_{\Lambda_-}(\boldsymbol{x}, \boldsymbol{w}^-))^2} \left(([\boldsymbol{x}]_s - [\boldsymbol{w}^+]_s)[\Omega^+(\boldsymbol{x} - \boldsymbol{w}^+)]_r \right) \tag{2.23}$$

and

$$\frac{\partial}{\partial [\Omega^-]_{rs}} \mu(\boldsymbol{x}) = \frac{-2 d_{\Lambda_+}(\boldsymbol{x}, \boldsymbol{w}^+)}{(d_{\Lambda_+}(\boldsymbol{x}, \boldsymbol{w}^+) + d_{\Lambda_-}(\boldsymbol{x}, \boldsymbol{w}^-))^2} \left(([\boldsymbol{x}]_s - [\boldsymbol{w}^-]_s)[\Omega^-(\boldsymbol{x} - \boldsymbol{w}^-)]_r \right). \tag{2.24}$$

One benefit of LGMLVQ is the ability to find non-linear decision boundaries that are composed of quadratic pieces (Schneider, Biehl, and Hammer 2009a). Each matrix Λ_j introduces an individual ellipsoidal cluster around every prototype. This effect can increase the classifier's performance as we will see in the next section. However, the number of parameters and the complexity of the model increase a lot. Instead of only training one matrix $\Lambda \in \mathbb{R}^{d \times d}$, in LGMLVQ, each prototype has its own matrix. Thus, for each prototype, we obtain $(d+1)d$ parameters where d is the data dimensionality. There is a strategy that limits the parameters per prototype by a fixed constant m such that each prototype has $(m+1)d$ parameters. We will see this method in the next section.

2.2.4 Limited Rank Matrix Learning

For the relevance matrices, it has been shown that the stationary matrices become singular with only one or a few non-zero eigenvalues under quite general conditions (Biehl, Hammer, Schleif, et al. 2015). Hence, we can enforce lower ranks and limit the maximum rank with the so-called limited rank approach that we will briefly introduce next.

So far, we do not make any restrictions to the projection matrices Ω and Ω_j for GMLVQ and LGMLVQ. They form the positive (semi-) definite relevance matrices Λ, $\Lambda_j \in \mathbb{R}^{d\times d}$, respectively. As Bunte, Schneider, et al. (2012) have presented, a rectangular projection matrices Ω, $\Omega_j \in \mathbb{R}^{m\times d}$ with $m < d$ can also be used. Here, the matrices can be interpreted as linear transformations and projections of the d-dimensional feature space into a smaller $\mathbb{R}^m$ vector space. The squared distances can be rewritten as

$$d_\Lambda(\boldsymbol{x}, \boldsymbol{w}) = (\boldsymbol{x}-\boldsymbol{w})^\top \Omega^\top \Omega(\boldsymbol{x}-\boldsymbol{w}) = (\Omega(\boldsymbol{x}-\boldsymbol{w}))^2 \tag{2.25}$$

and

$$d_{\Lambda_j}(\boldsymbol{x}, \boldsymbol{w}_j) = (\boldsymbol{x}-\boldsymbol{w}_j)^\top \Omega_j^\top \Omega_j(\boldsymbol{x}-\boldsymbol{w}_j) = (\Omega_j(\boldsymbol{x}-\boldsymbol{w}_j))^2 \tag{2.26}$$

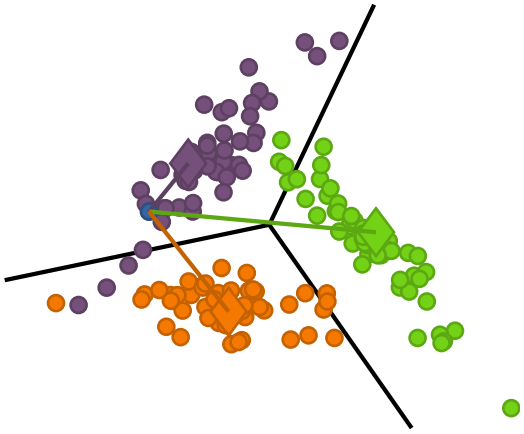
where the projections of the data and the prototypes become more clear. Of course, this reduces the memory and the computation complexity, too, especially if the number of features is high which will be discussed in more detail in Chapter 5. The derivatives as they are stated in Equations (2.16) to (2.19) and (2.22) to (2.24) still hold for these rectangular matrices. In the training scheme, the initialization of the relevance matrix is adopted. Instead of starting with a diagonal matrix, typically, a random rectangular matrix drawn from a uniform distribution is used, i. e. $[\Omega]_{rs} \in [-1, 1]$.

2.2.5 Visualization of GMLVQ and LGMLVQ

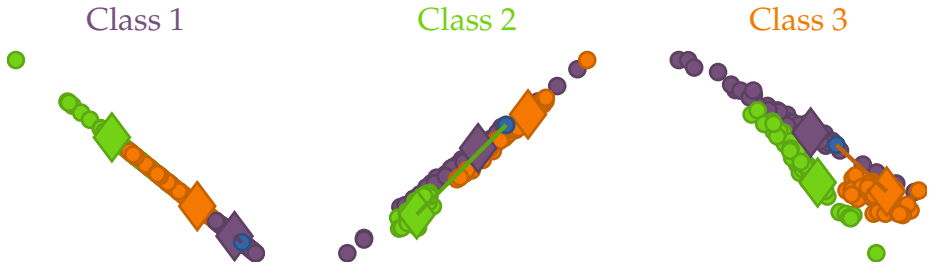
An additional benefit of the stationarity of the relevance matrices for GMLVQ and LGMLVQ models is the ability to visualize the classifier. In the case of low-rank matrices, the data and the prototypes can be plotted via a projection to a two-dimensional space resulting in a suitable approximation of how the classifier is separating the classes and classifying the data. For this purpose, a proper projection can be derived from the relevance matrix of a trained GLVQ model. In the case of one global relevance matrix, the samples and the prototypes can be projected onto the eigenvectors of the two main eigenvalues of Λ. This projection approximately displays the setting as it is relevant for classification since the two eigenvectors dominate the classification prescription. Indeed, it has been shown by Biehl, Hammer, Schleif, et al. (2015) that GMLVQ tends to learn low-rank matrices providing a good visualization and excellent generalization ability. We demonstrate this property in Example 2.

For LGMLVQ, we can also project the samples and prototypes to gain a good visualization. Note that each prototype has its own relevance matrix, and we obtain a different projection for each Λ_j (see Figure 2.1(b)).

Instead of using the relevance matrix and calculating the biggest eigenvectors, the limited rank approach from the previous section can also be applied to set the rank to two, i. e. set $m = 2$. Thus, the projection matrices Ω and Ω_j can directly be used as projections because they map the data and prototypes into a two-dimensional vector space where the classification takes place. This leads to an exact visualization of the trained classifier. However, the classifier might perform worse due to the strong regularization which will be discussed in more detail in Chapter 5.



(a) GMLVQ visualization. The data is projected on the first two eigenvectors of the relevance matrix Λ.



(b) LGMLVQ visualization. For each prototype, w_j, the data is projected on the first two eigenvectors of the corresponding relevance matrix Λ_j. Samples from the same class as the prototype are pushed together.

Figure 2.1: Visualization of the data and the models from Example 2 for GMLVQ and LGMLVQ. The distances between a new sample • and the prototypes are depicted as lines in their colors.

Example 2. *To demonstrate the visualization of GMLVQ and LGMLVQ, in Figure 2.1, we show the induced projection of a toy data set consisting of three blobs, which represent three classes. In Figure 2.1(a), the data and the prototypes are projected onto the two biggest eigenvectors of the relevance matrix of the GMLVQ model. Additionally, the decision boundaries are approximated by the perpendicular bisectors of two prototypes, drawn as black lines. The classes are separated by the borders satisfactorily. Only some samples are not classified correctly as they lie on a different side of the decision boundaries than the concerning prototype. In Figure 2.1(b), the intrinsic low rank of the relevance matrices becomes more clear as the projection of the data is nearly one-dimensional. For class 1 and 2 the samples are projected onto a single line and all samples belonging to that class coincide. That is because, in the training, the distances between the prototype and the corresponding samples are minimized. We depict the distance by drawing the line from one sample (•), the same as in Figure 2.1(a) for GMLVQ, to the prototype of class 1, 2, and 3 in their color, respectively.*

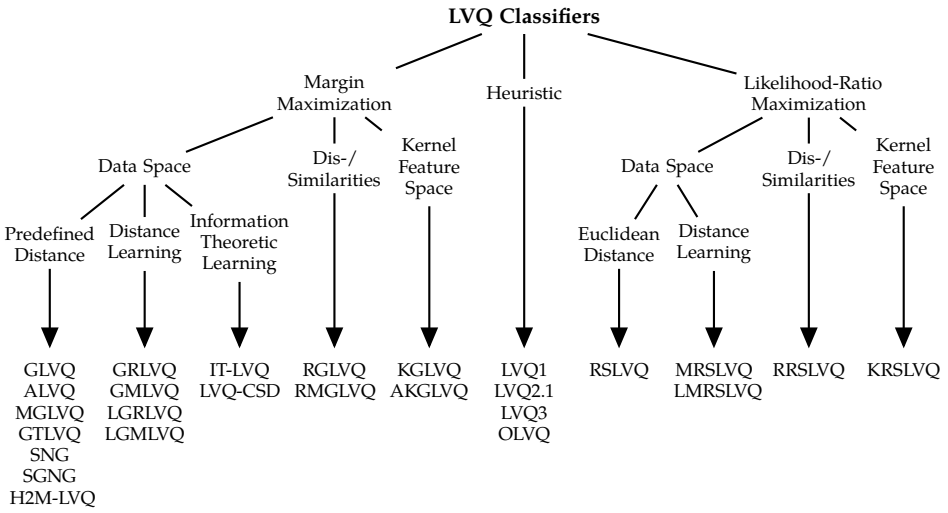


Figure 2.2: Taxonomy of relevant LVQ classifiers. This collection is extended and adapted from the work by Nova and Estévez (2014).

2.2.6 Taxonomy of Learning Vector Quantization

There exist many further variants of LVQ. We will now present a brief overview of the taxonomy of LVQ classifiers. The methodologies can be decomposed into three central families: methods based on *heuristics*, *margin maximization*, and *likelihood-ratio maximization*, as it is visualized in Figure 2.2. Due to space limitations, we will not refer to all the methods in the visualization. Instead, Table A.1 lists all methodologies with their properties and references in alphabetic order.

We have already seen the LVQ1 classifier that optimizes the prototype's positions using the Hebbian Learning *heuristic* (see Equation (2.10)). While this learning rule updates only the closest prototype at each step, further approaches immediately updating two prototypes have been proposed. They aim to achieve higher convergence speed and optimize the decision boundary towards the theoretical Bayes one. For example, some methods are LVQ2.1, LVQ3, and OLVQ (Kohonen 1990).

Next to the heuristic-based methods, numerous ones *maximizing the margin* have been proposed. They build on the cost function shown in Equation (2.11). These approaches solve some limitations of the original LVQ algorithms such as convergence, initialization sensitivity, and limitations in multidimensional data where correlations exist between dimensions. The data can be vectorial such that the classification takes place in the *data space*. In this scenario, the methods can be split into three groups.

In the first group, a *predefined dissimilarity*, typically the Euclidean distance or the squared one, is used. Examples are GLVQ, as introduced in Section 2.2.1, Supervised Neural Gas (SNG) approaches (Hammer, Strickert, and Villmann 2005; Jirayusakul and Auwatanamongkol 2007), and MGLVQ (Nebel, Hammer, and Villmann 2013). The latter one is special in the sense that the prototype are samples from the training set. Some methods use other dissimilarities, like the angle between two vectors (Bunte, Baranowski, et al. 2016) or the tangent distance defined by the distance between a point and its

projection onto a (bounded) subspace (Saralajew and Villmann 2016).

The second group consists of methods which *adapt the dissimilarity*, like GMLVQ and LGMLVQ as introduced in Sections 2.2.2 and 2.2.3. Similar to them, the relational methods, GRLVQ and LGRLVQ use a diagonal relevance matrix $\Lambda = \mathrm{diag}(\boldsymbol{\lambda})$ with $\boldsymbol{\lambda} \in \mathbb{R}^d$ and Λ_j for each prototype $\boldsymbol{w}_j$, $j \in \{1, \ldots, w\}$, respectively (Hammer, Schleif, and Villmann 2005; Hammer and Villmann 2002b).

Methods from the third group are based on *information theoretic learning*. Using one divergence measure, e. g. Cauchy-Schwarz divergence, the methods match data and prototype densities to supervised learning and classification. Hence, the methods can deal with fuzzy labels for samples as well as prototypes (Villmann and Haase 2011; Villmann, Hammer, et al. 2008).

All those methods deal (in their original version) with vectors lying in a fixed vector space. In contrast, there are methods utilizing either a *dissimilarity matrix* or a *kernel function*. For both options, samples do not need to be embedded into a vector space. Thus, the model can, for example, handle time series, structural, and textual data which makes it applicable in more areas. For these approaches defining a function that gives the *dissimilarity* of two samples is required. These methods are dubbed *Relational* GLVQ (Gisbrecht et al. 2012). In the case of the *kernelized* versions, the data space is mapped into a higher dimensional possibly linearly separable feature space. Hence, also non-linear decision boundaries to separate classes are possible (Qin and Suganthan 2004).

The last branch of LVQ classifiers called *Robust Soft* LVQ consists of methods that *maximize the likelihood-ratio*. Again, extensions implementing *distance learning* (MRSLVQ and LMRSLVQ (Schneider, Biehl, and Hammer 2009b)), *relational* methods (RRSLVQ (Hammer, Schleif, and Zhu 2011)), and *kernelized* ones (KRSLVQ (Hofmann, Gisbrecht, and Hammer 2012)) are proposed.

To summarize, extensive research has been conducted on extending LVQ algorithms to improve the training procedure and adapting to non-vectorial data and non-linear classification problems (Nova and Estévez 2014). While this is a substantial development, various aspects relevant to practical applications have not been focused on so far. In this thesis, I will present further extensions that ensure *privacy* and increase the *robustness* of LVQ models by introducing probabilistic outputs and enabling rejects of uncertain samples. Besides, I will explore the *sparsity* to reduce the model's complexity and propose the first framework for LVQ learning on *distributed* data sets.

To achieve those goals, I will exploit the property of GLVQ having a cost function and providing good results even for a small number of training samples. The last fact is since the prototypes are close to samples belonging to the same class and the decision boundary can be approximated well. Before focusing on these, I will introduce the benchmarks utilized to evaluate my contributions.

2.3 BENCHMARKS

In all chapters of this thesis, we will use public benchmarks to evaluate our novel approaches. We will briefly describe all benchmarks in this section

(a) Objects from the COIL-20 benchmark

(b) Examples from the MNIST benchmark

(c) Examples from the USPS benchmark

Figure 2.3: Examples of samples for COIL-20[1], MNIST[2], and USPS[3]

and will give the performances, measured as the classification accuracy, of the already introduced GLVQ models. Later in this thesis, we will deal with additional objectives and will compare our methods and their performance with these from the original versions as a baseline.

2.3.1 Description of all Benchmarks

In the following, we will give a short description of each benchmark.

- *COIL-20:* (Nene, Nayar, and Murase 1996) The Columbia Object Image Database Library contains gray-valued images of 20 objects. Each object was placed on a turntable and one picture was taken for every rotation by 5°, i. e. the data set consists of 72 images per object. Every picture was cropped and resized to 128×128 pixels and then normalized such that the intensity of the brightest pixel is 255. The objects are depicted in Figure 2.3(a).

- *Digit:* (Dua and Graff 2017) The Optical Recognition of Handwritten Digits data set contains 5620 pre-processed normalized binary bitmaps of handwritten digits from zero to nine. Each bitmap was scaled to 32×32 pixels and for each non-overlapping block of 4×4, the number of ones, on pixel, was counted. A matrix of 8×8 where each element is an integer in the range $[0, 16]$ was generated leading to the feature vector.

1 taken from https://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php
2 taken from https://commons.wikimedia.org/wiki/File:MnistExamples.png
3 taken from https://www.researchgate.net/figure/Sample-digits-images-from-the-United-States-Postal-Service-USPS-dataset_fig2_328787173

- *Image Segmentation:* (Dua and Graff 2017) The Image Segmentation data set consists of 2310 samples representing small patches from outdoor images, i. e. brick face, sky, foliage, cement, window, path, and grass. For each class, 330 small patches were randomly extracted. Each sample consists of 19 real-valued image descriptors gained from one of these patches.

- *Letter:* (Dua and Graff 2017) The Letter Recognition data set are pre-processed images from 20 different fonts of capital letters in the English alphabet. For each image, 16 primitive numerical attributes, such as statistical moments and edge counts, are scaled to fit into a range of integer values in $[0, 15]$.

- *MCE-Nose:* (Jiménez, Monroy, and Blanco 2011) The Multi-Chamber Electronic Nose data set consists of recordings from four odors like Acetone, Ethanol, Butane, and Gin. In the data set, the samples represent the output of four sensors where the chamber and the odor were switched every 20 seconds. This setup leads to a record where the samples are ordered.

- *MNIST:* (Lecun et al. 1998) The Modified National Institute of Standards and Technology data set represents monochrome images from handwritten digits from zero to nine. Each digit was pre-processed, centered, and resized to a fixed image size of 28×28 pixels. Some samples are depicted in Figure 2.3(b).

- *Motion Tracking:* (Anguita et al. 2013) The Human Activity Recognition Using Smartphones data set consists of pre-processed sensor signals. 30 volunteers performed six activities of daily living (walking, walking upstairs, walking downstairs, sitting, standing, and lying) while carrying a waist-mounted smartphone with embedded inertial sensors. Each sample results by applying noise filters and calculating variables from the time and frequency domain on 128 readings from the accelerometer and gyroscope. Due to the recording, the data is ordered, and at certain times the label change only when the next activity was performed.

- *USPS:* (Hull 1994) The United States Postal Service data set contains monochrome images of handwritten digits scanned from mails. Each image is centered, normalized, and resized to 16×16 pixels. The whole data set consists of letters and numbers because all the addresses are processed. However, we will use only the numbers. Some samples are depicted in Figure 2.3(c).

The characteristics, i. e. the number of samples, features, and classes, of all benchmarks are summarized in Table 2.1. For those benchmarks containing images, some examples are depicted in Figure 2.3. Two benchmarks, i. e. MCE-Nose and Motion Tracking, consist of records where the data has a time dependency. The labels change only if the odor is switched or a new activity is performed. This property is used later in Section 4.5 where we improve the robustness of a classifier for time-dependent data sets.

Table 2.1: Properties of all benchmarks

Data set	# samples	# features	# classes
COIL-20	1440	16 384	20
Digit	5620	64	10
Img. Seg.	2310	19	7
Letter	20 000	16	26
MCE-Nose	2257	4	4
MNIST	70 000	784	10
Mot. Track.	10 299	561	6
USPS	9298	256	10

2.3.2 Basic Classification Accuracy

We will briefly report the performances of all introduced methods on all benchmarks, as we will compare our novel against the vanilla versions. Since the new methods will also optimize additional objectives next to the classification accuracy, these scores are used as baselines. Here, we train GLVQ, GMLVQ, and LGMLVQ models on each benchmark. For those benchmarks which have more than 30 features, we reduced the dimensionality to 30 using Principle Component Analysis (PCA), i. e. all benchmarks except Image Segmentation, Letter, and MCE-Nose. In Table 2.2, the results are obtained from five times repeated 5-fold cross-validation. It reports the mean and the standard deviation for three numbers of prototypes per class, i. e. 1, 2, and 4. One can clearly observe that first metric learning improves the classification performance for all the benchmarks. Second, the local metrics outperform all other ones. In particular, this holds for Letter having the highest number of classes. For that one, the local relevance matrices of the LGMLVQ models enable a better separation of the classes than the single relevance matrix of the GMLVQ model.

Another observation is that for some benchmarks one prototype per class already performs well, and a higher number of prototypes does not improve the performance much. Therefore, we often use only one prototype for the models later, particularly when we use metric learning.

Table 2.2: Accuracy and standard deviation (in parenthesis) for GLVQ, GMLVQ, and LGMLVQ for different numbers of prototypes (PT) and all benchmarks.

method	Data set	one PT	two PTs	four PTs
GLVQ	COIL-20	0.899 (0.0176)	0.912 (0.0173)	0.936 (0.0159)
	Digit	0.916 (0.0083)	0.928 (0.0068)	0.937 (0.0072)
	Img. Seg.	0.836 (0.0116)	0.839 (0.0105)	0.845 (0.0099)
	Letter	0.688 (0.0094)	0.741 (0.0064)	0.793 (0.0065)
	MCE-Nose	0.593 (0.0277)	0.693 (0.0252)	0.769 (0.0320)
	MNIST	0.813 (0.0032)	0.852 (0.0038)	0.875 (0.0054)
	Mot. Track.	0.895 (0.0066)	0.900 (0.0063)	0.906 (0.0062)
	USPS	0.873 (0.0071)	0.907 (0.0064)	0.923 (0.0048)
GMLVQ	COIL-20	0.956 (0.0097)	0.962 (0.0116)	0.963 (0.0115)
	Digit	0.940 (0.0060)	0.943 (0.0052)	0.943 (0.0072)
	Img. Seg.	0.912 (0.0144)	0.927 (0.0155)	0.929 (0.0126)
	Letter	0.701 (0.0067)	0.737 (0.0082)	0.766 (0.0083)
	MCE-Nose	0.716 (0.0266)	0.738 (0.0228)	0.750 (0.0241)
	MNIST	0.852 (0.0030)	0.858 (0.0068)	0.865 (0.0078)
	Mot. Track.	0.909 (0.0058)	0.906 (0.0065)	0.907 (0.0063)
	USPS	0.909 (0.0062)	0.909 (0.0069)	0.914 (0.0069)
LGMLVQ	COIL-20	1.000 (0.0007)	0.999 (0.0012)	0.999 (0.0017)
	Digit	0.978 (0.0066)	0.982 (0.0049)	0.984 (0.0040)
	Img. Seg.	0.946 (0.0094)	0.952 (0.0098)	0.955 (0.0103)
	Letter	0.881 (0.0049)	0.918 (0.0063)	0.945 (0.0050)
	MCE-Nose	0.774 (0.0250)	0.869 (0.0277)	0.913 (0.0197)
	MNIST	0.943 (0.0054)	0.950 (0.0039)	0.955 (0.0038)
	Mot. Track.	0.931 (0.0039)	0.934 (0.0040)	0.937 (0.0038)
	USPS	0.961 (0.0051)	0.965 (0.0049)	0.967 (0.0050)

3 DIFFERENTIAL PRIVACY

Overview: This chapter will recap the notion of Differential Privacy (DP). We will explain some methods from the literature for which it is proven that they fulfill the definition of DP. To answer **RQ1**: *What is required to preserve personal privacy in LVQ models*, we first show in Section 3.3 how LVQ models leak information from samples in the training set and, thus, violate privacy. Later, in Section 3.4, we propose three novel training schemes that preserve personal privacy in LVQ models. Finally, we compare those with the vanilla variants of GLVQ to analyze the trade-off between performance and privacy.

Publications: Parts of this chapter are based on the following publications:

- Brinkrolf, Johannes, Kolja Berger, and Barbara Hammer (2017). "Differential Privacy for Learning Vector Quantization". In: *Proceedings of the Workshop New Challenges in Neural Computation (NC^2 2017)*. Ed. by Barbara Hammer, Thomas Martinetz, and Thomas Villmann, pp. 17–25. URL: https://www.techfak.uni-bielefeld.de/~fschleif/mlr/mlr_03_2017.pdf#page=17.
- Brinkrolf, Johannes, Kolja Berger, and Barbara Hammer (2018). "Differential private relevance learning". In: *26th European Symposium on Artificial Neural Networks, ESANN 2018, Bruges, Belgium, April 25-27, 2018*. URL: https://www.esann.org/sites/default/files/proceedings/legacy/es2018-119.pdf.
- Brinkrolf, Johannes, Christina Göpfert, and Barbara Hammer (2019). "Differential privacy for learning vector quantization". In: *Neurocomputing* 342, pp. 125–136. DOI: 10.1016/j.neucom.2018.11.095. URL: https://doi.org/10.1016/j.neucom.2018.11.095.

3.1 INTRODUCTION AND OVERVIEW

While obtaining well-performing models with regards to their accuracy is crucial to use them in real-world settings, other objectives need to be considered as well (Dwork, A. Smith, et al. 2017; Narayanan and Shmatikov 2008). One important consideration when learning from personal data is ensuring the *privacy* of individuals by making sure that no personal information gets available when using and publishing the model. This is not only a desirable design choice but also required by (European Commission 2020).

A model representation in the form of exemplars or prototypes, such as in LVQ, carries a high risk of revealing sensitive information about the used training data. The prototypes display typical feature values of the data since they are constructed by averaging samples provided in the training set. For simplified settings and in stationary states of the learning algorithms, the form of prototypes can be analyzed precisely. It can be shown that prototypes eventuate as centers of some samples assigned to their receptive field,

and metric parameters eventuate as directions that are similar to principal components (Biehl, Hammer, Schleif, et al. 2015). Hence, it seems likely that, at least in some (possibly extreme) settings of the training data, information about a single individual can be inferred from the models provided suitable auxiliary information becomes available. Note that we here address an individual not specified beforehand for which some auxiliary information gets available. Or someone who, for example, is a bit more atypical and influences one prototype more than an ordinary one. In this case, inferring new information from this individual might be more effortless. Even an answer to whether this individual was present during the training will be ranked as a leakage of privacy. Later, we will see that we can also provably preserve their privacy.

While such settings also occur for more complex models such as deep networks (Carlini et al. 2019; Hitaj, Ateniese, and Pérez-Cruz 2017), leakage of private information seems almost inevitable for interpretable models such as LVQ due to the fact that training data explicitly affects the location of prototypes to a substantial portion and in a 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 forty-five years (Dowell 1977). While encryption technologies enable secure data storage and, thereby, privacy because the data is not available for unauthorized users (Katz and Lindell 2007), the situation becomes more problematic whenever important information of the database is offered to the public. It occurs in settings when a machine learning model is trained based on possibly private data in a distributed network and when a machine learning model is released to the public.

Interestingly, various frameworks exist that go beyond classical encryption and enable public access to private information for specific settings without mitigating any sensitive information. One example is offered by homomorphic encryption (Armknecht et al. 2015; Martins, Sousa, and Mariano 2017). Here, encryption schemes are designed in such a way that they commute with arithmetic operations on the data. Consequently, learning based on these arithmetic operations becomes possible directly from encrypted data without the necessity of 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, decrypted 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 themselves (Du and Atallah 2001; Frikken 2010).

Yet, while these frameworks enable learning from private information without a direct access to the data of a single decrypted individual, they cannot prevent the risk that the resulting summarized model might reveal critical information if released to the public in decrypted form. Such leakage becomes possible as soon as the model itself is coupled with auxiliary data as available on the internet or via dedicated attacks (Dwork, A. Smith, et al. 2017). The question of possible attacks depends on the form of the resulting model and its relation to the individual data. There do exist approaches that derive bounds on the possible leakage of such models and which suggest substituting non-preferred ones with privacy-preserving surrogates (Ah-Fat

and Huth 2018).

One particularly popular formal approach focuses on how to design functions for model inference such that they do not reveal information about an individual even if coupled with auxiliary information. It comes under the notion of *Differential Privacy (DP)*. This formalism provably limits the possibility of receiving private information from published models no matter which auxiliary information or attacks are used (Dwork, McSherry, et al. 2006). Basically, DP formalizes the intuition that the amount of individual information contained in the models is strictly limited per query. This way, formal guarantees can be given about the privacy of individuals and the immunity of formalism to auxiliary data.

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 that enable an extension of popular ML tools to differentially private counterparts (Ji, Lipton, and Elkan 2014). Frequently, these rely on mathematical properties of DP 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 (Ji, Lipton, and Elkan 2014). Interestingly, such differentially private variants have been proposed for specific models, including, for example, naïve Bayes, Logistic Regression (LR), Decision Tree (DT), k-means, or Suport Vector Machine (SVM), and for general training mechanisms such as evolutionary algorithms or gradient schemes for optimization (Abadi et al. 2016a; Balle and Wang 2018; Dwork 2006; McSherry and Talwar 2007; J. Zhang et al. 2013). Despite the popularity of LVQ in sensitive application domains (Abadi et al. 2016a; Biehl 2017; Biehl, Bunte, et al. 2012; Losing, Hammer, and Wersing 2018), no effort has been done to investigate differentially private LVQ schemes, so far.

In this chapter, we investigate the previously introduced LVQ schemes as regards their preservation of privacy. We show in examples that the schemes are locally sensitive and hence are vulnerable to revealing private information. Due to this fact, we investigate three schemes which provably lead to DP: an extension of gradient-based training to a differentially private variant together with a differentially private initialization of the model (Abadi et al. 2016a; Balle and Wang 2018), a general scheme which is based on subsampling and aggregation (Dwork 2006), and a geometric variation thereof that has been proposed due to its better provable mathematical characteristics (Nissim, Raskhodnikova, and A. D. Smith 2007). In Section 3.5, we demonstrate that these schemes can be transferred to GLVQ. We investigate the behavior as regards the robustness to the choice of hyperparameters and the competitiveness of the resulting model accuracy compared to the vanilla, i. e. non-differentially private versions.

The roadmap of this chapter is as follows: In the next section, we introduce the formal notion of DP and the mathematical properties used in this chapter. The latter includes general differentially private schemes based on the addition of accurately defined noise, the composition of differentially private mechanisms, and differentially private initialization schemes for the models. Before addressing differentially private variants of LVQ, in Section 3.3, we will elucidate the question of whether LVQ models are vulnerable to revealing private information given auxiliary data. One condition yielding vul-

nerability is the fact that a model is sensitive to single samples 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 differentially private variants of LVQ if the model should be released to the public. Then, in Section 3.4, we introduce three different differentially private schemes for LVQ, which are based on three diverse generic mechanisms which can be transferred to the specific setting of LVQ. We evaluate these approaches as regards their sensitivity to hyperparameters in Section 3.5 to get a deeper insight into which parameters in particular as regards DP to use. Further, we evaluate their performance, i. e. accuracy, and obtained privacy in benchmarks.

3.2 DIFFERENTIAL PRIVACY

In the following, we briefly introduce the concept of Differential Privacy (DP). We recapitulate the notion of DP as well as a few popular differentially private strategies that will be of relevance for this chapter.

3.2.1 Notation of Differential Privacy

The notion of DP (Dwork 2011; Dwork, McSherry, et al. 2006; Dwork and Roth 2014) constitutes a strong standard for privacy guarantees for algorithms on aggregated databases. Thereby, it addresses the question of what additional information about an individual sample used for training a model can be extracted from the model given any auxiliary knowledge or data. One example of such settings occurs if a model reveals a previously unknown correlation of features for a specific sample that enables an adversary to retrieve one additional feature if they know the other. To avoid such problems, the idea is to limit the amount of individual information encoded within a given model.

Informally, DP requires that the output of a data analysis mechanism, such as an algorithm training 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 information from an individual. In Definition 4, we will see how this is defined mathematically and discuss it afterward

To measure the difference between two data sets and, as a consequence thereof, to identify an individual, we need the notion of *adjacent* data sets. Then, we define DP first and introduce specific differentially private mechanisms later. The definitions are adapted to our notation from Dwork and Roth (2014).

Definition 3 (Distance between Data Sets and Adjacency)**.** *Assume two data sets D, D' of samples (e. g. for training LVQ) are given. The* Hamming Distance *$\bar{d}_H(D, D')$ between two data sets is the number of entries on which D and D' differ, i. e.*

$$\bar{d}_H(D, D') = \left| D \setminus D' \cup D' \setminus D \right| .$$

Two data sets are adjacent *if they differ in only a single individual, thus* $\bar{d}_H(D, D') = 1$. *We denote* adjacency *of* D *and* D' *as* $\text{adj}(D, D')$.

Differential Privacy limits how the output of an operation like a machine learning algorithm can change if it is subject to adjacent data sets. Of course, a deterministic algorithm changes the output if a new sample is used during the training. In respect thereof, the definition deals with a so-called randomized algorithm $\mathcal{A}$ that outputs the result with a certain probability. This algorithm maps a given set of training data D to a model or to summary statistics revealed to the user. In other words, one can think of a noisy output where the algorithm does not return the exact same result for different runs. More formally:

Definition 4 (Differential Privacy)**.** *Assume* $\varepsilon > 0$ *and* $\delta \in [0, 1)$ *are given. A randomized algorithm* $\mathcal{A} \colon \mathcal{D} \to \mathbb{R}^k$ *satisfies* (ε, δ)-Differential Privacy *if and only if for all pairs of adjacent data sets* D *and* D'*, and for any subspace* S *of the algorithm's output range, it holds that*

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

If $\delta = 0$*, we say that* $\mathcal{A}$ *is* ε-differentially private.

Note that the notion of adjacency is symmetric and the definition still holds for interchanging D and D'. Hence, privacy with the choice $\varepsilon = \delta = 0$ would imply that a single example does not influence the output of the algorithm $\mathcal{A}$. 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 slight change in the distribution of possible algorithmic outcomes. Hence, it is not possible to observe a significant difference in the output of $\mathcal{A}$ if an adversary is allowed only a small number of observations.

Example 3. *We give a short example in a one-dimensional case and look at a statistical value of numbers, i. e. the mean value. Just reporting the exact mean of numbers in a set* $\mathcal{X}$ *would not lead to a differentially private algorithm since adding a new individual, call it* x*, to the set will move the mean slightly. Therefore, it will play a role in the outcome. More concretely, if someone might know the size of the set* $(N = |\mathcal{X}|)$ *and the means of both scenarios* (μ, μ')*, then* $x = (N+1)\mu' - N\mu$ *yields the value of the added sample. It would be more problematic to consider samples in* $\mathbb{R}^d$*. If one knows only the entry of a single dimension, i. e. one feature of a sample, one could conclude the impact of this sample on the mean and reconstruct the point. It is even worse in the case of discrete-valued features.*

The basic idea is that not an exact value is reported but instead a random point drawn from a probability distribution with a particular mean and a predefined scaling. In Figure 3.1, we plot some samples along the x-axis. Let • *are the samples of* $\mathcal{X}$*, a random sample drawn out of the Probability Density Function (PDF) plotted as* — *would be the result. If the individual* • *is added to the set, of course, the mean shifts and a different PDF emerges, here plotted as* — *. Since the PDFs are too similar, it is hard to determine whether one outcome is from the first or the second distribution. In the plot, the dashed line indicates a possible outcome, and the two likelihoods of both PDFs are marked. The privacy of each individual in the set is preserved because removing any other sample in* $\mathcal{X}$ *will provide the same argumentation.*

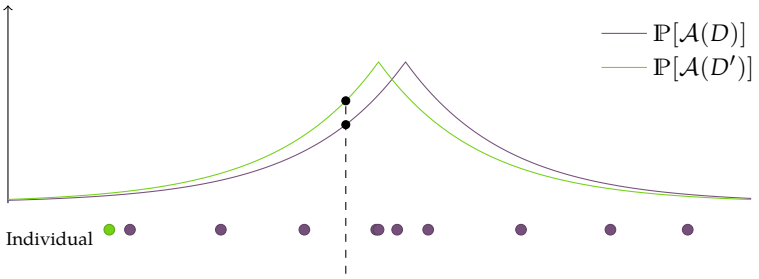


Figure 3.1: A one-dimensional example to illustrate DP. Adding an individual ● to the data set slightly shifts the mean to the left. Instead of returning the accurate mean, the output is a random sample drawn from the PDF. The dashed line indicates a potential outcome. The likelihood of both PDFs is too similar, such that one cannot determine whether the result is drawn from the PDF of the data set containing the individual (—) or the other (—). That is how the privacy of the individual, and all others, is preserved.

We briefly want to add some remarks about the definition of DP and paraphrase it. Rearranging the inequality for ε-Differential Privacy in Definition 4 gives us

$$\varepsilon \geq \log\left(\frac{\mathbb{P}[\mathcal{A}(D) \in S]}{\mathbb{P}[\mathcal{A}(D') \in S]}\right). \tag{3.1}$$

If a certain event is more likely under D than under D', the right-hand side is positive and vice versa negative. Because the definition of adjacent data sets are symmetric, ε-Differential Privacy ensures that for all adjacent D and D' the privacy budget is bounded by ε. The multiplicative $\exp(\varepsilon)$ in Definition 4 implies that zero-probable outcome on a given data set is also zero-probable on the adjacent one. The additive factor δ could be regarded as a *failure tolerance* of this constraint. In simple terms, the definition says that with a probability of at least $1 - \delta$ the outcome of a randomized algorithm $\mathcal{A}$ on D is (almost) equally likely to be observed on *every* adjacent data set, simultaneously. Thus, the outcome is invariant if any individual is removed from the data set.

Later, we will use one algorithm that consists of several steps. Hence, we would like to point out that the notion of 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 $\varepsilon = \sum_j \varepsilon_j$ (Dwork and Roth 2014). We state this for two mechanisms $\mathcal{A}_1$ and $\mathcal{A}_2$ in the following theorem.

Theorem 1. *Let $\mathcal{A}_1 \colon \mathcal{D} \to \mathbb{R}^k$ and $\mathcal{A}_2 \colon \mathcal{D} \to \mathbb{R}^{k'}$ be an ε_1- and ε_2-differentially private algorithms. Then, their combination, defined to be $\mathcal{A}_{1,2} \colon \mathcal{D} \to \mathbb{R}^k \times \mathbb{R}^{k'}$ by the mapping: $\mathcal{A}_{1,2}(\boldsymbol{x}) = (\mathcal{A}_1(\boldsymbol{x}), \mathcal{A}_2(\boldsymbol{x}))$ is $(\varepsilon_1 + \varepsilon_2)$-differentially private.*

We refer to Dwork and Roth (2014, Theorem 3.14) for the proof[1]. Theorem 1 can be applied repeatedly to obtain the proof for a limited number of independent mechanisms $\mathcal{A}_1, \ldots, \mathcal{A}_m$. Also, a generalization of this theorem to (ε, δ)-Differential Privacy is given by Dwork and Roth (2014) and is stated in the following theorem.

1 A to our notation adapted version can be found in Appendix B.1.1

Theorem 2 (Composition Theorem). *Let $\mathcal{A}_j \colon \mathcal{D} \to \mathbb{R}^{k_j}$ be $(\varepsilon_j, \delta_j)$-differentially private algorithms for $j \in \{1, \ldots, m\}$. Then, if $\mathcal{A}_{[m]} \colon \mathcal{D} \to \prod_{j=1}^{m} \mathbb{R}^{k_j}$ is defined as $\mathcal{A}_{[m]}(\boldsymbol{x}) = (\mathcal{A}_1(\boldsymbol{x}), \ldots, \mathcal{A}_m(\boldsymbol{x}))$, then $\mathcal{A}_{[m]}$ is $(\sum_{j=1}^{m} \varepsilon_j, \sum_{j=1}^{m} \delta_j)$-differentially private.*

We refer to Dwork and Roth (2014, Appendix B) for the proof. Here, we want to point out that Theorems 1 and 2 provide the way to construct -differentially private ML methods in a modular design. If all the components of a mechanism are -differentially private, then so is their composition.

There exist several approaches satisfying ε-Differential Privacy, including the *Laplace Mechanism* (Dwork, McSherry, et al. 2006). The latter deals with functions $f \colon \mathcal{D} \to \mathbb{R}^k$ from the data sets' domain to vectorial outputs. It adds symmetric and scaled noise to each dimension of the result. The magnitude of the required noise depends on the so-called *Global Sensitivity* of f. It refers to the maximum difference between the outputs of two adjacent data sets, or, more formally:

Definition 5 (Global Sensitivity). *The* Global Sensitivity *of $f \colon \mathcal{D} \to \mathbb{R}^k$ is defined as*

$$\Delta_{\mathrm{GS}}(f) = \max_{D, D' \colon \operatorname{adj}(D, D')} \left\| f(D) - f(D') \right\| \tag{3.2}$$

measured in any norm $\|\cdot\|$ and any two data sets D, $D' \subset \mathcal{D}$.

Note that for all possible data sets D and D', the Global Sensitivity is the same. Thus, it is independent of the data set itself and is conditional on only the function. This directly affects the design of an ML mechanism because the magnitude of the noise can be determined for a fixed domain for all possible data sets.

Similar to the Global Sensitivity, the *Local Sensitivity* is defined. Here, the maximum difference between one fixed data set and all adjacent ones is considered. Formally:

Definition 6 (Local Sensitivity). *The* Local Sensitivity *of $f \colon \mathcal{D} \to \mathbb{R}^k$ for one fix data set $D \subset \mathcal{D}$ is defined as*

$$\Delta_{\mathrm{LS}}(f) = \max_{D' \colon \operatorname{adj}(D, D')} \left\| f(D) - f(D') \right\|. \tag{3.3}$$

It has been shown that releasing $f(D)$ with noise magnitude proportional to $\Delta(\mathrm{LS})(f)$ is not differentially private because the noise magnitude itself reveals information about the database (Dwork 2006). The next example of returning the median will illustrate it. However, the Local Sensitivity is used later in Section 3.2.7.

Example 4. *We will shortly explain why a mechanism cannot simply add noise scaled to the Local Sensitivity to guarantee DP. Looking at the median for a set of only 0 and 10. In a extreme case, the Local Sensitivity of a set D is exactly zero if for an even number n the set of size $n + 1$ consists of $1 + n/2$ zeros, i. e.*

$$D = \{\underbrace{0, \ldots, 0}_{1+n/2}, \underbrace{10, \ldots, 10}_{n/2}\}.$$

Removing or adding any element from/to the set does not change the median if it is defined to break ties in favor of the smaller value. In contrast, looking at D' of size n containing $n/2$ zeros, i. e.

$$D' = \{\underbrace{0, \ldots, 0}_{n/2}, \underbrace{10, \ldots, 10}_{n/2}\},$$

the Local Sensitivity on the median of D' is 10. If we consider an algorithm that computes the result via adding noise scaled to the Local Sensitivity (we will see some in the subsequent sections), an adversary may be able to guess rather accurately whether D or D' is the right set. This is due to the fact that the hole probability mass is concentrated on the single point 0 when the set is D. Thus, if the result is not 0, the true set has to be D'. This destroys the idea of DP that no information from the data is gained by observing the outcome of a mechanism.

3.2.2 Laplace Mechanism

After defining the Global Sensitivity, we can introduce the *Laplace Mechanism*. As its name suggests, the mechanism computes the outcome of a function and perturbs each coordinate with noise drawn from the Laplace distribution. The next definition specifies it more.

Definition 7 (Laplace Mechanism)**.** *Given a function $f\colon \mathcal{D} \to \mathbb{R}^k$ the* Laplace Mechanism *is defined as*

$$\mathcal{A}_f(D) = f(D) + (Y_1, \ldots, Y_k)^\top \tag{3.4}$$

for a given database $D \subset \mathcal{D}$, where Y_i are i. i. d. random variables drawn from the Laplace distribution $\text{Lap}\,(0, \Delta_{GS}(f)/\varepsilon)$*, whereby the Global Sensitivity ($\Delta_{GS}(f)$) is measured based on the ℓ_1 norm.*

We refer to Dwork and Roth (2014, Theorem 3.6) for the proof[2].

3.2.3 Counting Queries

We will use the Laplace Mechanism in so-called *counting queries* where we want to know how many samples have a specific property, e. g. how many samples in the training sets belong to a particular class. This will lead to a very small Global Sensitivity of 1 because adding or removing one sample from the data set will change the outcome by maximal 1. In the example of counting all samples of one class, the number either remains the same or is raised or reduced by 1 if the sample contained in an adjacent data set belongs to that particular class.

A sequence of m counting queries can be viewed as a vector-valued query. Without any further information about the set, one sample might change every count. This gives us a Global Sensitivity of m. Thus, we achieve ε-Differential Privacy by adding noise drawn from the Laplace distribution and scaled by m/ε to the true result of each query.

2 The corresponding Theorem 6 and a proof adapted to out notation of the theorem can be found in Appendix B.1.2.

In a specific case where the queries are structurally disjoint, the counting can be more accurate with so-called *histogram queries*. For example, we want to know the number of samples for each class at once Then, noise scaled to $1/\varepsilon$, instead of c/ε (c is the number of classes), is sufficient because the data set is partitioned into disjoint cells and one individual can affect the count in exactly one cell. We will use this idea in Section 3.4.1 in the initialization step of training GLVQ models which calculates the means of each class.

The counting and histogram queries are two established methods using the Laplace Mechanism 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 the Global Sensitivity of f might be complicated to bound. Further, a random vector leads to a high scaling caused by the ℓ_1 norm as we will see later at the end of Section 3.3. Next, we will briefly introduce three methods where the ℓ_2 norm is used for calculating the Global Sensitivity.

3.2.4 Gaussian Mechanism

There exist several methods for substituting the ℓ_1 norm with the ℓ_2 norm in the Global Sensitivity (see Definition 5). For example, by Dwork and Roth (2014) the *Gaussian Mechanism* has been introduced. The basic idea of adding noise stays the same. Instead, the Laplace distribution is substituted by the Gaussian one. If $\varepsilon, \delta \in (0, 1)$, they show that for a given, fixed δ a standard deviation

$$\sigma \geq \frac{1}{\varepsilon} \Delta_{GS}(f) \sqrt{2 \log(1.25/\delta)} \tag{3.5}$$

is sufficient (Dwork and Roth 2014, Theorem A.1) to obtain a (ε, δ)-differentially private mechanism.

Liu (2019) gives a mechanism with a tighter bound and which does not constrain ε to be smaller than 1 as the proof in Dwork and Roth (2014, Appendix A) does. It is dubbed as the *generalized Gaussian Mechanism* and is (ε, δ)-differentially private if the standard deviation of the Gaussian distribution fulfills

$$\sigma \geq \frac{1}{2\varepsilon} \Delta_{GS}(f) \left(\sqrt{(\Phi^{-1}(\delta/2))^2 + 2\varepsilon} - \Phi^{-1}(\delta/2) \right) \tag{3.6}$$

where $\Phi^{-1}(x)$ denotes the inverse Cumulative Distribution Function (CDF) of the Gaussian distribution (Liu 2019). In addition, the author proves that there exists no ε-differentially private mechanism, i. e. $\delta = 0$, for any ℓ_p norm with an integer $p > 1$. So, if the norm for calculating the Global Sensitivity is changed, a small positive value for δ has to be set beforehand.

A third version, the *analytic Gaussian Mechanism*, is proposed by Balle and Wang (2018) with a more handy method for calculating a sufficient variance for the Gaussian distribution. They prove the following theorem which we adapted to our notation.

Theorem 3 (Analytic Gaussian Mechanism). *Let $f : \mathcal{D} \to \mathbb{R}^k$ be a function with $\Delta_{GS}(f)$ measured in ℓ_2. For any $\varepsilon > 0$ and $\delta \in (0, 1)$, the Gaussian Mechanism*

Algorithm 3.1: Analytic Gaussian Mechanism

Inputs: data set D, function f, Global Sensitivity $\Delta_{\text{GS}}(f)$, privacy budget $\varepsilon > 0$, and $\delta \in (0, 1)$
$\delta_0 \leftarrow \Phi(0) - \exp(\varepsilon)\Phi\left(-\sqrt{2\varepsilon}\right)$
if $\delta \geq \delta_0$ **then**
    Define $B^+(v) = \Phi\left(\sqrt{\varepsilon v}\right) - \exp(\varepsilon)\Phi\left(-\sqrt{\varepsilon(v+2)}\right)$
    Compute $v^* = \sup\{v \in \mathbb{R}_{\geq 0} \mid B^+(v) \leq \delta\}$
    $\alpha \leftarrow \sqrt{1 + v^*/2} - \sqrt{v^*/2}$
else
    Define $B^-(u) = \Phi\left(-\sqrt{\varepsilon u}\right) - \exp(\varepsilon)\Phi\left(-\sqrt{\varepsilon(u+2)}\right)$
    Compute $u^* = \inf\{u \in \mathbb{R}_{\geq 0} \mid B^-(u) \leq \delta\}$
    $\alpha \leftarrow \sqrt{1 + u^*/2} + \sqrt{u^*/2}$
end if
$\sigma \leftarrow \Delta_{GS}(f)\alpha/\sqrt{2\varepsilon}$
Return: $f(D) + \mathcal{N}\left(0, \sigma^2\mathbb{I}\right)$

$\mathcal{A}(\boldsymbol{x}) = f(\boldsymbol{x}) + \boldsymbol{Y}$ *with* $\boldsymbol{Y} \sim \mathcal{N}\left(0, \sigma^2\mathbb{I}\right)$ *is* (ε, δ)*-differentially private if and only if*

$$\Phi\left(\frac{\Delta_{\text{GS}}(f)}{2\sigma} - \frac{\varepsilon\sigma}{\Delta_{\text{GS}}(f)}\right) - \exp(\varepsilon)\Phi\left(-\frac{\Delta_{\text{GS}}(f)}{2\sigma} - \frac{\varepsilon\sigma}{\Delta_{\text{GS}}(f)}\right) \leq \delta.$$

We refer to Balle and Wang (2018, Theorem 8) for the proof. Instead of solving the inequality by σ in Theorem 3 using the inverse of the CDF, they propose to search for such a σ by the programming paradigm of binary search. Their algorithm is summarized in Algorithm 3.1. The correctness of it is provided by Balle and Wang (2018, Theorem 9).

However, using those mechanisms in sequential algorithmic steps, as is required in the training of a model, leads to weaker privacy guarantees using the Notation of Differential Privacy. Therefore, several methods directly relying on typical machine learning mechanisms have been proposed. A promising one adds DP to gradient techniques which we will introduce in the next section.

3.2.5 Differentially Private Stochastic Gradient Descent

An adapted differentially private method for *Stochastic Gradient Descent (SGD)* has been introduced by Abadi et al. (2016a). Essentially, the authors propose a variant of gradient descent with private operations only. It reflects mini-batch optimization techniques as they are well-known for the optimizing non-convex functions in machine learning.

An outline of the algorithm is given in Algorithm 3.2. It assumes a loss function $\mathcal{L}(\theta)$ that is optimized to obtain the model parameters θ. The algorithm performs T gradient descent steps on random subsets of size L each for a given number of steps T, a learning rate η_t which may change for each step, a batch size L, and a clipping bound C. First, a subset is taken from the training set with sample probability $q = L/n$ where n is the number of training samples. For each sample in this subset, the gradient is computed (see

Algorithm 3.2: Differentially Private Stochastic Gradient Descent

1: **Inputs:** samples $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, loss function $\mathcal{L}(\theta) = \sum_{i=1}^{n} \mathcal{L}(\theta, \boldsymbol{x}_i)$.
2: **Parameters:** number of steps T, learning rates η_t, noise scale σ, batch size L, gradient clipping bound C.
3: $\theta_0 = \mathbf{0}$
4: **for** $t \in [1, \ldots, T]$ **do**
5: Take subset L_t from all samples with sampling probability L/n
6: **for all** $\boldsymbol{x} \in L_t$ **do**
7: compute $g_t(\boldsymbol{x}) \leftarrow \nabla_{\theta_t} \mathcal{L}(\theta_t, \boldsymbol{x})$ ▷ compute gradient
8: $\overline{g}_t(\boldsymbol{x}) \leftarrow g_t(\boldsymbol{x}) / \max\left(1, \frac{\|g_t(\boldsymbol{x})\|_2}{C}\right)$ ▷ clip gradient
9: **end for**
10: $\hat{g}_t \leftarrow \frac{1}{L}\left(\sum_{\boldsymbol{x} \in L_t} \overline{g}_t(\boldsymbol{x}) + \mathcal{N}\left(0, \sigma^2 C^2 \boldsymbol{I}\right)\right)$ ▷ add noise
11: $\theta_t \leftarrow \theta_{t-1} - \eta_t \hat{g}_t$ ▷ descent step
12: **end for**
13: **Output:** θ_T

Line 7). Then, all gradients are normalized by the ℓ_2 norm such that their lengths are at most as large as the clipping parameter C (see Line 8). To guarantee a differentially private gradient, the sum of all gradients is perturbed by noise drawn from the Gaussian distribution $\mathcal{N}\left(0, \sigma^2 C^2\right)$ (zero mean and $\sigma^2 C^2$ as the variance for each dimension) (see Line 10). Finally, an ordinary gradient descent step with the noisy gradient and with η_t as the step size is performed (see Line 11). These commands are repeated T times.

In a practical environment, the commands inside the loop (from Line 5 to Line 10) can be put into a subroutine on a protected server that has access to the sensible training samples. This subroutine can return the privatized gradient. Thus, the descent step can be performed on any public computer because post-processing does not violate its privacy (Dwork and Roth 2014, Proposition 2.1). The theorem by Abadi et al. (2016a) gives the required standard deviation for the Gaussian distribution in Line 10.

Theorem 4. *There exist constants c_1 and c_2 such that given the sampling probability $q = L/n$ and the number of steps T, for any $\varepsilon < c_1 q^2 T$, Algorithm 3.2 is (ε, δ)-differentially private for any $\delta > 0$ if*

$$\sigma \geq c_2 \frac{q\sqrt{T \log(1/\delta)}}{\varepsilon}. \tag{3.7}$$

We refer to Abadi et al. (2016b, Appendix) of the full paper version for more details about both constants (c_1 and c_2) and the proof. The main remark is that the standard deviation of the Gaussian distribution is scaled linearly by the clipping parameter C as well the sample probability q. Since one sample from the training set is selected with this probability for each batch, it influences the gradient only with this probability. If so, this sample only affects the gradient at most by the size of the clipping parameter, measured in the ℓ_2 norm.

One drawback of this method is that the number of iterations has to be set beforehand. In the training scheme, this might lead to two disadvantages. On the one hand, if the number of steps is too low, a worse optimum is found

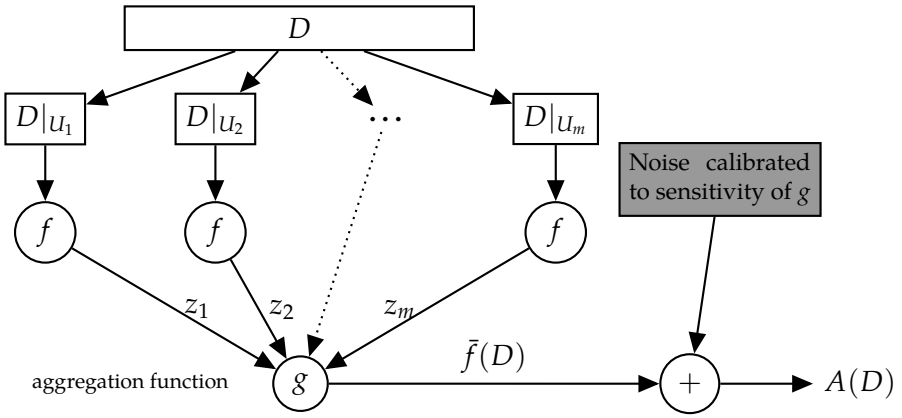


Figure 3.2: Subsample and Aggregate framework. U_i, $i \in \{1, \ldots, m\}$ are random, pairwise disjoint, subsets of $\{1, \ldots, |D|\}$ of size $|D|/m$ and $D|_U \subset D$ with indices in U. The function f is applied to each subset separately. The output of the algorithm is perturbed by adding noise scaled by the sensibility of the aggregation function g.

because the gradient descent stops too early, and the minimum is not reached. On the other hand, if T is too large, the magnitude of the noise in each step is larger than essential because the noise is proportional to $\sqrt{T}$. So, in each step, the direction is a bit more random. We will now consider two other mechanisms that privatize the output of a function by averaging.

3.2.6 Subsample and Aggregate

The *Subsample and Aggregate (SAA)* framework (Dwork and Roth 2014) can deal with functions that pose high global sensitivities or which sensitivities are hard to bound. The basic idea is to randomly divide the data set into m small batches and apply the function on each subset. Then, a so-called aggregation function g is used. It takes all outputs z_i for $i \in \{1, \ldots, m\}$ of f on each batch and aggregates them with a suitable mechanism. For geometric frameworks, such as prototypes, aggregation can be based on geometric averages, for example. To guarantee DP noise calibrated to the aggregation function g is added at this stage. This framework is depicted in Figure 3.2.

If disjoint subsets are used, one sample can only affect at most one batch and, therefore, the output of $f(D|_{U_i})$ for just one i. Thus, changing the data set by any sample can change at most a single input of g. Hence, the amount of noise is small compared to the Laplace Mechanism for the full data and becomes even smaller for a larger number of subsets. Note that the scaling of the noise is independent of m, so the mechanism adds the same amount of noise to the aggregation for small and large m. This results in a smaller ratio and less disturbance for larger subsets. The effect of m is analyzed in the experiments on different benchmarks in Section 3.5.

3.2.7 Center of Attention

Another variant of the SAA approach is introduced by Nissim, Raskhodnikova, and A. D. Smith (2007). It can handle functions with a high Global Sensitivity, like the median as seen in Example 4. To recap, the Global Sensitivity of the median on a bounded interval $\mathcal{D} = [0, M]$ is M. However, in typical scenarios, the difference between two results of adjacent data sets is much smaller since: Having a sorted set D (in ascending order) of size $n+1$ with an even number n like in Example 4, the $\Delta_{\mathrm{LS}}(\text{median}) = max(x_m - x_{m-1}, x_{m+1} - x_m)$ with $m = n/2$. Since using the Local Sensitivity to calibrate the noise is not differentially private, the authors propose a specific method that bounds the Local Sensitivity smoothly. They first define the admissible noise distribution (confer Nissim, Raskhodnikova, and A. D. Smith (2007, Definition 2.4)) as follows:

Definition 8 (Admissible Noise Distribution)**.** *A probability distribution h on $\mathbb{R}^d$ is (α, β)-admissible if, for $\alpha = \alpha(\varepsilon, \delta)$, $\beta = \beta(\varepsilon, \delta)$, the following two conditions hold for all $\|\Delta\| \leq \alpha$ and $\lambda \leq \beta$, and for all subsets $S \subseteq \mathbb{R}^d$:*

$$\textit{Sliding Property:} \quad \mathop{\mathbb{P}}_{Z\sim h}[Z \in S] \leq e^{\varepsilon/2}\,\mathbb{P}[Z \in S + \Delta] + \frac{\delta}{2} \tag{3.8}$$

$$\textit{Dilation Property:} \quad \mathop{\mathbb{P}}_{Z\sim h}[Z \in S] \leq e^{\varepsilon/2}\,\mathbb{P}[Z \in e^{\lambda} \cdot S] + \frac{\delta}{2} \tag{3.9}$$

where $S + \Delta = \{z + \Delta : z \in S\}$ and $e^{\lambda} \cdot S = \{e^{\lambda} \cdot z : z \in S\}$.

Using this definition, they propose the following lemma (confer Nissim, Raskhodnikova, and A. D. Smith (2007, Lemma 2.5)) that we adapt to our notation:

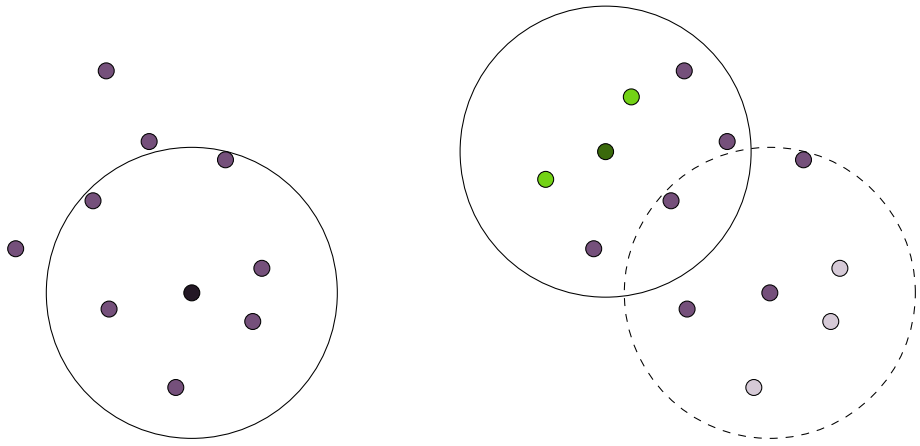
Lemma 1. *Let h be an (α, β)-admissible noise probability density function, and let Z be a random variable sampled according to h. For a function $f : \mathcal{D} \to \mathbb{R}^k$, let $s : \mathcal{D} \to \mathbb{R}$ be a β-smooth upper bound on the Local Sensitivity of f. Then the mechanism*

$$\mathcal{A}(D) = f(D) + \frac{s(D)}{\alpha} Z \tag{3.10}$$

is (ε, δ)-differentially private.

Besides, they provide proper values for α and β for the Laplace and the Gaussian distributions, such that these probability functions are (α, β)-admissible. In addition, in Claim 3.3, they give the *smooth sensitivity* of the median function.

We will briefly recap another mechanism called *Center of Attention (COA)* that we use later as an aggregation function dealing with points in any metric space. Here, we will restrict to the ℓ_2 norm in $\mathbb{R}^d$. The basic idea is that the result of the function is one point from a set representing the center. DP is ensured by disturbing this point. The scaling of the aggregation function is again its sensitivity that uses the so-called *t-radius*: Let $\mathcal{Z}$ be a set of points $\{z_1, \ldots, z_m\}$ with $z_i \in \mathcal{D}$, e. g. some outputs from a function on different subset. For a point $\boldsymbol{r} \in \mathcal{D}$ the t-radius $r(\boldsymbol{r}, t)$ is defined by the distance to the tth nearest neighbor of $\boldsymbol{r}$ in $\mathcal{Z}$. The center of attention $g(\mathcal{Z})$ is then the point



(a) first setting (b) after affecting m points

Figure 3.3: Illustration for Center of Attention in Example 5. Left: All points in a set $\mathcal{Z}$ of size ten are plotted. Their center of attention is colored darker, and the ball containing at least seven points is drawn as a circle. Right: After changing three points in $\mathcal{Z}$ a new center of attention is selected.
If an adjacent data set is used for training, not more than $\sqrt{m}$ points in $\mathcal{Z}$ are affected. The number t_0 is chosen so that if $\sqrt{m}$ points are removed from $\mathcal{Z}$, a majority of the remaining points is contained inside the ball. Considering an adjacent data set, the ball has at least one point in common. See a full description in Example 5.

in $\mathcal{Z}$ with the smallest t_0-radius, where $t_0 = \lfloor (m+\sqrt{m})/2 + 1 \rfloor$. It is shown by Nissim, Raskhodnikova, and A. D. Smith (2007) that adding Gaussian noise proportional to $\max_{l \geq 0} r^{(\mathcal{Z})}(t_0 + (l+1))$ to the center of attention is sufficient to guarantee DP, where $r^{(\mathcal{Z})}(t)$ is the minimum t-radius of any point in $\mathcal{Z}$.

The minimum can be calculated efficiently by computing all pairwise distances and sorting these distances for each point in $\mathcal{Z}$. Here, the idea is that one sample $\boldsymbol{p}$ from the training set occurs in less than $\sqrt{m}$ subsets. By choosing this t_0-radius, there exists at least one point $\boldsymbol{z}_i$ inside the ball from a subset where $\boldsymbol{p}$ does not occur that will be in t_0'-radius resulting in a run with the adjacent data set $D \setminus \{\boldsymbol{p}\}$. We will illustrate this fact in Example 5.

Example 5. *Looking at an example with $m = 10$ bins, e.g. having ten different positions of a prototype belonging to one class from 10 random subsets: By construction, every sample in the training set occurs in a maximum of three bins. Then, we set $t_0 = \lfloor (10+\sqrt{10})/2 + 1 \rfloor = 7$. So, we are looking for the Center of Attention having the smallest distance to its six closest neighbors. In Figure 3.3(a), we have ten points and draw a circle containing seven points with the COA as its center. In case having an adjacent data set, we know that, in the worst case, up to three points shift outside the circle. Even then, four out of seven points are still inside the circle. The updated circle of this adjacent data set must have at least one point in common. Figure 3.3(b) depicts the old position of the changed points as ● and their new positions as ●. The chosen center of attention is perturbed by a random noise scaled with twice this radius to satisfy DP.*

Table 3.1: All differentially private mechanisms used later to ensure DP for GLVQ.

Mechanism	Distribution	Noise scale
Laplace Mechanism	Laplace	$\beta = \frac{\Delta_{GS}(f)}{\varepsilon}$, $\Delta_{\mathrm{GS}}(f)$ based on ℓ_1
Counting Query	Laplace	$\beta = \frac{1}{\varepsilon}$
Histogram Query	Laplace	$\beta = \frac{1}{\varepsilon}$
Gaussian Mechanism	Gaussian	$\sigma \geq \sqrt{2\log{}^{1.25}/_{\delta}}\frac{\Delta_2(f)}{\varepsilon}$
generalized Gaussian Mechanism	Gaussian	$\sigma \geq \frac{1}{2\varepsilon}\Delta_{GS}(f)$ $\left(\sqrt{(\Phi^{-1}({}^{\delta}/_{2}))^2 + 2\varepsilon} - \Phi^{-1}({}^{\delta}/_{2})\right)$
analytic Gaussian Mechanism	Gaussian	$\delta \geq \Phi\left(\frac{\Delta_{\mathrm{GS}}(f)}{2\sigma} - \frac{\varepsilon\sigma}{\Delta_{\mathrm{GS}}(f)}\right)$ $- \exp(\varepsilon)\Phi\left(-\frac{\Delta_{\mathrm{GS}}(f)}{2\sigma} - \frac{\varepsilon\sigma}{\Delta_{\mathrm{GS}}(f)}\right)$
differentially private SGD	Gaussian	$\sigma \geq c_2\frac{q\sqrt{T\log(1/\delta)}}{\varepsilon}$
Subsample and Aggregate	depend on aggregation g	$\beta, \sigma \sim \frac{\Delta_{GS}(g)}{\varepsilon}$
Center of Attention	Gaussian	$\sigma \geq \max_{l\geq 0} r^{(\mathcal{Z})}(t_0 + (l+1))$

To summarize all mechanisms that we will use later to ensure privacy for GLVQ, we list them in Table 3.1. Before we introduce our methodologies, we will analyze how LVQ violates the privacy of samples in the training set, in the next section.

3.3 HOW VANILLA LVQ VIOLATES PRIVACY

Before answering **RQ1**, we 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 sample can leak. As my first contribution in the intersection of DP and LVQ, we will show that LVQ is not differentially private. Therefore, we demonstrate that LVQ is prone to at least two qualitatively different changes of the outcome that can be caused

by adding a single sample, namely significant changes of the distribution of the prototype's position, and significant changes of the data assignment to prototypes.

3.3.1 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 (see Section 2.2). As has been shown in (Biehl, Hammer, Schleif, et al. 2015; Hammer and Villmann 2002a), 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 (Dwork 2006). In particular, if an adversary knows the coordinates of all points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ from one class except for one point $\boldsymbol{x}'$ as well as the corresponding prototype $\boldsymbol{w}$, they are able to perfectly reconstruct $\boldsymbol{x}'$ via the formula

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

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

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

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

3.3.2 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 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 one 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 sample.

To illustrate this point, we have performed statistical test according to the following setups (see Figure 3.4(a)):

- *Extreme outlier:* We generate one data set with 100 samples from two classes, 50 per class, with points sampled from Gaussian distributions

with means $(-10, 0)$ and $(10, 0)$, respectively, and variance 1 in both dimensions. As the second data set, we create a copy of the data set and add one outlier sampled at mean $(10, 10)$ with again a variance 1. We sample these data set 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 large distance between the outlier and the class mean. We repeat the same experiment but reduce the distance of the outlier and set its mean to $(10, 5)$. We still observe a shift of the prototype on the y-axis for the second class by 0.0843 with p-value 2.2345×10^{-16}.

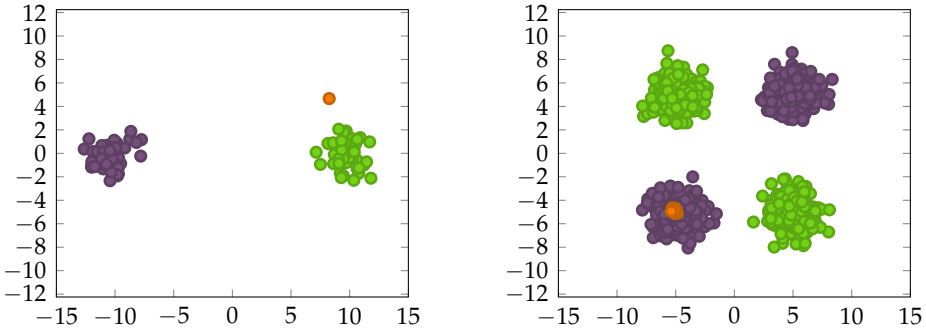
- *Weak outlier:* Setting the outlier mean to $(10, 2)$ results in a shift by 0.0368 and a p-value of 4.4007×10^{-4}.

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 in 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.

3.3.3 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 information about its input. A natural cure in the DP framework, as proposed by the Laplace Mechanism, is to add noise proportional to the Global Sensitivity of the learning algorithm. However, as we will see, this is not a feasible solution in the case of GLVQ since it 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 between 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 that are regarded as equally good by the algorithm, can be disturbed by adding a few additional points. We will show that this is the case using the noisy XOR problem (see Figure 3.4(b)):

- *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 1 while the bottom left and top right cluster belongs to class 2. We generate data sets according to the following experimental conditions: 1200 samples in total (300 in each cluster) sampled randomly at means $(10, 10)$, $(-10, 10)$, $(10, -10)$, and $(-10, -10)$ with



(a) Two clusters with a single outlier

(b) Asymmetrical XOR

Figure 3.4: Two settings with potential privacy leaks: for the toy settings pictured in Figure 3.4(a) and Figure 3.4(b), we compare GLVQ results on the data sets comprised of only the purple and green points to results on the data sets augmented by the orange points.

standard deviation 1. This is the "balanced" condition, the first data set. As the second data set, we create a copy of this data set and add 10 samples in the bottom left cluster. This is the "unbalanced" condition. We train an GLVQ 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 for 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.

Hence, symmetry breaking of the 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:

1. The result of LVQ and also of GLVQ is sensitive to single samples and there do exist settings where leakage of private information is possible, provided an adversary has access to auxiliary data.
2. It seems that GLVQ cannot easily be combined with differentially private mechanisms that rely on the Global Sensitivity of the algorithm itself, since, as shown in the last example, the Global Sensitivity cannot 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 propose mechanisms for differentially private GLVQ models that do not rely on privatizing a single already trained model. Instead, we will look at privatizing optimization as well as using two different Subsample and Aggregate mechanisms to exploit well-structured data sets that have low Local Sensitivity.

3.4 ENSURING PRIVACY FOR LEARNING VECTOR QUANTIZATION

In the last section, we have seen how GLVQ violates privacy. Finally, we start to answer the **RQ1** by showing how we change the training of GLVQ models to achieve differentially private ones using the mechanisms introduced in Section 3.2.1. We transfer the differentially private SGD (see Section 3.2.5) as well as two mechanisms of the SAA framework (see Section 3.2.6) to GLVQ. For the latter ones, 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 our analysis to GLVQ models with one prototype per class only. This is a limitation of the setting, albeit there exist a couple of applications where one prototype per class is sufficient (Biehl 2017; Schneider, Biehl, and Hammer 2009a). In Section 2.3.2, we have already shown that more prototypes per class do not enhance the result much for our benchmarks. The restriction of our analysis to one prototype per class is due to the circumstance that a few operations like initialization and merging of prototypes are much easier for this setting, hence easier to exchange by differentially private variants.

3.4.1 Privacy via differentially private Stochastic Gradient Descent

In the following, we describe how we change the gradient training of a GLVQ model to obtain a differentially private 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 differentially private variant. Essentially, the scaled stochastic gradient descent used for GLVQ is substituted by a differentially private variant as already introduced in Section 3.2. We refer to this method as *SGD*. The flowchart in Figure 3.5 visualized the training scheme. All steps required access to the data are shaded, and concerned functions have to be replaced by a differentially private one.

First step – Initialization of prototypes: As indicated above, we consider one prototype per class initialized by the class center as in vanilla LVQ. These are calculated based on the sum of all samples of each class and the number of class members:

$$\boldsymbol{w}_k = \frac{1}{N_k} \sum_{i:\, \mathrm{c}(\boldsymbol{x}_i)=k} \boldsymbol{x}_i \tag{3.11}$$

for each class $k \in \{1, \ldots, c\}$. These operations can directly be enhanced to differentially private versions based on the Laplace Mechanism as follows:

- *Cardinalities of classes:* The cardinalities of the classes are given by the function $f\colon \mathcal{D} \to \mathbb{N}^c$, $f(D) = (N_1, N_2, \ldots, N_c)$. As we already debated this function as a histogram query in Section 3.2.2, its sensitivity $\Delta_{\mathrm{GS}}(f)$ is equal to 1 because adding or removing one sample in the data set changes only the output of one N_k by 1. Hence, the Laplace Mechanism according to the sensitivity equal to 1 can render this computation differentially private.

- *Sum of points:* The sum of all points in each class is given by the function $g: \mathcal{D} \to \mathbb{R}^{c \cdot d}$, $g(D) = \left(\sum_{i:\ c(x_i)=1} \boldsymbol{x}_i, \ldots, \sum_{i:\ c(x_i)=c} \boldsymbol{x}_i\right)$. 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 $\Delta_{\mathrm{GS}}(g) = d$. One individual of an adjacent data set can change only the output of one sum at least by one in each dimension measured in the ℓ_1 norm because the classes are disjoint sets. This worst case occurs if a new sample $\boldsymbol{x} = \mathbf{1} \in \mathcal{D}$ (or lying in any other corner of $\mathcal{D}$) is added to the data set.

- *Composition of the two functions:* For a given privacy budget ε_1, we obtain all cardinalities and all sums with the Laplace Mechanism in a differentially private manner. The privacy budget is equally split for both functions f and g. So, the noise is drawn from Laplace distributions with a scaling factor $\beta_f = 2/\varepsilon_1$ for the function f and $\beta_g = 2d/\varepsilon_1$ for g. An ε_1-differentially private mechanism is achieved altogether due to standard arguments for composition (see Theorem 1).

Note that the noise which needs to be added in this Laplace Mechanism does not depend on the number of samples in the data set. Hence, it has a smaller impact if more training samples are present and vice versa.

This differentially private 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 (NG), for example, discussed by Hammer, Strickert, and Villmann (2005). Since an analytic solution of the stationary state of NG does not exist, differentially private variants of NG can be obtained, e. g. by differentially private variants of its gradient scheme.

Gradient descent: For the gradient descent, we rely on the algorithm as introduced in Section 3.2.5 by Abadi et al. (2016a). 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 the clipping parameter C and desired degree of privacy according to Abadi et al. (2016a)).

Let L be the batch size, C a bound for the norm of the gradients, $q = L/n$ 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 that 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 $\sigma = 2q\sqrt{T \log(1/\delta)}/\varepsilon_2$.

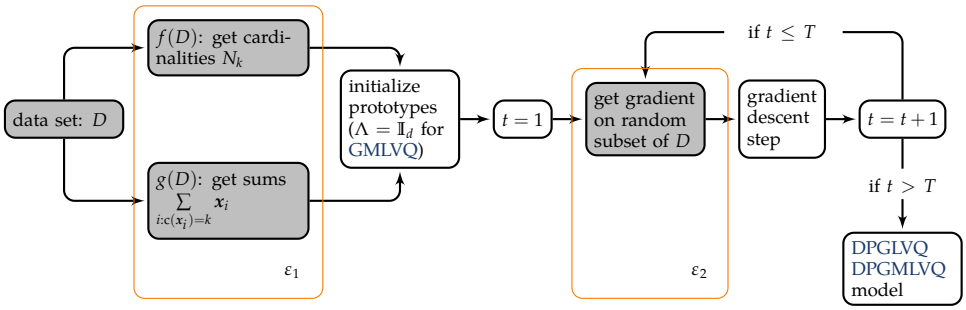


Figure 3.5: Flowchart of the training GLVQ and GMLVQ using differentially private SGD. For GMLVQ, the relevance matrix is initialized as an identity in the same step as the prototypes.

DP bounds for the full scheme: Hence, the total privacy budget of the whole training, i. e. initialization of prototypes followed by gradient-based optimization, is $\varepsilon = \varepsilon_1 + \varepsilon_2$ due to the Notation of Differential Privacy. We obtain an (ε, δ)-differentially private algorithm, note that δ equals 0 for the Laplace Mechanism.

3.4.2 Privacy via Subsample and Aggregate

As an alternative, we investigate the possibility to guarantee DP by a suitable version of SAA as already introduced in Section 3.2.6. 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 rely on the class-wise prototype mean,
- calculate the sum of all prototypes belonging to the same class using the analytic Gaussian Mechanism to guarantee DP and obtain the prototypes by dividing with m.

If more than one prototype per class is used, we would need to add a matching step beforehand minimizing their pairwise distance. The steps are visualized in Figure 3.6. Here, the first four steps are private because the results of the functions are accurate. Only after adding the noise, the result is differentially private.

We need to argue how to choose the variance for the Gaussian Mechanism: averaging is the composition of the sum of prototype's positions and the division by the number of bins. Since the latter is fixed, we only need to make the first operation differentially private by adding Gaussian noise to each dimension. The variance of it can be chosen based on the following estimate of its Global Sensitivity: If the bins are disjoint subsets, adding or deleting one sample to/from the data set affects the output of at most one bin. Hence, the Global Sensitivity of computing the sum can be bounded by $\sqrt{d}$ when measured in the ℓ_2 norm assuming again that $\mathcal{D} \subset [-1, 1]^d$. This is the

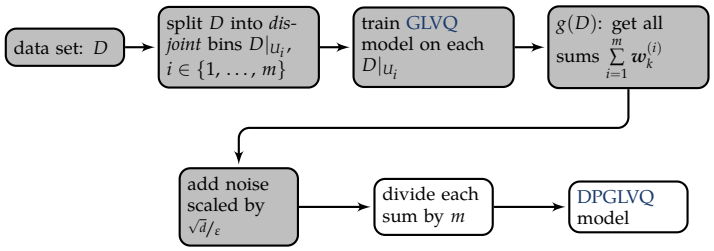


Figure 3.6: Flowchart of the training of differentially private GLVQ using SAA

same argument which we use for the initialization step for the differentially private version of SGD in the last Section 3.4.1.

We use the analytic Gaussian Mechanism (see Theorem 3) as proposed by Balle and Wang (2018) and compute the required variance as in Algorithm 3.1.[3] This method is referred to as *SAA*. Note that the aggregation scheme can also be used to derive differentially private variants of algorithms without explicit cost functions such as LVQ1.

3.4.3 Privacy via Center of Attention

For the method SAA, as introduced in Section 3.2.7, binning and aggregation are done randomly, and the Global Sensitivity is taken. The latter one is quite large because, theoretically, one individual can affect the position of a prototype a lot, as we already have seen in Section 3.3. As an alternative, we also use the Center of Attention (COA) proposed by Nissim, Raskhodnikova, and A. D. Smith (2007) as introduced in Section 3.2.7 which benefits from the local geometric setting and the Local Sensitivity. In contrast to SAA, the type of binning and aggregation is changed as follows:

Binning: We use m subsets from the n training samples with size n/m, which are chosen randomly, each one chosen uniformly without replacement. Since the subsets are chosen separately, there exist samples that are assigned to more than one. Hence, the sets are not pairwise disjoint as in the SAA approach. We consider the probability of one sample being assigned to several subsets. With probability at least $1 - 2^{-\sqrt{m}+\log n}$ no point occurs in more than $\sqrt{m}$ subsets (Nissim, Raskhodnikova, and A. D. Smith 2007). Hence, for suitable m, with a high probability, any sample from the training data affects the result of at most $\sqrt{m}$ subsets. This observation is then used for rating the sensitivity. Otherwise, we re-sample the subsets as suggested by the authors.

Aggregation: The resulting prototypes per subset are matched by their class label. Then, aggregation takes place by the center of attention as described in Section 3.2.7. It has been shown in (Nissim, Raskhodnikova, and A. D. Smith 2007) that adding noise calibrated by the magnitude of the *smooth sensitivity*, which is proportional to the t_0-radius (see Section 3.2.7), of the aggregation function g gives DP. Here, the smooth sensitivity is an upper bound of the

3 Interestingly, their mechanism extends applicability also to the region where $\varepsilon > 1$.

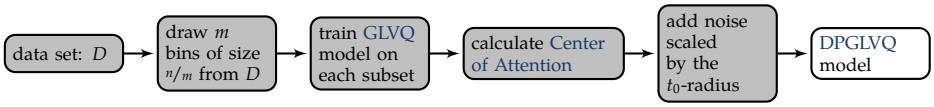


Figure 3.7: Flowchart of the training of differentially private GLVQ using COA

Local Sensitivity that does not share its drawback of possibly revealing private information due to the size of the noise. It can be computed based on the pairwise distances of the outputs from each subset. In particular, if the prototypes for each class are very close to those of the other subsets, a small variance for the Gaussian distribution can be used. The steps are visualized in Figure 3.7 as a flowchart. Similar to SAA, all the steps have to be executed on a private machine because the noise is added in the last one. We will refer to this approach as COA.

3.5 EXPERIMENTS

We provide experiments for the three methods, SGD, SAA, 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 the relevance matrix is very vulnerable to noise. However, we will have a deeper view of fusing metric parameters in Chapter 6. All setups will be evaluated on the same real-world data sets. In addition, we will also evaluate the latter two frameworks on theoretical data sets since this will enable us a deeper view of the unexpected behavior of COA for the benchmarks.

3.5.1 Differential Privacy for gradient based GLVQ

We test our approach with three real-world benchmarks, MNIST (Lecun et al. 1998), Motion Tracking (Anguita et al. 2013), and Image Segmentation (Dua and Graff 2017) that are already presented in Section 2.3. Here, we briefly recap the size of all benchmarks because it will be an issue. The first has 70 000 instances with pictures of handwritten digits. The second one consists of 10 299 samples of recorded acceleration data by a mobile phone. The last one consists of 2310 small image patches of landscapes. Because the first two have high-dimensional vectors (784 and 561 features, respectively), both are pre-processed and projected into 30 dimensions by a PCA. We have already investigated the effect of this pre-processing step in Section 2.3.

Experimental setup: For all settings, evaluation is done based on five times repeated 5-fold cross-validation, i. e. in total all results are averaged over 25 runs. Thereby, in each run, we use four folds as a training set and test the models on the left one. The total privacy budget 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 $E = 50$. In the experiments, we will see that the results are robust within the region around these hyperparameters.

We compare the misclassification rates of our differentially private versions to those of vanilla GLVQ and GMLVQ. Since in this chapter we are aiming for differentially private variants of LVQ rather than arbitrary classifiers, we do not compare to alternative differentially private classification schemes other than the ones introduced. For vanilla GLVQ and GMLVQ, the optimum is found by a standard SGD and, in comparison, by the LBFGS algorithm (Fletcher 1987).

We also test the setup with random initialization of the prototypes by setting $\varepsilon_1 = 0$ and drawing random numbers from the normal distribution for all prototypes. In this setting, the misclassification rates do not drop for any privacy budget up to 5 and any benchmark. However, we provide the plots in Figure B.4.

Evaluated quantities: In Figure 3.8, results for varied privacy budgets are shown for all data sets. Furthermore, the error curves of the hyperparameters (C, E and q) for values of $\varepsilon \in \{0.6, 0.75, 2.5\}$ are plotted in the second, third, and fourth rows in the same figure, respectively. Thereby, only one of the hyperparameters is changed, and the others are fixed. The green and blue lines show the results for GLVQ, and the purple and orange for GMLVQ. Note that our objective is to obtain settings where the priorly chosen privacy budget is as small as possible to ensure higher privacy. Likewise, we still want to gain a classifier with similar misclassification rates as the vanilla LVQ variants.

Lower accuracy for higher privacy: One can see that the privacy budget affects the classification strongly in some regions, and the curves drop steeply at a specific value. For the data sets, the values 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 data sets and smaller lot sizes.

Sensitivity concerning hyperparameters: The choice of the other hyperparameters has a strong influence in critical regions of the privacy budget only, being relatively robust for ε larger than 2. As an example, for Motion Tracking, we test ten different values for $C \in [0.05, 2]$ and $E \in [10, 100]$, and 20 values for $q \in [0.0005, 0.1]$. In a comparison of the second and third row, i. e. $\varepsilon = 0.6$ and $\varepsilon = 0.75$, where the privacy budget is changed only a bit, the impact of these hyperparameters is clearly less. In the last row ($\varepsilon = 2.5$), one cannot see large variances. Further results, likewise for the other two benchmarks where similar curves can be observed in the critical regions, are given in Appendix B.2.

Rationale for hyperparameter choice: Recall that the noise is drawn from a Gaussian 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 caused by a more potent influence of the gradients with smaller amplitudes. Here, a choice up to $C = 0.5$ seems the largest one that does not worsen the performance. Regarding the number of epochs, too few iterations

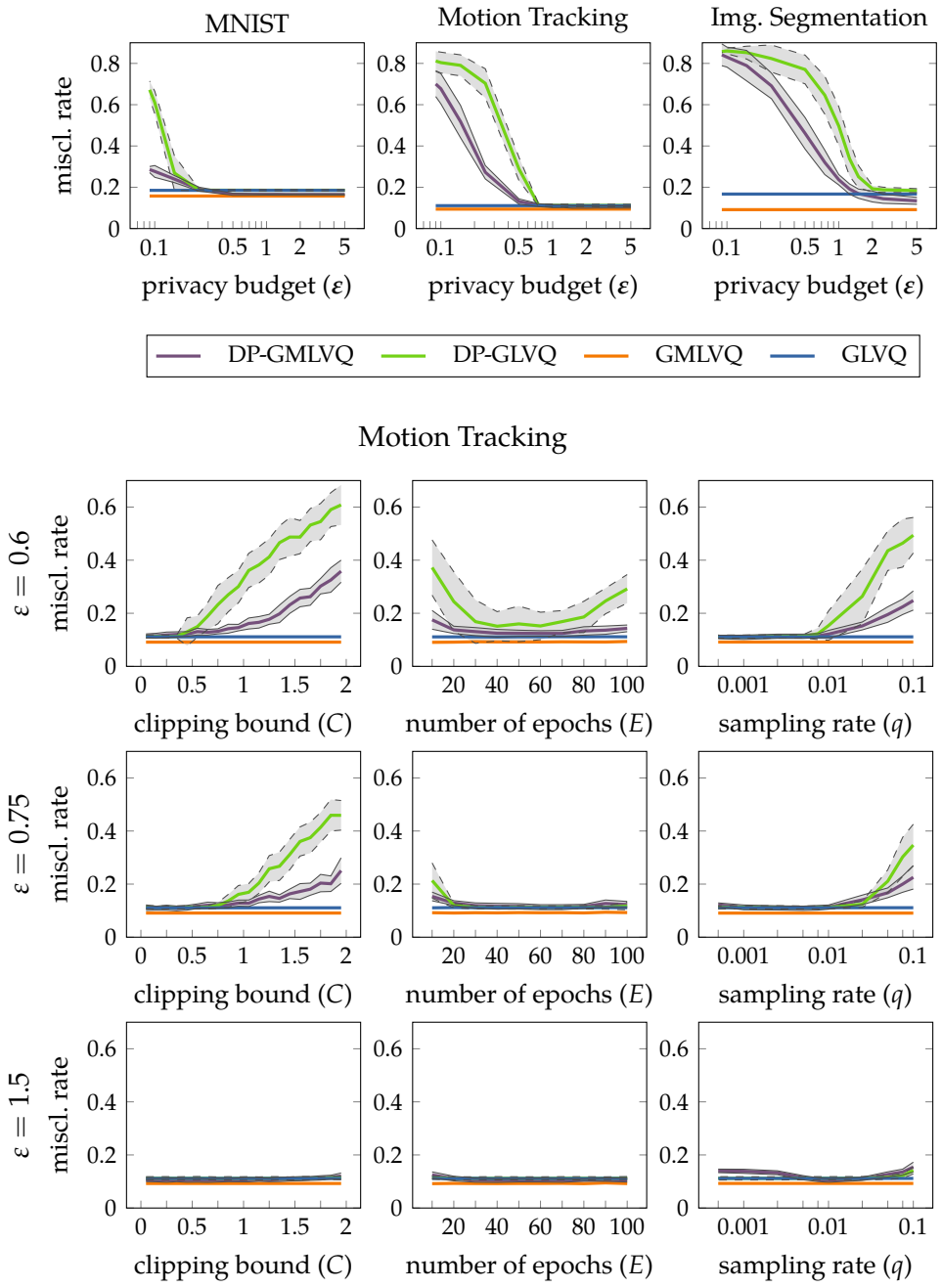


Figure 3.8: Averaged GLVQ and GMLVQ misclassification rates for SGD and in vanilla form on three benchmarks. Only the privacy budget varies and all other hyperparameters are fixed ($C = 0.5$, $E = 50$, and $q = 0.01$).
For the Motion Tracking data set the averaged misclassification rates are also plotted with varying hyperparameters (C, E, and q) for different privacy budgets; $\varepsilon = 0.6$ in the second, $\varepsilon = 0.75$ in the third, and $\varepsilon = 1.5$ in the fourth row.

can lead to premature stopping before convergence, while too many steps lead to an accumulation of the noise in each descent 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 since more than one sample is selected for random subsets. Hence, all hyperparameters are set accordingly to the sweet spots observed in these experiments.

Size of privacy budget: Naturally, the privacy budget ε should be as small as possible. Yet, it cannot reach zero less since this would render learning impossible. Which privacy budget is acceptable in practice? This depends very much on the application at hand, and it can be observed that bigger data sets enable better privacy in general. The privacy budget that we are able to obtain compares favorably to other approaches in the literature such as presented, e. g. in the work by Nissim, Raskhodnikova, and A. D. Smith (2007), since we need less iterations for convergence and, therefore, a much smaller ε. In this work, a neural network consisting of 1000 hidden units is presented with privacy budget up to 8 for the MNIST data set and 750 epochs. Due to the choice of the model as a comparably complex one, however, the accuracy as presented in this contribution is higher (up to 98 %).

Overall performance of gradient-based DP schemes for LVQ: In Table 3.2, the means and standard deviations of the misclassification rates for all three benchmarks are listed. For GLVQ, we often get trained models which perform on the test sets almost as well as the non-private ones. For GMLVQ, LBFGS 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 metric 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 misclassification rate for the original GMLVQ approach and the MNIST data set. For the Motion Tracking benchmark, the misclassification rate increases by 0.0234 ± 0.0085 using the same settings. Hence, alternative schemes tackling specifically stable differentially private variants of matrix adaptation are beneficial.

3.5.2 Subsample and Aggregate

For the SAA and COA mechanism, we evaluate the behavior of GLVQ only. As discussed in the last section, the aggregation of the relevance matrices is very sensitive to noise. However, we will have a detailed look at fusing different metric parameters without disturbing noise later in Chapter 6. First, we test both aggregation functions on artificial data sets with well-separable data that we will introduce to get a deeper insight into their behavior. Second, we evaluate the methods on the same three benchmarks as for the SGD approach.

Data set	$\varepsilon = 0.25$	$\varepsilon = 0.75$	$\varepsilon = 1.5$	$\varepsilon = 2.5$	non DP SGD	non DP BFGS
MNIST	0.186 (0.0038) 0.191 (0.0096)	0.186 (0.0038) 0.165 (0.0038)	0.186 (0.0036) 0.165 (0.0033)	0.186 (0.0037) 0.165 (0.0036)	0.185 (0.0033) 0.158 (0.0030)	0.184 (0.0037) 0.148 (0.0029)
Mot. Track.	0.703 (0.0705) 0.273 (0.0324)	0.113 (0.0058) 0.113 (0.0072)	0.112 (0.0059) 0.104 (0.0058)	0.112 (0.0058) 0.104 (0.0054)	0.111 (0.0065) 0.094 (0.0080)	0.111 (0.0063) 0.090 (0.0087)
Img. Seg.	0.823 (0.0658) 0.691 (0.0621)	0.646 (0.0918) 0.321 (0.0592)	0.251 (0.0541) 0.169 (0.0224)	0.188 (0.0126) 0.144 (0.0224)	0.167 (0.0103) 0.092 (0.0136)	0.167 (0.0105) 0.089 (0.0131)

Table 3.2: Mean and standard deviation (in parenthesis) of the misclassification rates for SGD and all benchmarks. As a baseline, the results of a non-private training with SGD and a Broyden-Fletcher-Goldfarb-Shanno (BFGS) optimizer are given. The first rows for each data set are results for GLVQ, and the second for GMLVQ.

Artificial data: The artificial data sets are generated by three multidimensional Gaussian distributions with means in $(5, 0)$, $(-5, 0)$, and $(0, 5)$ and identities as covariance matrices. For higher dimensions, we choose accordingly scaled unit vectors as centers for the Gaussian distributions, i. e. unit vectors in the first three dimensions scaled by 5. All clusters have 3000 samples, so the data set consists of 9000 samples. It is self-evident that classes are well and easily classifiable.

In Figure 3.9, the GLVQ misclassification rates on the test set are plotted for different values of the privacy budgets and a various number of bins in the first two rows. Again, the curves fall sharply at a particular value of the privacy budget. For SAA, a privacy budget greater than 1.25 is sufficient for all three data sets if m is set to 50. For COA, a smaller value of m gives better results though it becomes worse if the dimension rises due to the greater pairwise distances of the prototypes for the bins. As a consequence, the required level of noise increases which causes unsuitable methods for higher dimensionalities.

Increasing the samples up to 10 000 per class, i. e. this second data set consists of 30 000 samples in total, results in a comparable trend. We report the graphs for larger artificial data sets in two, four, and eight dimensions in Figure B.5. For this set, a privacy budget greater than 1 is in the two-dimensional case sufficient.

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 SAA: If the number of bins is larger, the output of the models trained on the bins varies more. SAA just calculates the mean of all prototypes which is a good approximation due to the robust geometric properties of LVQ. Since the Global Sensitivity, i. e. the variance of the noise, is proportional to $1/m$, the added noise is smaller for larger m. For COA, the noise is scaled by twice the minimal t-radius (see Section 3.4.3). Here, the noise added to the center of attention becomes vaster if the dimensionality increases. We observed that, de facto, the noise shifts prototypes outside the cluster, hence the results get worse.

Real-word data: In Figure 3.10, the misclassification rates for the three real-world data sets are depicted. The number of bins is set between 15 and 50 for COA and SAA, respectively. For the tested privacy budgets, COA does not provide any valuable solutions, which can be attributed to the problems

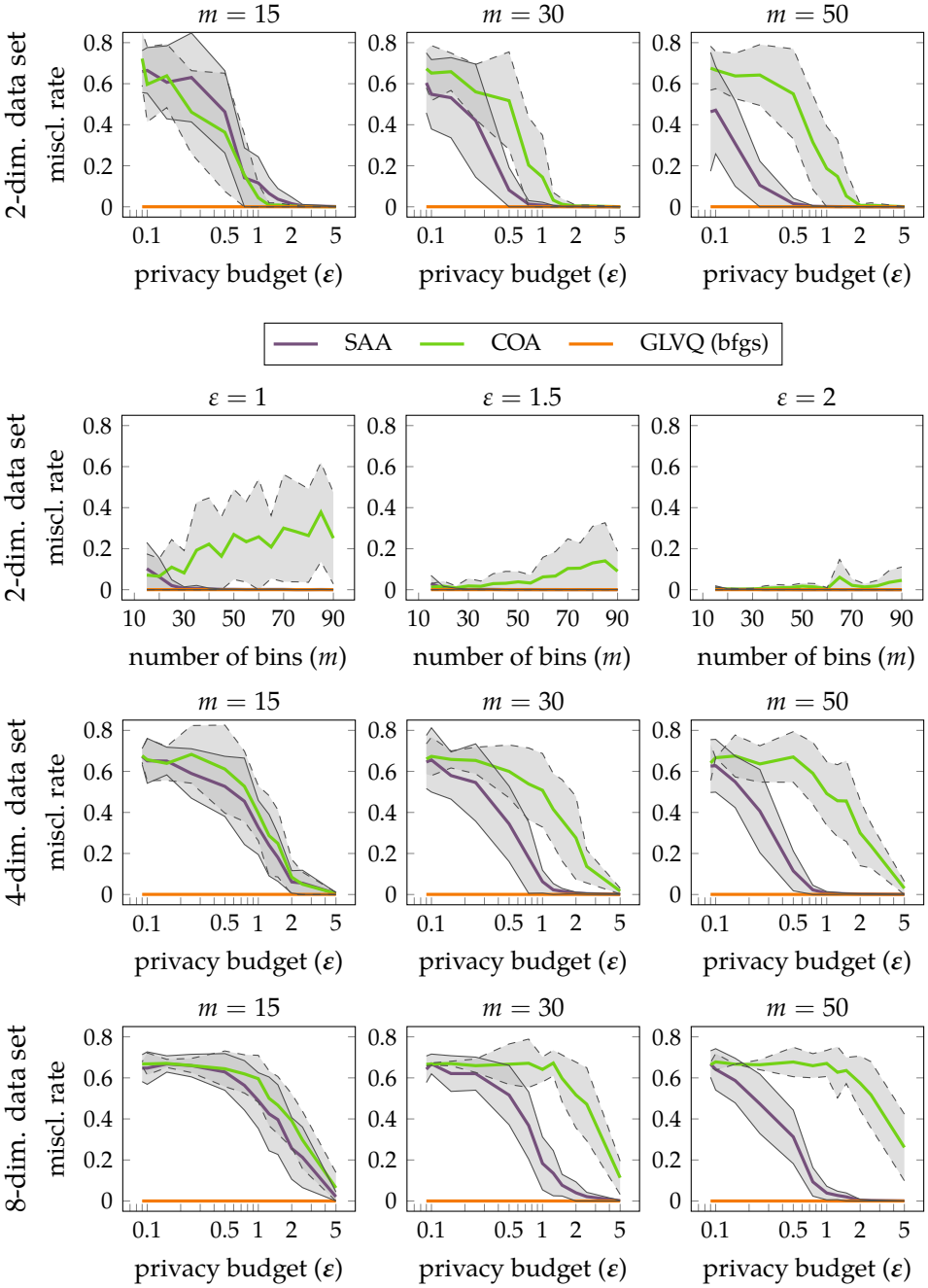


Figure 3.9: Averaged GLVQ misclassification rates for the artificial data sets with different dimensions (2 in the first, 4 in the third and 8 in the forth row) and different numbers of bins ($m \in \{15, 30, 50\}$). In the second row, the misclassification rate is plotted against m for three privacy budgets ($\varepsilon \in \{1, 1.5, 2\}$).

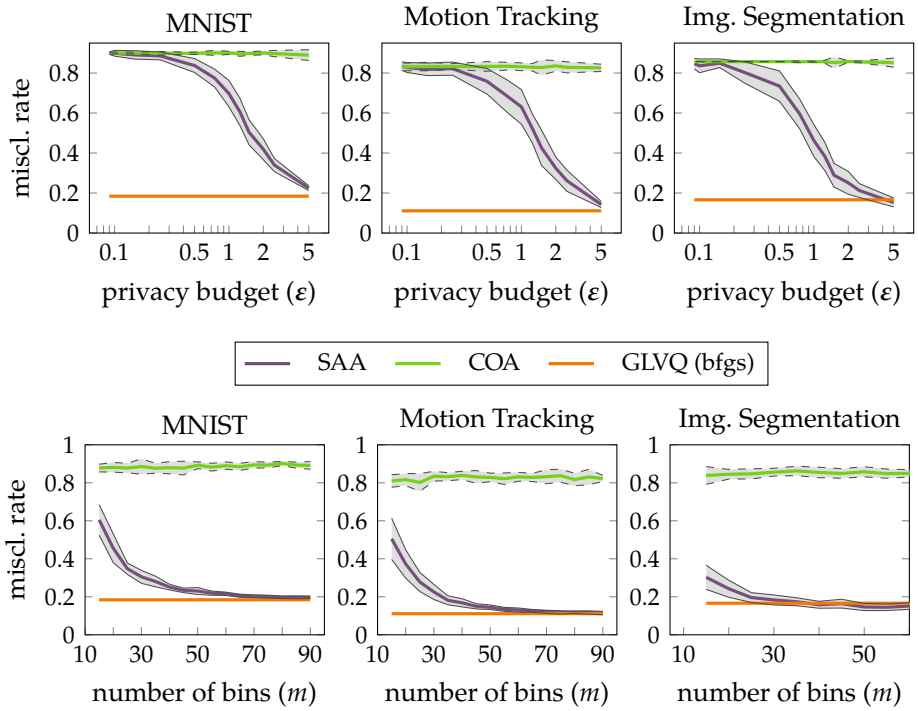


Figure 3.10: Averaged GLVQ misclassification rates for different privacy budgets for our approach and the non-private version with BFGS optimization on the three benchmarks. In the second row, the impact for the parameter m is shown for a privacy budget of 5.

of higher dimensionalities, as already discussed for the theoretical data sets. SAA provides solutions with privacy budget larger than 2, whereby the required noise is proportional to the data dimensionality. Albeit the result of COA are reasonable, SGD yields better performance and seems better suited to make LVQ private in realistic settings.

3.6 METHOD COMPARISON

In the previous sections, we have adapted, implemented, and tested three different methods for ensuring DP of GLVQ. Now, we provide a structured comparison, which is somewhat higher level, to shed some light on 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
4. formal guarantees.

Effectiveness

Our experiments show that, for realistic data sets, SGD gives generally good classification accuracy for privacy budget ranging from 0.75 to 2.5 at least.

SAA yields acceptable results for privacy budget of 2 or 5, respectively, whereas COA does not yield acceptable results for realistic data and the tested privacy budgets, which can be attributed to the high dimensionality of the data. Generally, SAA is more suited for large data sets with stable results on the single subsets, especially well-separated data sets. Since the variance of noise added to the mean in the SAA framework is inversely proportional to the number of subsets, the noisiness of the differentially private estimate can be significantly reduced when a data set allows a large number of subsets. On top of this, the COA framework adaptively reduces the noise added when the underlying data set exhibits a lower smooth sensitivity, meaning that less noise is added to large and good-natured data sets – a setting that 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 3.3, because of a mismatch between data and model parameters, SAA and COA can both become very noisy. In addition, SAA is not effortlessly applicable to other LVQ variants: GMLVQ, for example, would require an averaging of the relevance matrix, which requires careful thought lest it provides nonsensical results. SGD does not adapt the noise added to the size of the data sets or the stability of GLVQ on subsets of the data. However, through rigorous accounting of accumulated budget, good results are achieved even for small privacy budgets, and methods can easily be adapted to any other LVQ methodologies based on gradient descent.

Ease of use

Implementing SAA 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 m that may contain the same point. While Nissim, Raskhodnikova, and A. D. Smith (2007) gives recommendations for most of these parameters, though they do 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 corresponding 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 SAA and COA. Then, calculating sample and aggregate requires m GLVQ runs on n/m samples each, with complexity $n/m \cdot c \cdot$ number of iterations per LVQ. These runs are completely independent and, thus, can be easily parallelized. SAA requires computing an average over m GLVQ models. If there is more than one prototype per class, matching between the prototypes can easily be implemented, e. g. as a maximum flow problem per class. COA also requires m GLVQ runs on n/m samples each, and may

also require 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 standard minibatch gradient descent. COA and SAA 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 SAA for the MNIST data on a Intel® Xeon E5-1620 v3 with 3.5GHz and 16 GB RAM takes about 7s.

Formal guarantees

Per design, all methods guarantee DP with a certain privacy budget. Yet, to the best of our knowledge, there are no results that allow us to limit the noise which needs to be added by SGD or SAA. Nissim, Raskhodnikova, and A. D. Smith (2007, 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 subsets 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 unclear whether conditions on a data set exist 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 COA may differ markedly from the optimal solution, as we have observed in experiments.

3.7 SUMMARY

We have investigated technologies that enable an extension of LVQ schemes to variants respecting 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 representing the model are essentially close to the centers of samples. Hence, similar to the privacy risks of the mean, outliers might be detected. Second, the result of LVQ is not stable, particularly in settings where the number of prototypes does not match the inherent modality of the data distribution and several symmetric optimal solutions exist. Furthermore, in these settings, information about a few points may possibly be leaked since a few points can break those symmetries. Turning LVQ schemes into differentially private variants faces the challenge that, according to the possibly large sensitivity of LVQ models, different mechanisms are needed that focus on batches and

their suitable aggregation. The latter can be either incorporated into the 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 that can be uncovered as concerns a single sample from the final result (**RQ1**).

Depending on the accepted privacy budget, these technologies yield acceptable results, and the sensitivity as regards hyperparameters can be controlled. Interestingly, a simple aggregation seems better suited, particularly for high-dimensional data sets compared to a more advanced aggregation based on the Center of Attention. Further, gradient-based techniques provide convincing results.

These mechanisms are the first steps towards learning schemes for LVQ that enable the release of models also in areas with highly sensitive data such as activity profiles, in medicine, or needed for biomedical models. 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, differentially private 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 the accuracy of the model since random noise rather than dedicated information is added. Typically, it goes beyond the amount of noise used to regularize machine learning models. Hence, comparably big data is required for appropriate 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 that 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 by Ligett et al. (2017). It would be an interesting attempt to extend this technology to LVQ schemes.

4 ROBUSTNESS AND CONFIDENCE OF LEARNING VECTOR QUANTIZATION

Overview: In this chapter, we investigate reject options via post-processing to answer **RQ2**: *How can the predictions of LVQ models be more robust*? In Section 4.4, we show how to turn the output of an LVQ classifier into class probabilities which directly provides a formalization for time integration. It becomes handy in scenarios where a sequence of samples is classified and one class is present for a while. In these settings, time integration leads to a much more robust classification. Further, we analytically derive strict bounds that limit the space for non-rejected samples in Section 4.3. These bounds limit the distance between a point and its representing prototype.

Publications: Parts of this chapter are based on the following publications:

- Brinkrolf, Johannes and Barbara Hammer (2017). "Probabilistic extension and reject options for pairwise LVQ". In: *12th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization, WSOM 2017, Nancy, France, June 28-30, 2017*, pp. 205–212. DOI: 10.1109/WSOM.2017.8020028. URL: https://doi.org/10.1109/WSOM.2017.8020028.
- Brinkrolf, Johannes and Barbara Hammer (2018). "Interpretable machine learning with reject option". In: *at - Automatisierungstechnik* 66.4, pp. 283–290. DOI: 10.1515/auto-2017-0123. URL: https://doi.org/10.1515/auto-2017-0123.
- Brinkrolf, Johannes and Barbara Hammer (2020b). "Time integration and reject options for probabilistic output of pairwise LVQ". In: *Neural Computing and Applications* 32.24, pp. 18009–18022. DOI: 10.1007/s00521-018-03966-0. URL: https://doi.org/10.1007/s00521-018-03966-0.

4.1 INTRODUCTION

We have already discussed the privacy of the LVQ models and how we can publish trained models without leaking any information about the training samples. In this chapter, we will focus on the *certainty* of the model's prediction and the possible extension of *reject options*. The latter one will enhance the *robustness* of the classifier and address **RQ2**.

The problem of the reliability of a classification prescription has recently gained some popularity in particular, in the light of safety conditions of complex cyber-physical or autonomous systems (Varshney and Alemzadeh 2016). For example, popular deep networks are prone to attacks even if only one pixel of a high-dimensional input is changed (Su, Vargas, and Sakurai 2019) and even in realistic physical settings (Athalye et al. 2017). The cause of this effect is a matter of debate; one explanation is the model's linearity in combination with a high input dimensionality. The linearity is a characteristic of

many models, including LVQ if used for high-dimensional signals (Goodfellow, Shlens, and Szegedy 2014). In the latter contribution, it is debated that it is necessary to incorporate an explicit reject option of a classifier to avoid such problems.

More generally, many training settings, including learning from few data and learning in open environments with changing conditions, have the consequence that a given classifier is valid for a comparably narrow region of the data space. Since data will likely occur that is not covered by the training regions when the model is applied, we need the crucial capability of the model to reject the classification of such novel samples to guarantee that no fatal errors are made in such settings. Hence, besides the most likely class label, a classifier needs a judgment of its certainty to reject a classification in uncertain cases.

Classification with a reject option has been pioneered by Chow (1970), respectively Herbei and Wegkamp (2006), who provide strong theoretical guarantees on when and how to reject. These approaches build on probabilistic classifiers, which provide explicit class probabilities or an empirical (in the limit consistent) estimation thereof. These techniques can be used if a probabilistic or generative model is learned, but they are not directly applicable to discriminant classifiers. For many practical applications, an estimation of the exact probability is unfeasible, e. g. due to a high data dimensionality, few training data, or restricted computational resources. Approaches such as those proposed by Yuan and Wegkamp (2010), offer a relaxation of the requirement of an explicit probabilistic model aiming for convex costs as a surrogate function, which can efficiently be optimized.

In this chapter, we inspect reject options for deterministic LVQ methods. A few approaches have considered reject options for prototype-based methods, such as the work by Gamelas Sousa et al. (2015) for unsupervised Self-Organizing Maps (SOM), and the work by Fischer, Hammer, and Wersing (2015) for LVQ variants. Further, the approach by Villmann, Kaden, et al. (2016) extends GLVQ to an adjusted reject option provided the loss of a reject can be explicitly quantified. Further, note that variants of LVQ have been proposed which rephrase the classifier as an explicit probabilistic model using a class-wise mixture of Gaussians, the Robust Soft Learning Vector Quantization (RSLVQ) (Seo and Obermayer 2003). However, as demonstrated by Fischer, Hammer, and Wersing (2016), these versions do not offer an unbiased estimation of the probabilities such that its plug-in into reject options yields inferior results compared to deterministic counterparts. Therefore, we do not follow up on these approaches.

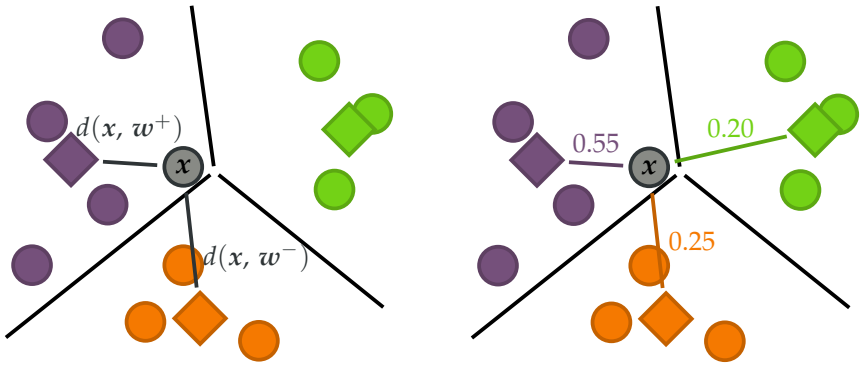
Now, we focus on LVQ as a supervised discriminative classifier and we aim to answer the question of whether an efficient probabilistic post-processing is possible and how to realize it best. The post-processing extends deterministic LVQ schemes to provide properly scaled probability measures, for which optimum reject options can be based on the work by Chow (1970). This twofold procedure offers the advantage that the probabilities can intuitively explain why an example is rejected. Besides, it is a foundation for additional post-processing techniques, e. g. other reject frameworks or increasing model robustness by aggregating outputs over time, as we will discuss in more detail later.

We compare our approach of explicit probabilistic post-processing to the

deterministic surrogate of LVQ proposed by Fischer, Hammer, and Wersing (2015). Here, a real-valued quantity that is related to the classifiers' hypothesis margin is directly computed from the model. Albeit this value does not provide a valid probability of the classification, its relative ordering induces a good reject strategy. It has been shown that its local scaling might be misleading and that locally adjusted thresholds can yield superior results, particularly for simple LVQ models based on the standard Euclidean metric (Fischer, Hammer, and Wersing 2016).

Further, we introduce an extension of these static classification schemes and corresponding certainty measures for reject options in complex settings. Such settings may be subsequent measurements taken over time where the task is to reduce the uncertainty in noisy settings. In practice, such approaches are relevant if characteristic signals cannot be robustly observed at all times of measurement. Examples are observations of objects on the street, which are partially hidden, e. g. by cars, or the classification of longer-lasting periodic activities via smartphone signals. There is vast research on how to classify time series (Bagnall et al. 2017). However, we are not interested in the temporal nature of signals, rather we are interested in ways how to increase the robustness of an instantaneous classification by integrating subsequent, possibly noisy classification results over time. This goal is related to approaches that aim for the classification of temporal signals as early as possible (Mori et al. 2017). Yet, we are interested in the integration of essentially static classifications over time by relying on a suitable integration scheme for the proposed reject option.

In the following, we consider GLVQ and GMLVQ again. We briefly recall the reject strategy as used by Fischer, Hammer, and Wersing (2015) for deterministic LVQ. In Section 4.3, we will derive strict boundaries for this reject strategy that limits the data space of non-rejected samples. Further, the minimum required perturbation needed to change the classification of one sample, is deduced from these boundaries. This is one measurement for the classifier's robustness because similar samples are not classified into different classes if the amount of the necessary perturbation is higher. Further, we show in Section 4.4 how to extend a probabilistic post-processing scheme, which is popular in the context of the SVM (Platt 1999; Wu, C.-J. Lin, and Weng 2004), to pairwise LVQ schemes, enabling an efficient probabilistic treatment of LVQ and hence explicit reject options. Thereby, an initial scaling relies on pairwise classifiers (Platt 1999), and different possibilities for how to extend those to multiple classes exist (Friedman 1996; Hastie and Tibshirani 1997; Price et al. 1994). We compare those in the context of pairwise LVQ, resulting in an overall classification confidence in all settings. One benefit of the latter approach, as opposed to the direct deterministic surrogate proposed by Fischer, Hammer, and Wersing (2015), consists of an interpretable scaling. It yields an estimate for the classification likelihood, while the value offered in the latter approach has no semantic meaning besides its relative ordering. This is also motivated in Example 6 where a new sample is classified with both methods. At the end of this chapter, we consider a time integration scheme for the resulting certainty measures in the case of observations in time series and the effectiveness of the resulting time integrated classification schemes with reject options in Section 4.5.



(a) Classification via reject function. The certainty of the unknown sample x is 0.295.

(b) Classification via class probabilities.

Figure 4.1: Comparison of rejects via a certainty measure in Figure 4.1(a) and via class probabilities in Figure 4.1(b). In the first scenario, the sample is rejected by a small certainty which only takes the distances to its two closest prototypes (purple and orange) into account. The probabilistic approach gives the likelihood for each class/prototype. See Example 6 for a full description.

Example 6. *In Figure 4.1, two reject options, one via a certainty measure (Figure 4.1(a)) and one via class probabilities (Figure 4.1(b)), are visualized. Until now, we consider LVQ models that classify a new sample by the smallest distance to one prototype, i. e. by the winner-takes-all scheme (see Equation (2.8)). In the plot, this new sample is classified as class purple. In the next section, we will introduce a suitable certainty measure* $\mathrm{r}\colon \mathbb{R} \to [0,1]$ *that is strongly related to the cost function of GLVQ and is deduced from the distances between the sample and the two closest prototypes; here, these are the distances to the purple* $d(\boldsymbol{x}, \boldsymbol{w}^+)$ *and the orange* $d(\boldsymbol{x}, \boldsymbol{w}^-)$ *prototype. Let the certainty for this sample be* 0.295*. Then, this sample would be rejected if we only allow samples with certainty greater than its, e. g.* 0.3*.*

From the certainty and both distances, one can only infer that the sample is close to a decision boundary because the certainty reaches 0 *for points there. Since the distance between the sample and the green prototype is slightly larger than to the orange one, the third prototype is not considered. In contrast, if class probabilities are calculated, likelihoods for each class and, in this example, each prototype are obtained. The sample is still classified as purple, but one can also conclude that it lies close to both decision boundaries since both classes are similarly likely (*20 % *and* 25 %*). The possibility of rejecting such samples is still there by setting a threshold to a probability higher than* 0.55*. In such cases, class probabilities are easier to interpret and can provide more insight into the prediction of a sample.*

4.2 REJECT OPTION FOR CLASSIFICATION

All LVQ models, as defined in Section 2.2, provide classifications of the entire input space, regardless of whether there is evidence that the prediction of a given input is reliable based on the trained model. Classification with a reject option extends a classifier by an explicit possibility to reject a clas-

sification in case the input comes from an uncertain region, i. e. the output range is extended by a new class which indicates that classification has been rejected due to a large uncertainty of the predicted class label. This option is useful whenever the costs of a wrong classification are higher than the costs to reject a classification, e. g. in safety-critical areas, such as assisted driving or decision-making in medical domains, or whenever another option is available in case of reject, such as a default behavior or failsafe option. The notion of classification with rejects has been pioneered by Chow (1970). He demonstrated that in the case of a two-class probabilistic classifier, an optimum reject is given by a simple threshold scheme: the classification is rejected whenever the risk of misclassification exceeds the costs for a false classification. In practice, true probabilities are usually not known. Hence, surrogates have to be used. Herbei and Wegkamp (2006) analyzed guarantees whenever true probabilities are substituted by plugin estimates. Due to their (under suitable conditions) positive results, reject options are often based on probabilistic estimates of the confidence of the classification. As an alternative, purely deterministic functions can be used as surrogates. A few variants have been proposed for LVQ, which rely on alternative classification certainty measures which are easier to compute from deterministic classifiers (Fischer, Hammer, and Wersing 2015). They have the drawback that their scaling is unclear since it does not represent statistical confidence, and the scaling is not necessarily uniform over the input space. This latter property has the consequence that, unlike Chow's global scheme, local reject thresholds can improve the overall behavior, but at the price of an additional effort (Fischer, Hammer, and Wersing 2016; Pillai, Fumera, and Roli 2013).

4.2.1 Reject Option

We extend the set of all classes by the symbol $\oslash$ indicating a *reject*, thus

$$\boldsymbol{x} \mapsto \mathrm{c}(\boldsymbol{x}) \in \{1, \ldots, c, \oslash\}. \tag{4.1}$$

Further, we focus on the simple reject option based on the certainty measure of the classifier as has been pioneered by Chow (1970), and investigate which certainty measures are suitable for this purpose. We assume that a certainty measure is given, i. e. a function on $\mathcal{D}$ that estimates the *certainty* of a sample

$$\mathrm{r}\colon \mathcal{D} \to \mathbb{R},\ \boldsymbol{x} \mapsto \mathrm{r}(\boldsymbol{x}). \tag{4.2}$$

The reject option for a classifier h is then given by

$$\mathrm{c}(\boldsymbol{x}) = \begin{cases} \mathrm{h}(\boldsymbol{x}) & \text{if } \mathrm{r}(\boldsymbol{x}) \geq \theta \\ \oslash & \text{if } \mathrm{r}(\boldsymbol{x}) < \theta \end{cases} \tag{4.3}$$

where $\theta \in \mathbb{R}$ characterizes the reject.

4.2.2 Relative Similarity as Certainty Measure

Several certainty measures have already been proposed for GLVQ models, their extensions to metric learning, and probabilistic counterparts. For example, Fischer, Hammer, and Wersing (2015) investigate several certainty

measures and propose the *Relative Similarity (RelSim)* as an efficient and particularly robust choice:

Definition 9 (Relative Similarity)**.** *Given a GLVQ model with* $d(\bullet, \bullet) \colon \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ *as its dissimilarity function. The* Relative Similarity *for a point* $\boldsymbol{x}$ *is defined as*

$$\mathrm{r}_{\mathrm{RelSim}}(\boldsymbol{x}) = \frac{d(\boldsymbol{x}, \boldsymbol{w}^-) - d(\boldsymbol{x}, \boldsymbol{w}^+)}{d(\boldsymbol{x}, \boldsymbol{w}^+) + d(\boldsymbol{x}, \boldsymbol{w}^-)} \tag{4.4}$$

where $\boldsymbol{w}^+$ *is the closest prototype from the model and* $\boldsymbol{w}^-$ *is the closest one belonging to a different class than* $\boldsymbol{w}^+$*.*

Note that the value $\mathrm{r}_{\mathrm{RelSim}}(\boldsymbol{x})$ is the negative of $\mu(\boldsymbol{x})$, i. e. the argument of the monotonic increasing function Φ in the cost function (see Equation (2.11)). It lies between 0 and 1, where 0 is approached whenever the closest two prototypes have comparably equal distances and different class labels. This is the case if the sample is either close to a decision boundary or the distance to all prototypes is large, i. e. it is an outlier that is likely not covered by the training data and thus not represented by one prototype. The denominator, i. e. the sum of both dissimilarities, pushes the value even more to zero. 1 is approached if a sample coincides with a prototype, i. e. $d(\boldsymbol{x}, \boldsymbol{w}^+) = 0$ (Fischer, Hammer, and Wersing 2015). As we will see in Section 4.3, this fact can be proved analytically by providing explicit bounds for a minimum and maximum required perturbation of a sample to change its class label.

4.2.3 Probabilistic Surrogates

A probabilistic classifier, such as the Bayes classifier, provides estimations of class probabilities for each class. In this case, a reject can be based on the certainty measure offered by the likelihood

$$\mathrm{r}_{\mathrm{Bayes}}(\boldsymbol{x}) = \max_{1 \leq k \leq c} p(k \mid \boldsymbol{x}) \tag{4.5}$$

yielding the Bayes optimal decision in case of homogeneous costs for errors (Chow 1970). In Section 4.4, we will propose four methods to extend a deterministic LVQ model to provide probabilistic outputs by a suitable post-processing. These will induce certainty measures

$$\mathrm{r}_{\star}(\boldsymbol{x}) = \max_{1 \leq k \leq c} p_k^{\star}(\boldsymbol{x}) \tag{4.6}$$

where the symbol $\star$ constitutes a placeholder for the techniques explained in Section 4.4.2, and an according reject option. In all cases, the certainty measure is lower bounded by $1/c$ and upper bounded by 1. Unlike deterministic measures such as $\mathrm{r}_{\mathrm{RelSim}}$, its scaling has a semantic meaning since it is designed as a surrogate for statistical confidence.

4.3 ROBUSTNESS OF LVQ

The robustness of prototype-based classifiers has been analyzed for more than 20 years. For example, Crammer et al. (2002) have derived a gener-

alization bound for LVQ using the Euclidean norm by relying on the hypothesis margin maximization properties of such classifiers. Later, Saralajew, Holdijk, and Villmann (2020) provided margin analyses for distances induced by semi-norms. Recently, the lower bounds have been extended to general semi-metrics (Vorácek and Hein 2022).

However, in this section, we will not only consider the margin of LVQ models but have a closer look at the RelSim. We will describe two properties of the induced reject options and show that those hold. First, samples with a high dissimilarity to the model's prototypes, i. e. outliers, and second, uncertain samples are rejected. Hence, we can increase the robustness since small changes in the input cause rejects instead of misclassifications. More precisely, the following holds:

Theorem 5 (Robustness of LVQ)**.** *Assume a binary GMLVQ model with two prototypes $\boldsymbol{w}^+$ and $\boldsymbol{w}^-$ and with reject option induced by RelSim, and a threshold $\theta \in (0, 1)$ is given. Then, we can find the points $\boldsymbol{x}$ that are either the closest or the furthest point from the prototype $\boldsymbol{w}^-$, respectively, are not rejected by RelSim, and are classified as $\mathrm{c}(\boldsymbol{w}^+)$, i. e. the following optimization problems*

$$\begin{aligned} &\min_{\mathrm{c}(\boldsymbol{x})=\mathrm{c}(\boldsymbol{w}^+)} \pm d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^-) \\ \text{such that} \quad &\mathrm{r}_{\mathrm{RelSim}}(\boldsymbol{x}) \geq \theta \end{aligned} \tag{4.7}$$

admit the solution

$$\boldsymbol{x} = \boldsymbol{w}^+ + \lambda_\pm(\boldsymbol{w}^- - \boldsymbol{w}^+) \text{ where } \lambda_\pm = \frac{\theta - 1 \pm \sqrt{1-\theta^2}}{2\theta}. \tag{4.8}$$

Proof. Recall that $d_\Lambda(\boldsymbol{x}, \boldsymbol{0}) = \boldsymbol{x}^\top \Lambda \boldsymbol{x}$ for all $\boldsymbol{x} \in \mathbb{R}^d$ where Λ is the model's relevance matrix. It holds $d_\Lambda(\boldsymbol{p}, \boldsymbol{q}) = d_\Lambda(\boldsymbol{p} + \boldsymbol{z}, \boldsymbol{q} + \boldsymbol{z})$ for all $\boldsymbol{z} \in \mathbb{R}^d$. Thus, without loss of generality, we may assume $\boldsymbol{w}^+ = \boldsymbol{0}$, i. e. translate the data by $-\boldsymbol{w}^+$. Furthermore, by choice of the basis, we assume $\boldsymbol{w}^-$ is the first basis vector. In particular, $\mathbb{R}^d$ decomposes into an one-dimensional subspace spanned by $\boldsymbol{w}^-$ and its orthogonal complement, so that we may write every vector $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{x}^\perp = \lambda \boldsymbol{w}^- + \boldsymbol{x}^\perp$ with $(\boldsymbol{w}^-)^\top \Lambda \boldsymbol{x}^\perp = 0$ and $\lambda \in \mathbb{R}$. By the binomial formula, it holds:

$$d_\Lambda(\boldsymbol{x}', \boldsymbol{w}^\pm) = d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^\pm) + d_\Lambda(\boldsymbol{x}^\perp, \boldsymbol{0}). \tag{4.9}$$

Further, the RelSim for $\boldsymbol{x}'$

$$\begin{aligned} \mathrm{r}_{\mathrm{RelSim}}(\boldsymbol{x}') &= \frac{d_\Lambda(\boldsymbol{x}', \boldsymbol{w}^-) - d_\Lambda(\boldsymbol{x}', \boldsymbol{w}^+)}{d_\Lambda(\boldsymbol{x}', \boldsymbol{w}^+) + d_\Lambda(\boldsymbol{x}', \boldsymbol{w}^-)} \\ &= \frac{d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^-) + d_\Lambda(\boldsymbol{x}^\perp, \boldsymbol{0}) - d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^+) - d_\Lambda(\boldsymbol{x}^\perp, \boldsymbol{0})}{d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^+) + d_\Lambda(\boldsymbol{x}^\perp, \boldsymbol{0}) + d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^-) + d_\Lambda(\boldsymbol{x}^\perp, \boldsymbol{0})} \\ &= \frac{d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^-) - d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^+)}{d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^+) + d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^-) + 2d_\Lambda(\boldsymbol{x}^\perp, \boldsymbol{0})} \\ &\leq \frac{d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^-) - d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^+)}{d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^+) + d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^-)} \\ &= \mathrm{r}_{\mathrm{RelSim}}(\boldsymbol{x}). \end{aligned}$$

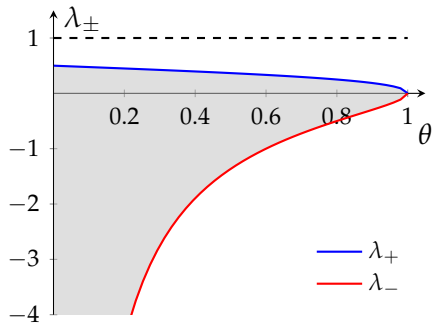


Figure 4.2: The values for $\lambda_\pm(\theta)$ for all possible $\theta \in (0, 1]$ from Theorem 5. Putting $\boldsymbol{w}^+$ on the x-axis in $(\theta, 0)$ and $\boldsymbol{w}^-$ on the dashed line above in $(\theta, 1)$, the graphs gives the closest and furthest point to/from $\boldsymbol{w}^-$ which is classified as $c(\boldsymbol{w}^+)$. In this normalized setting where the distance between $\boldsymbol{w}^+$ and $\boldsymbol{w}^-$ equals one, the gray area indicates all points classified as $c(\boldsymbol{w}^+)$ for a given θ.

Thus, the RelSim of $\boldsymbol{x}'$ is always smaller than for the projected point $\boldsymbol{x}$ lying on the line defined by both prototypes. For $\boldsymbol{x} = \lambda \boldsymbol{w}^-$, we obtain $d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^+) = (\lambda d_-)^2$ and $d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^-) = (d_- - \lambda d_-)^2$ where $d_- = \|\boldsymbol{w}^-\|_\Lambda$. Hence, it is sufficient to find all λ for which the following inequality holds:

$$\frac{(d_- - \lambda d_-)^2 - (\lambda d_-)^2}{(d_- - \lambda d_-)^2 + (\lambda d_-)^2} \geq \theta. \tag{4.10}$$

Since the set of all λ for which Inequality (4.10) holds is a connected[1] and closed subset of $\mathbb{R}$, and thus an interval, it suffices to find the boundary points, i. e. solve Inequality (4.10) in the equality case:

$$\begin{aligned} & \theta = \frac{(d_-)^2\left((1-\lambda)^2 - \lambda^2\right)}{(d_-)^2\left((1-\lambda)^2 + \lambda^2\right)} \\ \Leftrightarrow \quad & 0 = 2\theta\lambda^2 - 2(\theta - 1)\lambda + \theta - 1 \\ \Leftrightarrow \quad & \lambda_\pm = \frac{\theta - 1}{2\theta} \pm \frac{\sqrt{1-\theta^2}}{2\theta} = \frac{\theta - 1 \pm \sqrt{1-\theta^2}}{2\theta} \end{aligned}$$

For a given $\theta \in (0, 1]$, the point $\boldsymbol{x}$ is classified as $c(\boldsymbol{w}^+)$ if $\lambda \in [\lambda_-, \lambda_+]$. Putting the conditions into $\boldsymbol{x} = \boldsymbol{w}^+ + \lambda(\boldsymbol{w}^- - \boldsymbol{w}^+)$ yields the claim.

□

Note that $\lambda_+ = {}^{(\theta-1+\sqrt{1-\theta^2})}/_{2\theta}$ is larger than 0 and gives the distance from $\boldsymbol{x}$ and the closest prototype $\boldsymbol{w}^+$ relative to the distance between both prototypes. Further, it encodes the uncertainty due to another likely class. In contrast, λ_- is smaller than 0 and describes the point that is the furthest point from both prototypes and is still classified as $c(\boldsymbol{w}^+)$. It encodes uncertainty due to a lack of data.

1 by considering the first derivative of the left-hand side of Inequality (4.10) and monotony arguments

Furthermore, by considering λ as a function of θ, due to l'Hôspital's rule

$$\lim_{\theta\to 0} \lambda_+(\theta) = \lim_{\theta\to 0} \frac{\theta - 1 + \sqrt{1-\theta^2}}{2\theta} = \lim_{\theta\to 0} \frac{1 - 2\theta(1-\theta^2)^{-0.5}}{2} = \frac{1}{2}$$

and

$$\lim_{\theta\to 1} \lambda_+(\theta) = \lim_{\theta\to 1} \frac{\theta - 1 + \sqrt{1-\theta^2}}{2\theta} = 0.$$

So, Theorem 5 is consistent in the limits to the observation we have made in Section 4.2.2 where we introduced the RelSim: Once again, it shows that on the decision boundary defined by two prototypes, i. e. where $d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^+) = d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^-)$, the RelSim is equal to zero and no point between two prototypes is rejected if θ becomes arbitrarily small. For $\theta = 1$, all points not coinciding with one prototype are rejected. On the other hand,

$$\lim_{\theta\to 0} \lambda_-(\theta) = \lim_{\theta\to 0} \frac{\theta - 1 - \sqrt{1-\theta^2}}{2\theta} = \lim_{\theta\to 0} \frac{-2}{2\theta} = -\infty$$

and

$$\lim_{\|\boldsymbol{x}\|\to\infty} \mathrm{r}_{\mathrm{RelSim}}(\boldsymbol{x}) = 0$$

show that points too far away from the prototypes will always be rejected for any $\theta > 0$. To visualize the behavior of λ, both graphs are plotted in Figure 4.2 for all $\theta \in (0, 1]$.

From these observations, we gain the following two corollaries.

Corollary 1 (Reject uncertain samples)**.** *Assume a GMLVQ model with a reject option induced by RelSim and a threshold $\theta \in (0, 1)$ is given. Denote $w_{\min} := \min_i \min_j \{d_\Lambda(\boldsymbol{w}_i, \boldsymbol{w}_j) \mid \mathrm{c}(\boldsymbol{w}_i) \neq \mathrm{c}(\boldsymbol{w}_j)\}$ with $i, j \in \{1, \ldots, w\}$. Every point closer to any class boundary than*

$$w_{\min}\left(\frac{1}{2} - \frac{\theta - 1 + \sqrt{1-\theta^2}}{2\theta}\right) \tag{4.11}$$

is rejected.

Proof. From the proof of Theorem 5, it follows that the RelSim is changed the most if a point $\boldsymbol{x}$ is shifted on the line defined by its two closest prototypes $\boldsymbol{w}^+$ and $\boldsymbol{w}^-$ where $\mathrm{c}(\boldsymbol{w}^+) \neq \mathrm{c}(\boldsymbol{w}^-)$. Further, we know from the theorem that the point $\boldsymbol{x} = \boldsymbol{w}^+ + \lambda_+(\boldsymbol{w}^- - \boldsymbol{w}^+)$ is also the closest one to $\boldsymbol{w}^-$ as well as to their decision boundary which is still classified as $\mathrm{c}(\boldsymbol{w}^+)$. Thus, all points $\boldsymbol{x} = \boldsymbol{w}^+ + \lambda(\boldsymbol{w}^- - \boldsymbol{w}^+)$ with $\lambda \in (\lambda_+, 1/2]$ are rejected by $\mathrm{r}_{\mathrm{RelSim}}(\boldsymbol{x}) < \theta$.

The claim in Equation (4.11) follows directly by defining the smallest margin of the model, i. e. the closest two prototypes belonging to different classes measured in the semi-norm induced by Λ. □

Corollary 2 (Reject outlier)**.** *Assume a GMLVQ model with a reject option induced by RelSim and a threshold $\theta \in (0, 1)$ is given. For a point $\boldsymbol{x}$, denote $\boldsymbol{w}^+$ as its closest prototype and $\boldsymbol{w}^-$ as the closest one from another class. Then, $\boldsymbol{x}$ is rejected if*

$$d_\Lambda(\boldsymbol{x}, \boldsymbol{w}^+) > d_\Lambda(\boldsymbol{w}^+, \boldsymbol{w}^-)\frac{1 - \theta + \sqrt{1-\theta^2}}{2\theta}. \tag{4.12}$$

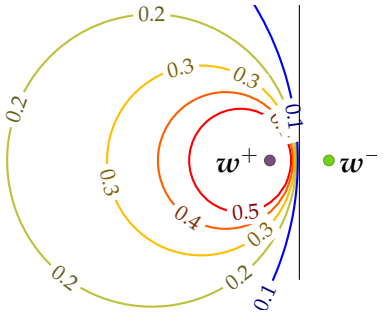


Figure 4.3: Contour plot for Relative Similarity. The different levels in the plot depict all points where $r_{RelSim}(\boldsymbol{x})$ equals θ for 0.1 (drawn as –), 0.2, 0.3, 0.4, and 0.5 (drawn as –), i. e. all points on the curves and inside the circles would not be rejected by the RelSim and would be classified as $c(\boldsymbol{w}^+)$ for that given θ.

Proof. From the proof of Theorem 5 it follows that the furthest point from $\boldsymbol{w}^+$ and $\boldsymbol{w}^-$ which is classified as $c(\boldsymbol{w}^+)$ is $\boldsymbol{x} = \boldsymbol{w}^+ + {}^{(\theta-1-\sqrt{1-\theta^2})}/_{2\theta}(\boldsymbol{w}^- - \boldsymbol{w}^+)$. For all points $\boldsymbol{x}'$ further away from $\boldsymbol{w}^+$ it holds that $r_{RelSim}(\boldsymbol{x}') < r_{RelSim}(\boldsymbol{x})$ and they will be rejected. □

Note that Corollary 2 gives a neighborhood for each prototype where the model classifies samples. Conversely, all points outside the union of all those neighborhoods are rejected by r_{RelSim}, for a given θ and model. Further, in the feature space, they are hyper-ellipsoids because of the projection with Ω. For the two-dimensional case, neighborhoods are depicted in Figure 4.3 as circles for $\theta \in [0.1, 0.5]$. It is apparent that for a smaller θ, all samples close to the decision boundary are classified as well as those far away from any prototype.

Further, from Corollaries 1 and 2, it follows that Relative Similarity rejects samples close to decision boundaries and outliers, too. This is an advantage to other methods, as detecting both types requires two different reject mechanisms (Hendrickx et al. 2021).

4.4 PROBABILISTIC OUTPUTS FOR LVQ

We are interested in probabilities generated from deterministic LVQ outputs via post-processing. We do not consider probabilistic LVQ versions such as RSLVQ, since these models do not yield unbiased estimates of the underlying probabilities (Fischer, Hammer, and Wersing 2016). Hence, their output does also benefit from additional post-processing and scaling of its outputs as shown in the work by Fischer, Hammer, and Wersing (2015) and Schneider, Biehl, and Hammer (2010). Instead, we propose to mimic an approach that is popular in the context of the SVM: a data-dependent scaling of the real-valued SVM output for pairwise class distinction followed by a coupling of these pairwise probabilities.

4.4.1 Adaptive Scaling of Pairwise Probabilities

The SVM provides uncalibrated outputs to distinguish two classes in terms of the distance to the decision boundary. Similarly, for LVQ models, a possible output would be the distance to the closest prototype, which is also unbounded. Platt (1999) proposes fitting a sigmoid function after training to turn these uncalibrated outputs into posterior probabilities. Assume a binary classification problem and that the output of a classifier is given as $\mathrm{h}(\boldsymbol{x}) \in \mathbb{R}$. Then, a calibration function in the form of a sigmoid is proposed as

$$\mathbb{P}(y = 1 \mid \mathrm{h}(\boldsymbol{x})) = \frac{1}{1 + \exp(a\,\mathrm{h}(\boldsymbol{x}) + b)} \tag{4.13}$$

where a and b are real-valued parameters adapted based on the given data. These parameters are determined by minimizing the empirical log-likelihood from a training or a calibrating set $\{(\mathrm{h}(\boldsymbol{x}_i), y_i) \in \mathbb{R} \times \{0, 1\} \mid i = 1, \ldots, n\}$:

$$\arg\min_{a,b} \; -\sum_{i=1}^{n} y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \tag{4.14}$$

where

$$p_i = \frac{1}{1 + \exp(a\,\mathrm{h}(\boldsymbol{x}_i) + b)}. \tag{4.15}$$

Optimization takes place by gradient techniques. To avoid over-fitting, typically, distinct training sets are used for training the classifier and fitting the sigmoid function's parameters a and b. In addition, Platt (1999) proposes to manipulate the target values y_i for the two output classes as follows: Assume N_+ positive examples, i. e. samples belonging to class 1, and N_- negative ones, belonging to class 0, are contained in the data set used for fitting the sigmoid function. Then, manipulated values y_+ and y_- (instead of 1 and 0, respectively) are proposed, where

$$y_+ = \frac{N_+ + 1}{N_+ + 2} \;\text{ and }\; y_- = \frac{1}{N_- + 2}. \tag{4.16}$$

This redefinition ensures that no degenerate step-function, i. e. a 0-1 function, is learned. A more detailed treatment and a justification of these target values can be found in (Platt 1999).

This procedure can be transferred to LVQ provided a classification into two classes is considered. We will use pairwise GMLVQ models which distinguish class l and m for all pairs of classes and according to subsets of the training set. Note that the resulting classifier is linear if one prototype per class is used, and it becomes nonlinear for more than one prototype per class. This classifier aims for a large margin classification similar to SVM, but it represents the data in terms of typical prototypes rather than archetypical values at the decision boundary. In this chapter, we will discuss to base the Platt Scaling on the continuous output offered by the RelSim, i. e. to choose

$$\mathrm{h}(\boldsymbol{x}) := \mathrm{r}_{\mathrm{RelSim}}(\boldsymbol{x}) = \frac{d(\boldsymbol{x}, \boldsymbol{w}^-) - d(\boldsymbol{x}, \boldsymbol{w}^+)}{d(\boldsymbol{x}, \boldsymbol{w}^+) + d(\boldsymbol{x}, \boldsymbol{w}^-)}. \tag{4.17}$$

This scaling yield estimates r_{lm} of the pairwise probabilities

$$r_{lm}(\boldsymbol{x}) := \mathbb{P}(y = l \mid y = l \vee m, \boldsymbol{x}) \tag{4.18}$$

for every sample $\boldsymbol{x}$ and pairs of classes l and m.

Algorithm 4.1: Pairwise Coupling by Hastie and Tibshirani (1997)

Input: pairwise probabilities $r_{lm} = \mathbb{P}(y = l \mid y = l \vee m)$
Start with some initial $p_k > 0 \; \forall \, k \in \{1, \ldots, c\}$, e. g. $p_k = 1/c$
Compute $\rho_{lm} = p_l/(p_l + p_m)$
repeat
  for $l = 1, \ldots, c$ **do**
    $\alpha = \frac{\sum_{m:m \neq l} n_{lm} r_{lm}}{\sum_{m:m \neq l} n_{lm} \rho_{lm}}$
    $p_l \leftarrow \alpha p_l$
    $\rho_{lm} \leftarrow \frac{\alpha \rho_{lm}}{\alpha \rho_{lm} + \rho_{ml}}$
    $\rho_{ml} \leftarrow 1 - \rho_{lm}$
  end for
  $\boldsymbol{p} \leftarrow \boldsymbol{p} / \sum_{k=1}^{c} p_k$ ▷ normalize
until all α are close to ones

4.4.2 Probabilities from Pairwise Classifiers

For multi-class classification problems, one simple method is to combine pairwise comparisons by voting (Friedman 1996). Let $p_l^{\texttt{voting}}$ denotes the number of pairs (l, m) where class l is preferred to class m, i. e. $m \in \{1, \ldots, c\} \setminus \{l\}$. A normalized value (between 0 and 1) is obtained by dividing the results by the number of classes. We will use this scheme as a baseline in our experiments.

A few, more refined techniques have been proposed to turn such pairwise probability estimates r_{lm}, such as those from Equation (4.18), into class-wise probabilities $p_l := \mathbb{P}(y = l \mid \boldsymbol{x})$ of class l which differ in complexity. In this chapter, we consider and compare four methods as concerns their suitability for LVQ variants. They derive class probabilities p_l by suitable pairwise coupling. We denote the resulting estimates as $\boldsymbol{p}^\star$ according to the method $\star$. All methods assume inputs of pairwise probability estimate r_{lm}, e. g. the result from the Platt Scaling, for every pair of classes. Further, a symmetric completion holds, i. e. $r_{lm} + r_{ml} = 1$.

HT: Hastie and Tibshirani (1997) introduce the auxiliary variables $\rho_{lm} = p_l/(p_l + p_m)$ and propose to minimize the average weighted Kullback-Leibler divergence between r_{lm} and ρ_{lm} with respect to the variables p_k for one point:

$$D_{KL}(r_{lm} \parallel \rho_{lm}) = \sum_{l \neq m} n_{lm} r_{lm} \log \frac{r_{lm}}{\rho_{lm}} \tag{4.19}$$

where n_{lm} is the number of samples belonging to class l or to class m. This formula is equivalent to finding a vector $\boldsymbol{p} = (p_1, \ldots, p_c)$ that satisfies

$$\sum_{m:m \neq l} n_{lm} \rho_{lm} = \sum_{m:m \neq l} n_{lm} r_{lm} \quad \forall \, l \in \{1, \ldots, c\}\}$$
$$\text{such that} \quad \sum_{k=1}^{c} p_k = 1 \text{ and } p_k > 0 \; \forall \, k \in \{1, \ldots, c\}. \tag{4.20}$$

We implemented the iterative algorithm as given by Wu, C.-J. Lin, and Weng (2004) and adapted it to our notation in Algorithm 4.1. In this chapter, we will refer to the class probabilities as HT.

PKPD: An approach that is easier to implement is considered by Price et al. (1994). They emphasize the relation

$$\left(\sum_{m:m\neq l} \mathbb{P}(y = l \vee m \mid \boldsymbol{x})\right) - (c-2)\,\mathbb{P}(y = l \mid \boldsymbol{x})$$
$$= \sum_{k=1}^{c} \mathbb{P}(y = k \mid \boldsymbol{x}) = 1 \tag{4.21}$$

and obtain the estimates

$$p_l = \frac{1}{\sum_{m:m\neq l} \frac{1}{r_{lm}} - (c-2)} \tag{4.22}$$

disregarding the constraint $\sum_{k=1}^{c} p_k = 1$. After calculating all p_l, a normalization of $\boldsymbol{p}$ takes place. We refer to this approach as PKPD

WLW1/2: Two related approaches have been proposed by Wu, C.-J. Lin, and Weng (2004). They illustrate possible issues for HT. It relies on the assumption $p_l + p_m \approx 2/c$ which can yield erroneous results when the class probabilities are not equally distributed. Instead, they suggest solving the system:

$$p_l = \sum_{m:m\neq l} \frac{p_l + p_m}{c-1} r_{lm} \tag{4.23}$$
$$\text{such that} \quad \sum_{k=1}^{c} p_k = 1$$
$$\text{and} \quad p_k \geq 0 \;\forall k.$$

Rewriting this problem leads to the following convex problem:

$$\min_{\boldsymbol{p}} \quad \sum_{l=1}^{c} \left(\sum_{m:m\neq l} r_{ml} p_l - \sum_{m:m\neq l} r_{lm} p_m\right)^2 \tag{4.24}$$
$$\text{such that} \quad \sum_{k=1}^{c} p_k = 1,\; p_k \geq 0 \;\forall k.$$

A slightly different phrasing is obtained by the second approach:

$$\min_{\boldsymbol{p}} \quad \sum_{l=1}^{c} \sum_{m:m\neq l} (r_{ml} p_l - r_{lm} p_m)^2 \tag{4.25}$$
$$\text{such that} \quad \sum_{k=1}^{c} p_k = 1,\; p_k \geq 0 \;\forall k.$$

Both minimization problems can be written as a linear system with c and $c+1$ equations, respectively (Wu, C.-J. Lin, and Weng 2004). We refer to these approaches as WLW1 and WLW2.

These four approaches can immediately be used to transfer pairwise probabilities of pairwise LVQ solutions to global classification probabilities, and according to reject options.

4.5 TIME INTEGRATION

Classification schemes, as introduced in this thesis, provide an instantaneous certainty measure and classification with reject option for a given observation $\boldsymbol{x}$. Now, we assume that a stream of observations is given $\ldots, \boldsymbol{x}^{\tau}, \boldsymbol{x}^{\tau+1}, \boldsymbol{x}^{\tau+2}, \ldots$ where every single observation $\boldsymbol{x}^{\tau}$ can be classified independently, but the nature of the process suggests a smoothness of the labels, i. e. a change in the labeling of subsequent samples $\boldsymbol{x}^{\tau}$ and $\boldsymbol{x}^{\tau+1}$ happens only rarely. Such settings can be monitoring systems of, for example, industrial machines that state the machine's status or are watching for failures and changes. Even personal use cases exist, like smartwatches or other peripheral device classifying the performed activity immediately. In all these settings, the classification of the signals at one point in time might be uncertain, and integration of classification outputs over subsequent time steps can be beneficial.

We will propose a suitable fusion scheme that relies on the class probability estimates p_l as introduced in Section 4.4. More specifically, we denote $p_l^{\tau} := p_l(\boldsymbol{x}^{\tau})$ as the class probability of the sample $\boldsymbol{x}$ for class l at time point τ. Then, for every $t > 0$, we define the average class' confidence of l in the time window around τ of size $t + 1$ as

$$p_l^{\tau}(t) := \frac{\prod_{w \in \mathcal{W}} p_l^{\tau+w}}{\sum_{k=1}^{c} \prod_{w \in \mathcal{W}} p_k^{\tau+w}} \tag{4.26}$$

where $\mathcal{W} = \{-\lceil t/2 \rceil, \ldots, -1, 0, 1, \ldots, \lfloor t/2 \rfloor\}$. This definition depends on the assumption that p_l^{τ} resembles a probability which is then combined over a short period via product probabilities. The resulting values are contained in the interval $[1/c, 1]$, with $1/c$ denoting low confidence, while 1 resembles a high confidence. Since we normalize with all possible classes, only mildly confident but consistent classification of a class is reinforced. Whereas settings with subsequent varying classes yield unclear decisions, corresponding to an uncertain classification in the time interval. Further, this time integration has the consequence that in the case of a subsequent uncertain and certain classification, the more certain one will determine the final result.

4.6 EXPERIMENTS

In the following, first, we compare the result of classification with the reject option based on probabilistic plugin rules from pairwise LVQ methods as introduced in Section 4.4 and multi-class LVQ with the Relative Similarity as a surrogate for reject. Later, we will investigate the effectiveness of using the class probabilities for time integration which yields a more robust classification.

4.6.1 Data Sets

We evaluate our approach for one artificial data sets with known ground truth for the Bayes optimal reject and six real-world benchmarks; these in-

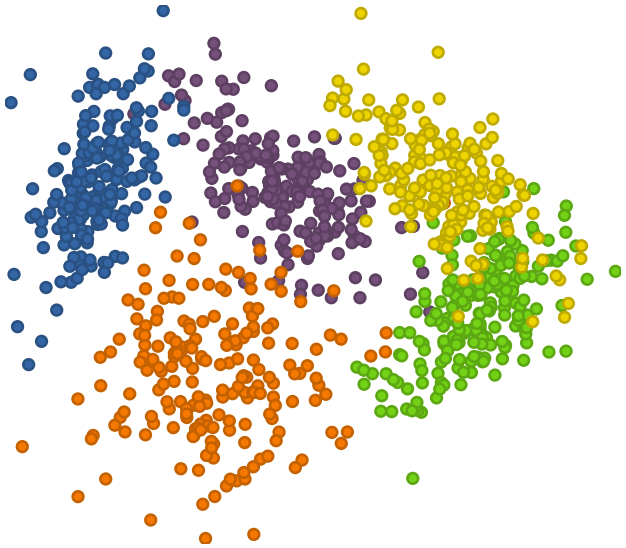


Figure 4.4: Scatter plot of a subset of the artificial data set (first 200 samples per class). One can easily see that some samples are misclassified due to overlapping regions.

clude those already proposed in the article by Fischer, Hammer, and Wersing (2015).

The *artificial data set* contains five overlapping two-dimensional Gaussian clusters. Each Gaussian represents one class and consists of 1500 samples. The centers are $\mu_1 = (2, 3)$, $\mu_2 = (5, 1)$, $\mu_3 = (1, 0)$, $\mu_4 = (-1, 3)$ and $\mu_5 = (4.5, 3)$ and the covariances are rotated by $\pi/4$ for classes 1, 2, and 5 and rotated by $\pi/8$ for class 4. Class 3 has an identity matrix as the covariance. For each sample, we gain confidence by evaluating the PDF of the Gaussian distributions at that point for each class and by normalizing with their sum. A subset of the data set with 200 samples per class is depicted in Figure 4.4. The other benchmarks are Coil-20, Image Segment, Letter, MCE-Nose, Motion Tracking, and USPS which are all already introduced in Section 2.3.1.

As a recap, the characteristics of these data sets are summarized in Table 4.1 again.

Table 4.1: Number of samples, features, and classes for each data set used in the experiments

Data set	# samples	# features	# classes
Artificial	7500	2	5
COIL-20	1440	16 384	20
Img. Seg.	2310	19	7
Letter	20 000	16	26
MCE-Nose	2257	4	4
Mot. Track.	10 299	561	6
USPS	9298	256	10

All prototype-based classifiers are trained with one prototype per class. For pairwise LVQ, we use the GMLVQ scheme. For multi-class classification, we use GMLVQ and LGMLVQ.

4.6.2 Evaluation Scheme

We repeat a ten-fold cross-validation for each benchmark ten times. At each of the ten cross-validations, sequentially nine folds are used as the training set while one fold holds as the validation set. A z-score is fitted on the training set. If the variance is zero for one feature, i. e. all values are the same for this, the feature is removed in both sets. Then, a further split on the training set into five folds is performed avoiding over-fitting of the sigmoid functions (as it is discussed in Section 4.4.1). Four folds are used to train $c(c-1)/2$ one-vs-one GMLVQ models, where only the samples of the class l and m are taken for each pair of classes. The sigmoid functions are fitted with one fold put aside for each couple. Next, for each instance in the validation set, we apply first the same z-score defined by the training set and second the pairwise coupling methods to obtain class probabilities. For comparison, we use GMLVQ and LGMLVQ trained with the same training sets and its induced RelSim measure.

All methods were implemented in Matlab® using the toolbox by Bunte, Schneider, et al. (2012) for the GLVQ variants. To also evaluate the elapsed time for training and testing, all computations were running on the same computer with two Intel® Xeon® X5690 CPUs and 48 GB memory.

In the evaluation, we report the mean of the area under the Accuracy-Reject-Curve (ARC) (Nadeem, Zucker, and Hanczar 2010) over all hundred runs. The ARC refers to the following principle: for a given θ the test set is decomposed into two disjoint sets $\mathcal{X} = \mathcal{X}_\theta \cup \mathcal{R}$, where $\mathcal{X}_\theta$ is the set with remaining samples and $\mathcal{R}$ contains the rejected samples. The accuracy is evaluated on the samples in $\mathcal{X}_\theta$ and the reject-rate is the ratio of $\mathcal{R}$ and $\mathcal{X}$. The threshold θ starts from no reject ($\theta = \min_i \mathrm{r}(\boldsymbol{x}_i) \sim 1/c$) to full reject ($\theta = \max_i \mathrm{r}(\boldsymbol{x}_i) \sim 1$) where no sample is classified anymore. To ensure that all curves meet $(1, 1)$, meaning that all points are rejected and by definition the accuracy equals one, we explicitly add this point to the graphs. This step is crucial for comparing the area under the curve because the curves can otherwise stop at different reject-rates and can thereby influence the area a lot.

4.6.3 Results

First, we evaluate the artificial data set with known ground truth. Figure 4.5 displays the mean ARCs for PKPD as exemplary probabilistic transformation, for LGMLVQ with RelSim, and the Bayes Optimal Classifier. The curves for the other three coupling methods and of GMLVQ are alike PKPD and LGMLVQ, respectively, and we omit them in the plot for readability and clarity. The means of the areas under the ARCs are 0.9876 for PKPD, 0.9805 for GMLVQ, 0.9806 for LGMLVQ, and 0.9916 for the Bayes Optimal Classifier. We observe good performances of all methods for reject-rates between 0 and 0.1. The errors mostly occur from ambiguity in the overlapping regions, e. g. the yellow and green class around $(5, 2)$ (see Figure 4.4). The performance of GMLVQ and LGMLVQ are so similar because the data set has only two dimensions. Thus, there exist no local projections to separate one class from the others. For higher reject-rates the probabilistic classifier outperforms LGM-

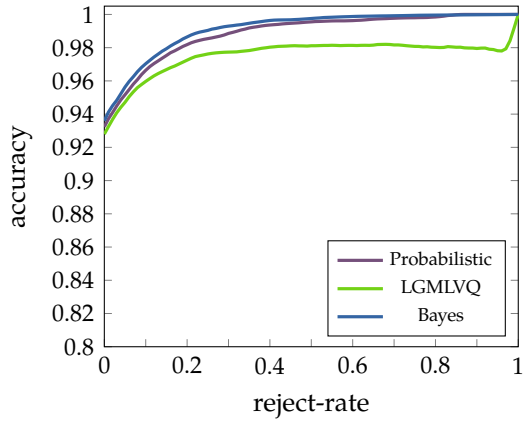


Figure 4.5: Averaged Accuracy-Reject-Curves for PKPD, LGMLVQ, and the Bayes Optimal Classifier. The areas under the ARCs are 0.9876, 0.9806, and 0.9916, respectively.

Table 4.2: Mean and standard deviation (in parenthesis) of accuracy rates for the benchmarks without rejects. The best accuracy of one data set is bold-faced and significantly better or worse values are underscored.

Method	COIL-20	Img. Seg.	Letter	MCE-Nose	Mot. Track.	USPS
HT	0.997 (0.0050)	0.952 (0.0133)	0.853 (0.0086)	**0.766** (0.0279)	0.936 (0.0063)	0.954 (0.0064)
PKPD	0.997 (0.0053)	0.953 (0.0133)	0.849 (0.0087)	0.765 (0.0277)	0.936 (0.0063)	0.954 (0.0066)
WLW1	0.997 (0.0053)	0.952 (0.0131)	0.853 (0.0086)	0.765 (0.0277)	0.936 (0.0063)	0.954 (0.0065)
WLW2	0.997 (0.0055)	**0.953** (0.0131)	0.85 (0.0090)	0.765 (0.0278)	**0.936** (0.0065)	0.953 (0.0067)
Voting	0.996 (0.0060)	0.95 (0.0147)	0.843 (0.0091)	0.759 (0.0254)	0.935 (0.0065)	0.951 (0.0071)
GMLVQ	0.954 (0.0171)	0.913 (0.0170)	0.702 (0.0103)	0.714 (0.0332)	0.91 (0.0087)	0.908 (0.0077)
LGMLVQ	**0.999** (0.0035)	0.937 (0.0156)	**0.877** (0.0071)	0.765 (0.0363)	0.931 (0.0078)	**0.96** (0.0068)

LVQ. A slight drop for LGMLVQ is observed for a reject-rate between 0.6 and 0.8 due to rejects of correct samples. This is one indication that RelSim has no semantic meaning besides its relative ordering and sometimes gives smaller confidences in regions where no other class is contradicting. The ARC for the PKPD is very close to the Bayes Optimal Classifier's curve at all reject-rates which indicates that the post-processed probabilities seem to be reasonable.

Table 4.2 reports the mean of the accuracy for all classifiers with no reject options. The numbers are bold-faced for the best result of the data sets. All numbers which are significantly better or worse than all others are underlined. Here, we used a test from (Rodríguez-Fdez et al. 2015) with a Friedman test for ranking the scores, a Holm test for the post-hoc, and a significance level of 0.05. The averaged areas under the ARCs are reported in Table 4.3 and the averaged ARCs are displayed in Figure 4.6.

It can be observed that, for the given benchmarks, the accuracies for all four coupling methods do not differ much. This can be attributed to the fact that the class distribution is approximately uniform. Hence, the most efficient method, i. e. PKPD, is suited in this case.

Another observation is that a probabilistic treatment and an according scaling can drastically improve the accuracy as compared to multi-class GMLVQ schemes. For the latter, all accuracies on all benchmarks are significantly worse. Hence, the posterior scaling of the certainties has a rewarding

Table 4.3: Mean and standard deviation (in parenthesis) of the areas under the ARCs. The best values are bold-faced.

Method	COIL-20	Img. Seg.	Letter	MCE-Nose	Mot. Track.	USPS
HT	1.0000 (0.000 05)	0.9946 (0.0023)	0.9668 (0.0037)	0.9393 (0.0135)	**0.9900** (0.0018)	0.9942 (0.0023)
PKPD	1.0000 (0.000 07)	0.9923 (0.0047)	0.9666 (0.0035)	0.9370 (0.0140)	0.9898 (0.0021)	0.9933 (0.0027)
WLW1	1.0000 (0.000 06)	**0.9946** (0.0023)	0.9668 (0.0037)	0.9389 (0.0134)	0.9900 (0.0018)	0.9942 (0.0023)
WLW2	1.0000 (0.000 07)	0.9941 (0.0026)	0.9676 (0.0035)	0.9382 (0.0137)	0.9899 (0.0019)	0.9941 (0.0022)
GMLVQ	0.9978 (0.001 46)	0.9847 (0.0052)	0.9173 (0.0053)	0.8749 (0.0268)	0.9763 (0.0037)	0.9867 (0.0024)
LGMLVQ	**1.0000** (0.000 02)	0.9922 (0.0038)	**0.9856** (0.0014)	**0.9399** (0.0147)	0.9870 (0.0027)	**0.9972** (0.0014)

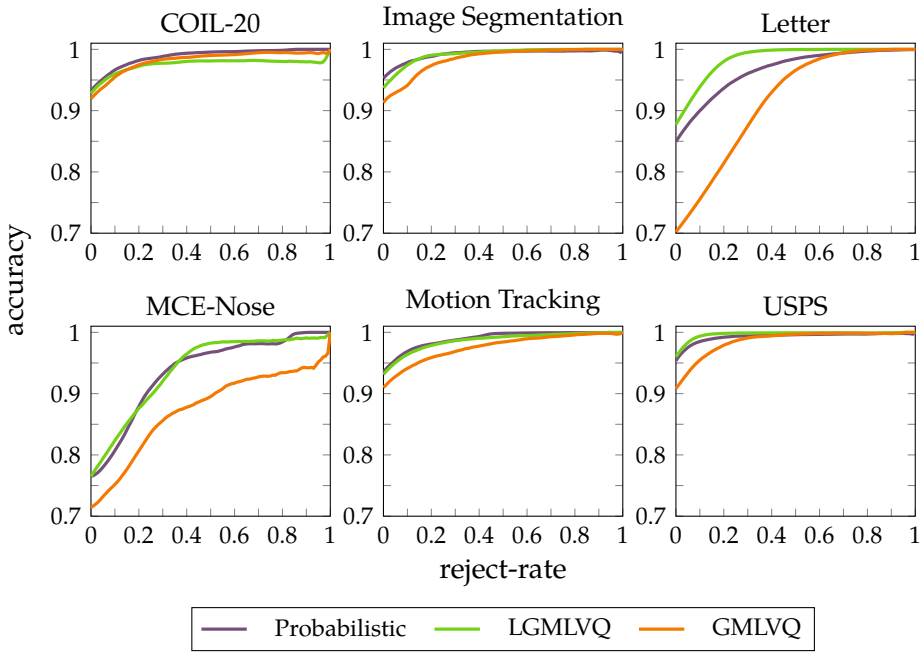


Figure 4.6: Averaged Accuracy-Reject-Curves over all 100 runs for each benchmark

effect and enables better coordination of the otherwise unscaled values. The results also show that the performance of GMLVQ decreases with a larger number of classes. This effect is diminished if local metric learning is used, as can be observed in the last rows in the Tables 4.2 and 4.3. For many data sets, an explicit probabilistic treatment based on pairwise LVQ and direct multi-class LVQ with local metric learning yields comparable results; a few data sets exist where one of the methods is significantly better. Hence, albeit coordination of the classification confidence is not aimed for in the LGMLVQ costs, the localized treatment of the metric seems to have this effect. In particular, this seems a viable option if the number of classes is large, such as for COIL, LETTER, and USPS.

The reported areas under the ARCs do not differ for all four coupling methods. From this and the ARCs, one can deduce that the class probabilities and the rejected samples are almost identical. Therefore and for clarity, we only show the curves for PKPD as a representative in Figure 4.6. Here, analog conclusions can be drawn: First, for GMLVQ, the accuracies at higher reject-rates are consistently lower than those of the other methods. Second, the curves for LGMLVQ and our method are in the same shape, except for Letter.

Table 4.4: Averaged elapse time (in seconds) for training in the first three rows and testing in the last six rows.

Method	COIL-20	Img. Seg.	Letter	MCE-Nose	Mot. Track.	USPS
One vs. One	36.9250	9.4710	105.7960	0.7670	28.7520	32.5060
GMLVQ	3.4020	2.6320	26.2580	1.1150	9.5550	8.8300
LGMLVQ	12.6540	4.0310	78.7070	1.6230	28.8780	21.5910
HT	2.9900	3.3000	186.0000	1.1100	35.1000	33.8000
PKPD	0.0027	0.0030	0.0443	0.0030	0.0140	0.0138
WLW1	0.0071	0.0064	0.1300	0.0072	0.0341	0.0341
WLW2	0.0096	0.0093	0.1730	0.0104	0.0487	0.0575
GMLVQ	0.0014	0.0006	0.0095	0.0002	0.0049	0.0039
LGMLVQ	0.0011	0.0004	0.0051	0.0003	0.0032	0.0025

The accuracies strongly increase in the beginning. Often, a reject-rate of 0.2 is sufficient and the slope flats out for higher rates. This is the most attractive region because, in practice, someone is looking only for small reject-rates such that not too many correct classified samples are rejected.

The means of the elapsed time are reported in Table 4.4. For training, the elapsed time severely depends on the situation at hand. Partially, this is due to the different number of classes, hence, various classifiers are required for pairwise coupling. It is also affected by the complexity of the single learning problems since LBFGS converges in different numbers of epochs. In the classifying scenario, HT sticks out. The class probabilities are calculated interactively, and several steps have to be made to find the solution. Also, the elapsed time for WLW1 is a bit shorter than for WLW2 because the linear system, which is solved there, is smaller. Sometimes, finding the probabilities with PKPD takes twice as long as the classification for GMLVQ or LGMLVQ, see COIL-20, and sometimes a bit longer, e. g. 8 times as long for Letter. However, for the last three methods, the classification of new samples can still be performed in a reasonable time.

4.6.4 Time Integration

We evaluate the effectiveness of time integration with the two benchmarks Motion Tracking and MCE-Nose. For the first one, we do a leave-one-out cross-validations for keeping time dependencies in the test set. The test set contains the records for one single volunteer, and we train the models with all others 29. Every volunteer changes their movement on average 13.33 times (between 12 and 17) and each set contains roughly 340 samples. For the second benchmark, we decompose the sample into eight disjoint sets of temporal sequences, and we sort the samples in the test set by their occurrence in the original data set. Because of the experimental setup, each odor is tested four times and the classes change 15 times. Thus, on average 17.6 subsequent samples belong to the same class.

In Figure 4.7 the ARCs for both benchmarks and different window sizes are plotted. Time integration yields an improved accuracy at nearly every reject-rate. For example, with $t = 4$ and a reject-rate of 0.2 the accuracy-rate

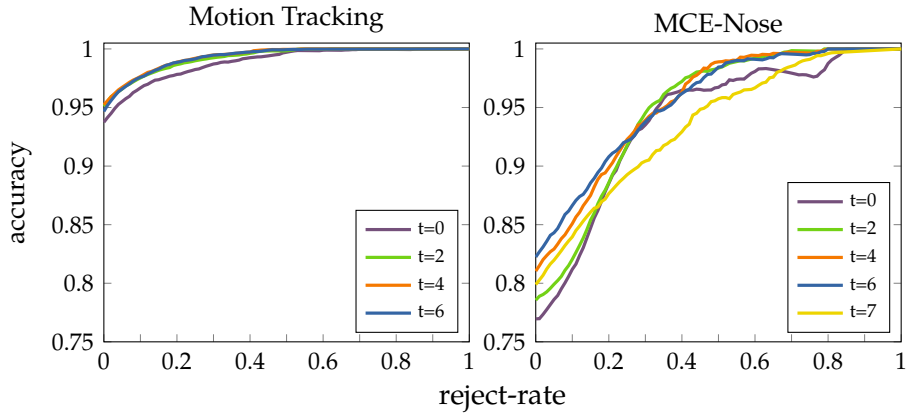


Figure 4.7: Averaged Accuracy-Reject-Curves for different values of t. The purple line ($t = 0$) is the baseline without integration. The areas under the ARCs are 0.9891, 0.9928, 0.9936, and 0.9933 for Motion Tracking and 0.9399, 0.9492, 0.9542, 0.9553, and 0.9360 for the MCE-Nose benchmark for the given different window sizes t.

Table 4.5: Averaged areas under the Accuracy-Reject-Curves for different window sizes t. Best values are bold-faced.

t	0	1	2	3	4	5	6	7	8	9	10
Mot. Track.	0.9891	0.9899	0.9928	0.9922	**0.9936**	0.9925	0.9933	0.9921	0.9925	0.9914	0.9916
MCE-Nose	0.9399	0.9241	0.9492	0.9334	0.9542	0.9372	**0.9553**	0.9360	0.9518	0.9310	0.9443

is 0.988 74 compared to 0.978 25 for the approach without time integration. With a reject-rate of 0.54 no sample is misclassified anymore for the same t.

Similar behavior is observed for the MCE-Nose data set. Here, the initially comparably low accuracy is justified by the recording of the data itself. Whenever a new odor is recorded, the sensors take some samples to react to the new signals. Nevertheless, the accuracy can be improved by time integration for small reject-rates up to 0.2 and small even window sizes, i. e. $t = 4$ or $t = 6$. For a higher reject-rate (0.5), we achieve an accuracy of 0.989 with $t = 4$ as compared to 0.967. If we chose larger window sizes the accuracy drops due to the misclassification of points in the region where the class changes. In Table 4.5, we list the area under the ARC for different window sizes. One can observe that an odd value of t is worse than an even one. As an example, we also plot the ARC for $t = 7$ and MCE-Nose in Figure 4.7. Note that time integration relies on a probabilistic treatment such that products yield meaningful values under the simplifying assumption of independent measurements. Therefore, we only report the results based on probabilistic values and abandon RelSim here.

4.7 SUMMARY

In this chapter we have investigated two possibilities to enhance LVQ schemes by a reject option to increase its robustness (**RQ2**): the deterministic surrogate function as introduced by Fischer, Hammer, and Wersing (2015) and a probabilistic post-processing that transfers a common practice for SVM to

LVQ techniques. For the latter, we proposed how to extend pairwise LVQ models to a suitable scaling of the probabilities and how to include this into the pairwise coupling to obtain the likelihoods of every class. Both methods can then be integrated into threshold-based reject option, yielding convincing results in a couple of benchmarks. Interestingly, probabilistic post-processing can severely enhance simple LVQ schemes while this effect is greatly diminished if local metric learning is integrated into the latter. When evaluating the respective ARCs, local metric learning often even yields slightly superior results as compared to a probabilistic post-processing.

Still, a heuristic surrogate function has the drawback that the chosen threshold for reject options has no semantic meaning since the scaling of the RelSim as a surrogate is arbitrary, while the probabilistic output can be interpreted as classification confidence. Hence, given explicit costs for a reject option, the former requires problem-dependent threshold optimization while the latter enables picking the threshold explicitly based on the given loss. Furthermore, the probabilistic output enriches the classifier with more information. Instead of revealing the certainty for only the winning class, it also gives confidences for all other classes. In a multi-class scenario, this can be of interest in regions where different classes meet. Here, the probabilities differ for various classes, i. e. each class has the probability close to $1/c$ where c is the number of involved classes in this region, whereas the RelSim reaches zero on all class boundaries no matter how many prototypes belonging to different classes have similar distances.

Further, probabilistic values offer richer possibilities to combine the results into global models, e. g. in the case of ensembles over different models or over time are considered. We presented a post-processing scheme for integrating confidence over time to gain higher certainties in the predictions and obtained promising results for two benchmarks.

5 SPARSE LEARNING VECTOR QUANTIZATION

Overview: This chapter deals with sparsity and feature selection to answer **RQ3**: *How can the model's complexity be reduced by enforcing sparsity*? We propose to adapt the cost function of the LGMLVQ model by adding a penalty term based on Group . This penalty shrinks some model parameters toward zero such that sparse feature vectors achieve comparable or even better performance. Thus, instead of using the full feature vector, a smaller one is sufficient. At the same time, the size of the relevance matrices specifying the dissimilarity measure for each prototype can also be reduced such that the models need less memory.

Publication: Parts of this chapter are based on the following publication:

- Brinkrolf, Johannes and Barbara Hammer (2020a). "Sparse Metric Learning in Prototype-based Classification". In: *28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2020, Bruges, Belgium, October 2-4, 2020*. Ed. by Michel Verleysen, pp. 375–380. URL: https://www.esann.org/sites/default/files/proceedings/2020/ES2020-138.pdf.

5.1 INTRODUCTION

Metric learning schemes based on a global or local adaptive quadratic form, as those introduced in Sections 2.2.2 and 2.2.3, lead to enhanced model interpretability. For example, we gain *feature relevances* in terms of the diagonal entries of their matrices. Yet, particularly for high-dimensional data, relevance weightings of the features are no longer easily interpretable or efficient due to the sheer number of involved features. Further, Schulz, Mokbel, et al. (2015) pointed out that possibly high feature correlations induce an increased risk of spurious relevance terms. Therefore, we aim for efficient schemes that extend LVQ models to *sparse* models relying on as few input features as possible. Such models have the potential of increased interpretability since a human observer needs to inspect only a subset of relevant features (Ribeiro, Singh, and Guestrin 2016). At the same time, a restriction to a subset of features increases model efficiency and the suitability for edge computing since only a smaller feature set needs to be measured and processed. Here, we will address **RQ3** and propose a suitable adaption of the cost function to obtain sparser models.

Considering LVQ models, for example, Biehl, Hammer, Schleif, et al. (2015) showed that popular matrix learning approaches are sparse in the sense that they tend to converge to low-rank matrices. Yet, as discussed in Section 2.2.4, low-rank matrices do not necessarily imply feature sparsity because they can still exploit plenty of features unless the relevant directions align with the coordinate axes of the original data model. Hence, other techniques have to be applied.

First, we will give a brief overview of some techniques for feature selection. Those techniques can be decomposed into three classes: *filter*, *wrapper*, and *embedded* methods.

Filter methods, e. g. mutual information (Beraha et al. 2019; Paninski 2003), Fisher's Score (Duda, Hart, and Stork 2001), or Laplacian score (He, Cai, and Niyogi 2005), provide scores that measure the amount of information that the input and the output share. These procedures are typically done separately for each feature and, therefore, are less computationally expensive compared to other methods. One drawback is that features are selected without having the model or its effect in contemplation. Here, on the one hand, correlated features might rank similarly and are selected due to the same (high) score. Likely, one of them does not enhance the classification if a correlated one is already chosen for the subset. On the other hand, it neglects the combination of features, i. e. the evaluation of two or more features together (Duch 2006).

Wrapper methods train the model multiple times by using a subset of the features (Kohavi and John 1997). Each time the current model is evaluated, and the set is adapted based on the conclusions, e. g. features are added or removed from it. One advantage of wrapper methods over filter methods is that they include the model and provide a set of features for training. Thus, correlated features are discarded if this feature does not contribute new information to the model. However, these methods are computationally more expensive since a new model and the whole training have to be optimized several times.

For these reasons, *embedded* methods are popular (Lal et al. 2006). Here, feature selection is accomplished as a part of the learning algorithm. They do not suffer from the drawbacks of filter and wrapper methods and merge their advantages. We will enlarge upon the so-called regularization where a penalty is applied to the model coefficients (Bishop 2007). Thereby, some coefficients are decayed towards zero, and the model's prediction does not change much by omitting those features. Therefore, by discarding some input dimensions, smaller feature vectors are obtained. Due to the addition of a penalty term to the cost function of the method, the training algorithm has to be adapted.

As an embedded method, one approach for GLVQ using a differentiable approximation of the ℓ_1 norm is proposed by Riedel et al. (2013). Each feature is weighted, and all weights are penalized such that some are shrunk toward zero. For GMLVQ, they suggest using the consistent matrix norm that gets intricate, especially when local relevance matrices are considered.

In the chapter, we will show how GLVQ models with local metric learning can be adapted and how to obtain sparse feature relevance vectors. We will approach this goal by combining two modeling steps: first, the cost function of GLVQ is extended by an objective based on Group (N. Simon et al. 2013). We will show that this technique improves sparseness but does not enable a sparseness that is comparable to alternatives such as random forests (Breiman 2001). Based on the insights obtained in the work by Schulz, Mokbel, et al. (2015), we add a posterior optimization step, which can be solved approximately and efficiently using Orthogonal Matching Pursuit (OMP) (Davis, S. G. Mallat, and Z. Zhang 1994)

In the following two sections, the objectives are introduced. Then, in Section 5.4, we explain our new models in detail and evaluate them on three

benchmarks with different numbers of features between 50 and 5000 in Section 5.5.

5.2 REGULARIZATION

We will use a regularization term to shrink some model coefficients towards zero such that features belonging to those coefficients do not influence the outcome of a model and could be omitted. A popular method for linear models is introduced next. Subsequently, a generalization of it to shrink a group of model coefficients is explained that is used to shrink all metric parameters close to zero of as many features as possible.

5.2.1 Least Absolute Shrinkage and Selection Operator

The *Least Absolute Shrinkage and Selection Operator (LASSO)* is coined by Tibshirani (1996) for linear models. Let $\boldsymbol{y} \in \mathbb{R}^n$ be the outcomes of the samples $\boldsymbol{x}_i \in \mathbb{R}^d$ for $i \in \{1, \ldots, n\}$. Then, LASSO is minimizing the loss

$$\mathcal{L}(\boldsymbol{\beta}) = \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + S \cdot \|\boldsymbol{\beta}\|_1 \tag{5.1}$$

with respect to the coefficient vector $\boldsymbol{\beta} \in \mathbb{R}^d$ where $[\boldsymbol{X}]_{ij} = [\boldsymbol{x}_i]_j \in \mathbb{R}^{n \times d}$, i. e. the ith row in $\boldsymbol{X}$ consists the sample $\boldsymbol{x}_i^\top$, is called the design matrix and S is a hyperparameter controlling the sparseness of $\boldsymbol{\beta}$. A larger S will cause shrinkage of more coefficient in $\boldsymbol{\beta}$ towards zero (Tibshirani 1996).

Note that LASSO weights all features equally. Later, we want to exclude a couple of parameters simultaneously, i. e. discarding one feature for our model which is not used in any metric. Therefore, we need a generalization of the standard LASSO, introduced next, that allows excluding jointly all coefficients from the model in predefined groups.

5.2.2 Group LASSO

Group (Yuan and Y. Lin 2006) aims for a selection of features which come in J predefined groups of sizes N_j, $j \in \{1, \ldots, J\}$, whereby either all or none of the features belonging to a group are chosen. For a linear model, the loss function becomes

$$\mathcal{L}(\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_J) = \frac{1}{2} \left\| \boldsymbol{y} - \sum_{j=1}^{J} \boldsymbol{X}_j \boldsymbol{\beta}_j \right\|_2^2 + S \cdot \sum_{j=1}^{J} \|\boldsymbol{\beta}_j\|_{\boldsymbol{K}_j} \tag{5.2}$$

where $\|\boldsymbol{z}\|_{\boldsymbol{K}_j} = (\boldsymbol{z}^\top \boldsymbol{K}_j \boldsymbol{z})^{0.5}$ and $\boldsymbol{K}_j$ are positive definite matrices, often $\boldsymbol{K}_j = N_j \mathbb{I}$ where N_j is the size of the jth group, and the usual design matrix $\boldsymbol{X}$ and covariate vector $\boldsymbol{\beta}$ is replaced by a collection of $\boldsymbol{X}_j$ and $\boldsymbol{\beta}_j$ for each group. Defining $\boldsymbol{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_J)$ and $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^\top, \boldsymbol{\beta}_2^\top, \ldots, \boldsymbol{\beta}_J^\top)^\top$ results in the Sum Squared Error (SSE), $\|y - \boldsymbol{X}\boldsymbol{\beta}\|_2^2$, which we will substitute to model's

cost function later. Setting the number of groups J to the number of dimensions results in the same solution as in Equation (5.1) because the regularization term $\sum_{j=1}^{J} \|\boldsymbol{\beta}_j\|_{\boldsymbol{K}_j}$ gets the ℓ_1 norm of $\boldsymbol{\beta}$ if each group consists only one element.

Since the groups will have the same size, i. e. $N_j = d$, we choose $\boldsymbol{K}_j$ as the identity matrix, which yields the simplification of the regularization term

$$S \cdot \sum_{j=1}^{J} \|\boldsymbol{\beta}_j\|_2 \tag{5.3}$$

where the Euclidean norm is used and the group size N_j is contained inside the sparsity parameter S.

As we will discuss later, LASSO will shrink some model parameters close to zero. However, the training is done with gradient descent and therefore the coefficients will not be exactly zero. That is why we do an efficient post-processing after training using OMP which we will briefly introduce next.

5.3 ORTHOGONAL MATCHING PURSUIT

The task is to find an approximation of a signal $\boldsymbol{f}$ in terms of a sparse linear combination of elements $\boldsymbol{g}$ from a given dictionary. More formally,

$$\boldsymbol{f} \approx \hat{\boldsymbol{f}}_N = \sum_{n=1}^{N} a_n \boldsymbol{g}_{J(n)} \tag{5.4}$$

where $\boldsymbol{g}_{J(n)}$ is the $J(n)$th element of the dictionary and $a_n \in \mathbb{R}$ the scaling factor. The number of chosen elements is parameterized by N. Since solving this problem exactly is NP-hard, S. Mallat and Z. Zhang (1993) proposed the *Matching Pursuit (MP)* that constitutes a greedy approximation where dictionary elements are iteratively chosen to minimize the reconstruction error by adding a single element and suitable scaling. *Orthogonal Matching Pursuit (OMP)* extends this scheme in such a way that the orthogonal projection of the signal onto the subspace spanned by the set of already selected base elements is calculated (Davis, S. G. Mallat, and Z. Zhang 1994). Classically, the dictionary is also trained or generated such that the reconstruction of the signal using only a few elements is guaranteed, e. g. with an overcomplete dictionary (Agarwal et al. 2014). As we do not need to design such a dictionary, we will not go deeper into this topic. How the dictionary is set for our aim to find sparse model parameters will be discussed at the end of the next section.

5.4 SPARSE LEARNING VECTOR QUANTIZATION

We will use a simplified regularization term derived from the Group inside the cost function of LGMLVQ. This term pushes values of the projection matrices (see Section 2.2.3) towards zero. Analog to Group where the model parameter of some groups are optimized, the values of the projection matrices for only a few features should influence the dissimilarity measure. All others

should be shrunk to zero and have no impact such that features associated with those can be disregarded for the classification task. As a recap, a (rectangular) projection matrix $\Omega_j \in \mathbb{R}^{m \times d}$ for a prototype $\boldsymbol{w}_j$, with $1 \leq m \leq d$ (see Section 2.2.4), forms the relevance matrix Λ_j in the parameterized dissimilarity. Hence, manipulating Ω_j does not affect the positive semi-definedness of Λ_j.

5.4.1 Sparse LGMLVQ

For local adaptive matrices Λ_j, the model does not use the feature f if and only if the column f equals $\mathbf{0}$ in all linear transformation matrices Ω_j. Hence, we can form groups of elements $[\Omega_\bullet]_{\bullet f}$ for a fix f, and we can penalize such groups by extending the cost function (see Equation (2.11)) as follows

$$E_{\text{SLGMLVQ}} = E_{\text{LGMLVQ}} + S \cdot \sum_{f=1}^{d} \left(\sum_{k=1}^{w} \sum_{i=1}^{m} \left([\Omega_k]_{if} \right)^2 \right)^{0.5} \tag{5.5}$$

where $\boldsymbol{\beta}_j$ from Equation (5.3) is replaced by a vector containing all entries in the fth column of all matrices. The gradient of the projection matrix Ω_j for $\boldsymbol{w}_j$ yields

$$\nabla_{\Omega_j} E_{\text{SLGMLVQ}} = \nabla_{\Omega_j} E_{\text{LGMLVQ}} + \boldsymbol{S} \tag{5.6}$$

with

$$[\boldsymbol{S}]_{rs} = \begin{cases} 0 & \text{for } [\Omega_j]_{rs} = 0 \\ S \cdot \dfrac{[\Omega_j]_{rs}}{\left(\sum\limits_{k=1}^{w} \sum\limits_{i=1}^{m} \left([\Omega_k]_{is} \right)^2 \right)^{0.5}} & \text{otherwise.} \end{cases} \tag{5.7}$$

This extension simultaneously shrinks all values in some columns of all projection matrices toward zero.

5.4.2 Reducing the Number of Dimensions

In practice, however, we do not necessarily observe values of zero exactly because the solver does not find such a solution, e. g. the cost function is optimized by a gradient descent which only shrinks values to zero. A simple way of tackling this is by defining a threshold such that all values smaller than the threshold are set to zero. This would lead to an additional hyperparameter and to a suitable choice of the threshold that might be problematic. A small value in the relevance matrix does not infer a less important feature since this feature could consist of bigger numbers.

Therefore, we add an efficient heuristic to choose the columns which contribute most and delete those which are equal to $\mathbf{0}$. For simplicity, we will concatenate all local matrices of an LGMLVQ model in the next step. Let $\Omega = [\Omega_1^\top, \ldots, \Omega_w^\top]^\top \in \mathbb{R}^{m' \times d}$ with $m' = mw$ be the concatenation of all single projection matrices Ω_j of each prototype.

The motivation is based on the observation that the model will not change if the linear projection induced by Ω is not altered on the data. The work by Schulz, Mokbel, et al. (2015) provides an exact characterization of the null

Table 5.1: Characteristics of all data sets used for Sparse LGMLVQ (SLGMLVQ)

Data set	# samples	# features	# classes	# samples per class
Fri	3000	50	2	1528 / 1472
Mot. Track.	10 299	561	6	1722 / 1544 / 1406 / 1777 / 1906 / 1944
Gisette	7000	5000	2	3500 / 3500

space of this mapping. Here, we aim for a substitution of Ω by a sparse matrix $\overline{\Omega}$ which keeps the data projection invariant, i. e. $\overline{\Omega}(\boldsymbol{x}_i - \boldsymbol{w}_j) \approx \Omega(\boldsymbol{x}_i - \boldsymbol{w}_j)$ for all samples $\boldsymbol{x}_i$ in the training set and $j \in \{1, \ldots, w\}$. The *sparsity* σ of $\overline{\Omega}$ is measured by the ratio of the number of columns containing only zeros to all features, more formally:

$$\sigma = 1 - \frac{1}{d} \left\| \boldsymbol{\sigma} \right\|_0, \text{ with } [\boldsymbol{\sigma}]_s = \sum_{i=1}^{m'} \left| [\overline{\Omega}]_{is} \right| \quad \forall s \in \{1, \ldots, d\} \tag{5.8}$$

where $\left\| \boldsymbol{\sigma} \right\|_0$ corresponds to the number of non-zero elements in $\boldsymbol{\sigma}$, i. e. the ℓ_0 norm.

Let $(\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_{m'})^\top = \Omega$ the m' rows of the projection matrix. Then, we are seeking a sparse $\overline{\boldsymbol{\eta}_i}$ with $\overline{\boldsymbol{\eta}_i}^\top \boldsymbol{X} \approx \boldsymbol{\eta}_i^\top \boldsymbol{X}$ where $\boldsymbol{X} = (\boldsymbol{x}_1 - \boldsymbol{w}_1, \ldots, \boldsymbol{x}_n - \boldsymbol{w}_1, \ldots, \boldsymbol{x}_1 - \boldsymbol{w}_w, \ldots, \boldsymbol{x}_n - \boldsymbol{w}_w)$ is the matrix containing all pairwise differences of all samples and prototypes. We can use OMP to approximate the problem $\boldsymbol{\eta}_i^\top \boldsymbol{X} =: y \approx \sum_{n=1}^{d} a_n [\boldsymbol{X}']_{\bullet n}$ where $[\boldsymbol{X}']_{\bullet n}$ is the normalized nth column of the data matrix $\boldsymbol{X}$. The normalization is required by OMP. From the (sparse) coefficients a_n and the normalization factors of each column, i. e. the variance of $[\boldsymbol{X}]_{\bullet n}$, we gain all $\overline{\boldsymbol{\eta}_i}$.

5.5 EXPERIMENTS

We evaluate the method on the Motion Tracking benchmark (see Section 2.3) and two additional ones. The first one, FRI, is generated by the toolbox by Göpfert, Pfannschmidt, and Hammer (2017) and the latter one, Gisette, is a more complex data set from a feature selection challenge (Guyon et al. 2004):

- *FRI* includes an artificially generated data set of 3000 points. Linearly separable data are enriched by ten redundant features (linear combinations of original ones) and 37 uniformly distributed irrelevant features.
- *Gisette* is a high-dimensional data set generated for a feature selection challenge. The classification task is to distinguish between the numbers 4 and 9.

In Table 5.1, the number of samples, classes, features, and samples per class for all three data sets are given. The number of features varies from 50 to 5000 and all classes are balanced. A feature-wise z-score is performed leading to a better comparison of the feature relevances.

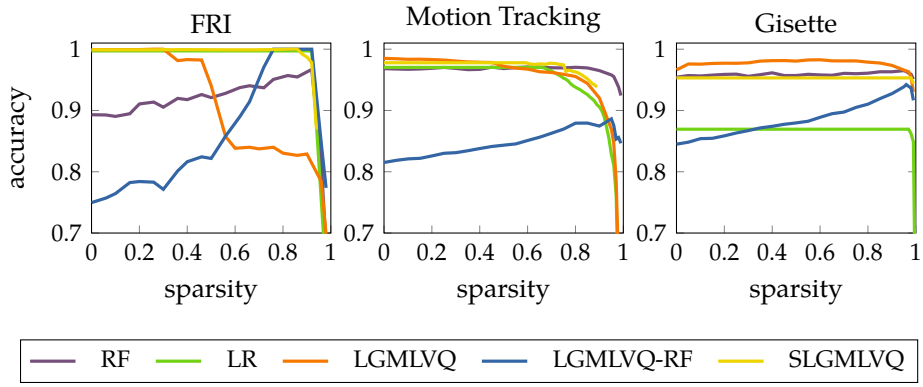


Figure 5.1: Averaged accuracy vs. sparsity of all three data sets and a fixed projection dimension $m = 5$

5.5.1 Evaluation scheme

We report the accuracy for all data sets out of a five-fold cross-validation and compare our approach with the following techniques (based on Scikit-learn (Pedregosa et al. 2011)) and the LVQ Matlab Toolbox by Bunte, Schneider, et al. (2012):

LR: We train various Logistic Regression (LR) models with LASSO and increasing weights of the ℓ_1-penalty, i. e. $C \in [1 \times 10^{-5}, 1]$[1].

RF: First, one Random Forrest (RF) is trained using all features that provide feature relevances. Second, we select only the most relevant features and train RF models for a decreasing number of those.

LGMLVQ: Analog to RF, we train one LGMLVQ model and obtain feature relevances by the diagonal of the relevance matrices. Then, for an increasing number of selected features, LGMLVQ models are trained using only the most relevant features.

LGMLVQ-RF: LGMLVQ models are trained using the same subsets of features that are used for RF.

SLGMLVQ: SLGMLVQ with Group penalty are trained, whereby the hyperparameter for the penalty is increased, i. e. $S \in [0, 10]$. Again, low-rank matrices for a various number of dimensions are used. Feature selection is accomplished by subsequently applying OMP.

5.5.2 Results

In Figure 5.1, the accuracy is plotted against the sparsity of all methods and each data set. In Table 5.2, accuracies and standard deviations are given for

1 In Scikit-learn the parameter C has to be positive and is the inverse of the regularization term such that smaller values specify stronger regularization.

Table 5.2: Mean and standard deviation (in parentheses) of the accuracy for all test sets and a fixed projection dimension $m = 5$

data	sparsity	RF	LR	LGMLVQ	LGMLVQ-RF	SLGMLVQ
FRI	0.5	0.921 (0.0168)	0.997 (0.0035)	0.940 (0.1320)	0.822 (0.0238)	0.999 (0.0015)
	0.75	0.947 (0.0045)	0.997 (0.0035)	0.840 (0.0763)	0.993 (0.0070)	0.999 (0.0009)
	0.9	0.963 (0.0080)	0.997 (0.0035)	0.829 (0.0866)	1.000 (0.0000)	0.987 (0.0241)
	0.94	0.891 (0.0125)	0.875 (0.0024)	0.799 (0.0709)	0.923 (0.0079)	0.869 (0.0765)
	0.98	0.677 (0.0065)	0.631 (0.0005)	0.698 (0.0195)	0.774 (0.0125)	|
Motion Tracking	0.5	0.968 (0.0023)	0.970 (0.0025)	0.972 (0.0035)	0.843 (0.0116)	0.978 (0.0036)
	0.75	0.969 (0.0021)	0.953 (0.0052)	0.959 (0.0026)	0.870 (0.0084)	0.974 (0.0025)
	0.9	0.965 (0.0033)	0.906 (0.0081)	0.920 (0.0083)	0.875 (0.0034)	0.938 (0.0045)
	0.94	0.959 (0.0020)	0.854 (0.0059)	0.880 (0.0108)	0.884 (0.0076)	|
	0.98	0.938 (0.0155)	0.594 (0.0311)	0.646 (0.0089)	0.856 (0.0331)	|
Gisette	0.5	0.957 (0.0052)	0.869 (0.0053)	0.981 (0.0037)	0.880 (0.0114)	0.953 (0.0086)
	0.75	0.960 (0.0058)	0.869 (0.0053)	0.981 (0.0044)	0.905 (0.0140)	0.953 (0.0086)
	0.9	0.963 (0.0083)	0.869 (0.0053)	0.974 (0.0066)	0.927 (0.0120)	0.953 (0.0086)
	0.94	0.965 (0.0059)	0.869 (0.0053)	0.968 (0.0062)	0.936 (0.0130)	0.953 (0.0086)
	0.98	0.959 (0.0058)	0.862 (0.0260)	0.954 (0.0072)	0.937 (0.0110)	0.953 (0.0086)

five sparsity values. Some methodologies do not enable a high degree of sparsity, indicated by a blank entry. Interestingly, for all data sets, the accuracy stays the same up to a sparsity of 0.9 and beyond. For RF, there are some cases where a higher sparsity improves the model's performance. In all cases, highly sparse models lead to high accuracy. At this point, strongly relevant features are discarded such that an accurate classification is prevented.

From these results, we can conclude that vanilla LGMLVQ does not yield sparse models via omitting features according to the relevance weighting, but both – explicit sparsity terms and prior feature selection by using RFs – enable sparse models. Surprisingly, RF features yield a good accuracy only for a small number of features for both RF itself and subsequent use of LGMLVQ models. OMP enables decent classification accuracy and reasonable sparsity, but in two cases, it does not permit an extreme sparsity of only 2 % of all features. A linear classifier is naturally restricted for non-linear classification tasks such as Motion Tracking and Gisette.

Further results for all data sets and different projection dimensionalities, i. e. $m \in \{2, 5, 10, 20\}$, can be found in Figures C.1 to C.3, where we plot the sparsity parameter S against accuracy and sparsity for our methodology in addition. We can make two main observations: First, a higher projection dimensionality does not yield better performance. Even if we set $m = 20$, comparable accuracies for LGMLVQ and SLGMLVQ are obtained. Second, the accuracy drops sharply at a definite value of S. This point differs for both the data set and the projection dimensionality. Due to an increasing number of rows of the projection matrices, we gain a less sparse model for larger m. Simultaneously, the impact of S is higher since the curves raise already for smaller values of the sparsity parameter S.

5.6 SUMMARY

In this chapter, we introduce an approach that provides sparse LGMLVQ models by interpreting the relevance matrices as projection matrices and searching for sparse approximations. By adding a penalty term to the cost

function, the coefficients of the projection matrices are shrunk toward zero. After a post-processing step, those coefficients reach zero such that they do not affect the dissimilarity and classification. These steps enhance the sparsity of the model (**RQ3**).

In the experiments, we show that the number of features can immensely be reduced in many real-world data sets. We observed that it is possible to remove more than 90 % of the features without deteriorating the model's performance. Such sparse models are particularly interesting for their efficient realization in hardware, in real-time classification scenarios, or on peripheral devices. Furthermore, a smaller model is easier to study and interpret.

6 FEDERATED LEARNING VECTOR QUANTIZATION

Overview: This chapter deals with our last research question **RQ4**: *Which adaptions to the LVQ training scheme are necessary to train a model using distributed data sets*? We will look at scenarios where the entire data is not accessible at once due to private, legal, strategic, or further concerns. Instead, we propose to rely on locally trained models and introduce a merging scheme of several LVQ models from each separated training set. We obtain a single model that generalizes better on the respective data sets by fusing those models. Further, it deals with scenarios of missing classes and thus, with training sets from different data distributions.

Publication: Parts of this chapter are based on the following publication:

- Brinkrolf, Johannes and Barbara Hammer (2021). "Federated Learning Vector Quantization". In: *29th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2021, Online event (Bruges, Belgium), October 6-8, 2021*. DOI: 10.14428/esann/2021.ES2021-141. URL: https://doi.org/10.14428/esann/2021.ES2021-141.

6.1 INTRODUCTION

In practice, a large amount of data is produced by distributed devices expedited by the Internet of Things (IoT) (Khan and Salah 2018). Collecting all this data eventually results in big data that can be vital in uncovering hidden patterns. Many decisions are directly made on peripheral devices for speed, economy, or security reasons. In this context, *Federated Learning (FL)* plays a central role. It is a learning scheme that enables an efficient, possibly privacy-preserving integration of the information of local data produced by individual persons towards an integrated model (Q. Yang et al. 2019). Again, similar to the last chapter, we will focus on LVQ extensions to metric learning schemes. We will rely on the fact that LVQ models naturally offer sufficient statistics of the model, a fact that has already been used in the context of drift-resistant incremental LVQ learning schemes for streaming data (Zhu, Schleif, and Hammer 2012). Based on this observation, we propose a method to fuse different LVQ models that enables distributed optimization. A specific focus is paid to the scenario of imbalanced classes.

In the next section, we will give a short overview of FL and explain how we gain sparse models out of pre-trained models on different disjoint training sets. In the experiments, we compare federated LVQ with the original version on three benchmarks and in two different scenarios including imbalanced classes. Note that we do not incorporate privacy as we already have discussed it in Chapter 3.

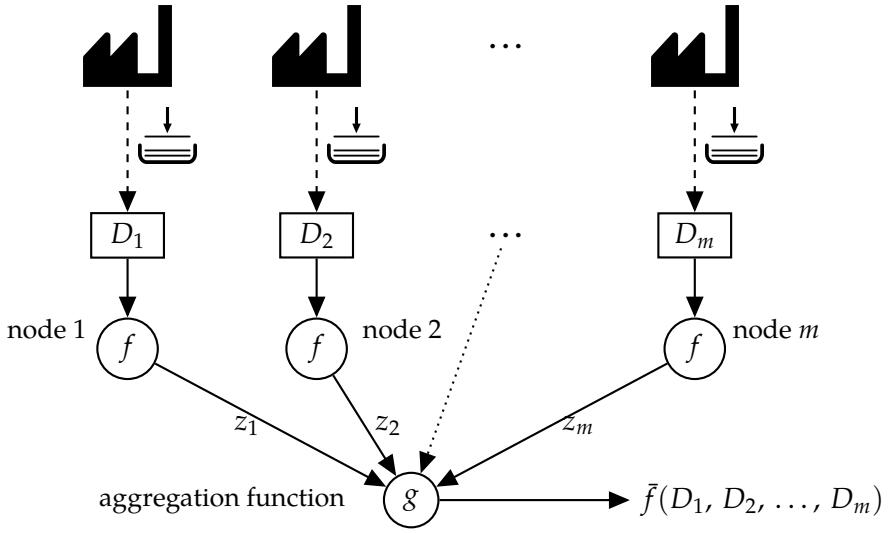


Figure 6.1: Visualization of Federated Learning. Given disjoint data sets that might have different data distributions for numerous nodes. The same function f, e. g. the training of an ML model, is first applied on each data set and node separately. Then, a fused outcome is observed by gathering all outcomes from all nodes and aggregating those.

6.2 FEDERATED LEARNING

Federated Learning aims for algorithmic realizations across multiple *decentralized* devices or servers with *distributed* data storage. Each server, in this context we will use the term *node* from now on, holds its own data samples. The task is to learn or/and improve an overarching model locally on every node without exchanging any data samples. In contrast to other techniques, such as more classical decentralized approaches, FL is not assuming identically distributed data samples on each node. Often, also the privacy of data is realized, but this is left out in this chapter as it is discussed in Chapter 3.

The general idea of it is depicted in Figure 6.1. On each node, the data is used as an input of a function or an algorithm f. Then, all outputs z_i from all nodes are transmitted and used as an input for the aggregation function g that results in the final output $\bar{f}$. Note that this scheme is very similar to the one already introduced in Section 3.4.2, but the data set is not divided into m smaller sets this time. Instead, the individual sets are predefined and may have (slightly) deviating distributions.

6.3 FEDERATED LEARNING VECTOR QUANTIZATION

For LVQ, FL can be based on the observation that these models can be described by sufficient statistics given by prototypes, metrics, and the size of each receptive field. This information enables a merge of models by suitably accumulating their parameters (Zhu, Schleif, and Hammer 2012). For FL, we assume that on each node one LVQ model is trained using only its local data samples. Further, we assume that the classes are consistent across all nodes, i. e. class labels are elements of a common finite set. The model on a node i

possesses a set of prototypes $\mathcal{W}_i$, where every class is represented by at most one prototype, and a projection matrix $\Omega^{(i)}$ in case of a global metric, or local projection matrices $\Omega_j^{(i)}$ for each prototype $\boldsymbol{w}_j \in \mathcal{W}_i$ for LGMLVQ. We obtain a new model by the following combination of the node information:

Prototypes: We calculate the new prototype's position for class k via

$$\boldsymbol{w}_k = \frac{1}{N_k} \sum_{i=1}^{m} \sum_{\substack{\boldsymbol{w} \in \mathcal{W}_i: \\ \mathrm{c}(\boldsymbol{w})=k}} \boldsymbol{w} \quad \forall\, k \in \{1, \ldots, c\} \tag{6.1}$$

where $N_k = |\{\boldsymbol{w} \mid \boldsymbol{w} \in \mathcal{W}_i : \mathrm{c}(\boldsymbol{w}) = k,\ i \in \{1, \ldots, m\}\}|$ is the number of prototypes belonging to class k in all trained models. In the case that the class is missing in the data set of some nodes, the number could be smaller than m. Note that this definition is consistent with the theoretical finding of stationary solutions of LVQ models as presented by Biehl, Hammer, Schleif, et al. (2015): it is shown that prototype $\boldsymbol{w}_j$ is given by the ratio of the expectations $\langle \Phi_j^i \boldsymbol{x}_i \rangle / \langle \Phi_j^i \rangle$ where Φ_j^i is ± 1 if $\boldsymbol{w}_j$ is winner and the label of $\boldsymbol{w}_j$ and $\boldsymbol{x}_i$ coincide/do not coincide, and 0 otherwise. Assuming similar values $\langle \Phi_j^i \rangle$ per node and prototype, a simple average of the prototypes approximates the complete solution.

Metric: The projection matrices Ω and Ω_j cannot be simply averaged because they are not unique, e. g. swapping two rows in Ω leads to the same solution. However, the relevance matrix itself is unique. Using N_k as above, we gain new relevance matrices Λ and Λ_k by

$$\Lambda = \frac{1}{m} \sum_{i=1}^{m} {\Omega^{(i)}}^\top \Omega^{(i)} \tag{6.2}$$

and

$$\Lambda_k = \frac{1}{N_k} \sum_{i=1}^{m} \sum_{\substack{\boldsymbol{w}_j \in \mathcal{W}_i: \\ \mathrm{c}(\boldsymbol{w}_j)=k}} {\Omega_j^{(i)}}^\top \Omega_j^{(i)} \quad \forall\, k \in \{1, \ldots, c\}. \tag{6.3}$$

Note that this choice does not directly correspond to the stationary solution of metric learning for the full data set as described by Biehl, Hammer, Schleif, et al. (2015). This would correspond to a matrix induced by principal components of a variation of a local covariance matrix that takes labeling into account. Hence, techniques similar to federated PCA or Singular Value Decomposition (SVD) would be required for exact solutions (Grammenos et al. 2020). Here, we take a different stance because taking the mean of the relevance matrices is equivalent to calculating the mean of all distances:

$$\begin{aligned} \frac{1}{m} \sum_{i=1}^{m} d_{\Lambda^{(i)}}(\boldsymbol{x}, \boldsymbol{w}) &= \frac{1}{m} \sum_{i=1}^{m} (\boldsymbol{x} - \boldsymbol{w})^\top \Lambda^{(i)} (\boldsymbol{x} - \boldsymbol{w}) \\ &= (\boldsymbol{x} - \boldsymbol{w})^\top \left(\frac{1}{m} \sum_{i=1}^{m} \Lambda^{(i)} \right) (\boldsymbol{x} - \boldsymbol{w}) \\ &= (\boldsymbol{x} - \boldsymbol{w})^\top \left(\frac{1}{m} \sum_{i=1}^{m} {\Omega^{(i)}}^\top \Omega^{(i)} \right) (\boldsymbol{x} - \boldsymbol{w}) \end{aligned} \tag{6.4}$$

and analogous for local matrices. A computationally efficient realization for the fused model relies on projection matrices Ω and Ω_j with $\Lambda = \Omega^\top \Omega$ and $\Lambda_j = \Omega_j^\top \Omega_j$. These exist since relevance matrices are symmetric and positive semi-definite by construction. Here, we use Schur's algorithm to calculate the square root $\Lambda^{1/2}$ of a relevance matrix by a Schur decomposition $\Lambda = \boldsymbol{Q}\boldsymbol{T}\boldsymbol{Q}^*$ where $\boldsymbol{T}$ is upper triangular and $\boldsymbol{Q}$ is unitary. We use an efficient algorithm proposed by Deadman, Higham, and Ralha (2012) where the calculation is blocked. It aims for the unique *principal square root* $\Omega = \Lambda^{1/2} = \boldsymbol{Q}\boldsymbol{T}^{1/2}\boldsymbol{Q}^*$ where all eigenvalues are non-negative. Thus, these square roots can be used directly as the projection matrices.

Additionally, as a side note and similar to the low-rank approximation described in Section 2.2.4, it is possible to use only the first k biggest eigenvectors from the principal square root. Hence, the projection matrices are rectangular, and all benefits of less memory consumption (Ω, $\Omega_j \in \mathbb{R}^{k \times d}$ instead of $\mathbb{R}^{d \times d}$) and less computational cost calculating all dissimilarities still hold.

6.4 EXPERIMENTS

We evaluate the method based on three benchmarks already introduced in Section 2.3. They are the *Image Segmentation* (Dua and Graff 2017), *Motion Tracking* (Anguita et al. 2013), and *Digit* (Dua and Graff 2017). These benchmarks are balanced but differ in the number of samples between 2310 and 10 299. All of them have at least 6 classes and are still quite small such that splitting the data into disjoint subsets for each node has the biggest impact.

We evaluate the methods by their performance in a five-fold cross-validation. For each training set, we use feature-wise z-score and for the latter two benchmarks a PCA projection into 30 dimension. The latter method reduces the complexity and yields better performing models as seen in Section 2.3.2. Both transforms are applied to the test sets in the evaluation. For federated LVQ, we train a separate model on a disjoint subset and obtain a final model by Equations (6.1) to (6.3). We compare our new method with a centralized version of GMLVQ/LGMLVQ using the toolbox by Bunte, Schneider, et al. (2012) and all training data. Two different scenarios are tested. For the first one, the training set is randomly split into five disjoint sets, one for each node. In the second, erratic subsets are generated by removing all samples belonging to class i for each node i.

In Table 6.1, we report the averaged class-wise F1-scores on the test sets and the mean values over all classes in the rows labeled *avg* besides their standard deviations. The results for the first scenario and for both LVQ variants are in the columns named *1st scenario* and *vanilla*. All scores of the federated versions are comparable to those of the centralized ones, except for three classes of the Image Segmentation and GMLVQ. All other differences are less than 0.015 and smaller than half of the standard deviation. The drops for the three classes, 3, 4, and 5 of Image Segmentation, are where also the vanilla version performs worse. Here, the number of training samples is an issue because we end up having only roughly 370 samples per node (slightly more than 50 per class). The local versions do not exhibit such behavior.

Table 6.1: Mean and standard deviation (in parentheses) of the class-wise F1-scores over a five-fold cross-validation for the first scenario in columns *1st scenario* and for the second one in columns *2nd scenario* where training samples belonging to some classes are removed

Data set	class	GMLVQ			LGMLVQ		
		1st scenario	2nd scenario	vanilla	1st scenario	2nd scenario	vanilla
Image Segmentation	1	0.908 (0.0234)	0.918 (0.0310)	0.901 (0.0282)	0.991 (0.0113)	0.986 (0.0075)	0.988 (0.0176)
	2	0.995 (0.0037)	0.989 (0.0185)	0.994 (0.0058)	1.000 (0.0000)	1.000 (0.0000)	1.000 (0.0000)
	3	0.843 (0.0270)	0.824 (0.0276)	0.888 (0.0184)	0.887 (0.0212)	0.882 (0.0355)	0.901 (0.0215)
	4	0.866 (0.0242)	0.820 (0.0088)	0.880 (0.0294)	0.914 (0.0485)	0.897 (0.0118)	0.917 (0.0452)
	5	0.684 (0.0615)	0.644 (0.0464)	0.753 (0.0145)	0.824 (0.0274)	0.806 (0.0230)	0.844 (0.0239)
	6	0.964 (0.0176)	0.921 (0.0205)	0.979 (0.0142)	0.978 (0.0047)	0.975 (0.0149)	0.992 (0.0067)
	7	0.992 (0.0096)	0.995 (0.0037)	0.992 (0.0049)	0.995 (0.0061)	0.995 (0.0037)	0.995 (0.0061)
	avg	0.893 (0.0239)	0.873 (0.0224)	0.913 (0.0165)	0.941 (0.0170)	0.935 (0.0138)	0.948 (0.0173)
Motion Tracking	1	0.957 (0.0048)	0.945 (0.0148)	0.956 (0.0057)	0.972 (0.0047)	0.970 (0.0054)	0.972 (0.0054)
	2	0.932 (0.0072)	0.915 (0.0173)	0.931 (0.0083)	0.953 (0.0071)	0.950 (0.0073)	0.951 (0.0047)
	3	0.936 (0.0107)	0.923 (0.0114)	0.934 (0.0107)	0.956 (0.0084)	0.954 (0.0054)	0.955 (0.0071)
	4	0.795 (0.0101)	0.805 (0.0124)	0.801 (0.0124)	0.855 (0.0050)	0.857 (0.0089)	0.856 (0.0061)
	5	0.859 (0.0045)	0.802 (0.0447)	0.861 (0.0034)	0.873 (0.0056)	0.875 (0.0060)	0.871 (0.0053)
	6	0.984 (0.0039)	0.992 (0.0047)	0.986 (0.0050)	0.997 (0.0018)	0.997 (0.0017)	0.997 (0.0022)
	avg	0.910 (0.0069)	0.897 (0.0175)	0.911 (0.0076)	0.934 (0.0054)	0.934 (0.0058)	0.934 (0.0051)
Digits	0	0.988 (0.0096)	0.990 (0.0018)	0.986 (0.0075)	0.993 (0.0036)	0.993 (0.0046)	0.993 (0.0022)
	1	0.884 (0.0109)	0.871 (0.0218)	0.888 (0.0145)	0.969 (0.0119)	0.975 (0.0078)	0.976 (0.0082)
	2	0.945 (0.0134)	0.935 (0.0156)	0.946 (0.0187)	0.986 (0.0065)	0.985 (0.0061)	0.988 (0.0061)
	3	0.959 (0.0176)	0.943 (0.0268)	0.944 (0.0201)	0.972 (0.0192)	0.973 (0.0120)	0.971 (0.0179)
	4	0.943 (0.0155)	0.937 (0.0249)	0.946 (0.0139)	0.980 (0.0078)	0.980 (0.0110)	0.982 (0.0069)
	5	0.953 (0.0129)	0.950 (0.0250)	0.950 (0.0133)	0.971 (0.0177)	0.973 (0.0136)	0.972 (0.0124)
	6	0.963 (0.0079)	0.955 (0.0100)	0.953 (0.0162)	0.987 (0.0058)	0.988 (0.0022)	0.988 (0.0058)
	7	0.972 (0.0059)	0.972 (0.0155)	0.973 (0.0089)	0.986 (0.0086)	0.987 (0.0116)	0.985 (0.0060)
	8	0.917 (0.0278)	0.911 (0.0149)	0.914 (0.0287)	0.951 (0.0208)	0.957 (0.0129)	0.963 (0.0174)
	9	0.915 (0.0247)	0.903 (0.0230)	0.908 (0.0225)	0.957 (0.0170)	0.960 (0.0080)	0.964 (0.0182)
	avg	0.944 (0.0146)	0.937 (0.0179)	0.941 (0.0164)	0.975 (0.0119)	0.977 (0.0090)	0.978 (0.0101)

For the second scenario, the averaged F1-scores are given in the columns *2nd scenario*. For the GMLVQ models, the scores drop slightly for the federated as well as for the vanilla version if, for the latter, the union of the training sets from all nodes is used that contains less data. The mean of the class-wise F1-scores drops down to 0.906 (Motion Tracking), 0.909 (Image Segmentation), and 0.935 (Digits), respectively. Again, this effect does not emerge when using local relevance matrices. If looking at the performances of each single trained model for each node in Table 6.2, one can see that the impact of missing classes plays a heavy role for only a few ones on these models. For example, by removing one of the classes 1, 2, or 3 in Motion Tracking, the scores of both other classes drop due to a lower precision and, therefore, a lower F1-score. The same holds for classes 4 and 5. Furthermore, for the other two data sets, some scores drop significantly, e. g. the scores drop by up to 0.15 for digit 8 if class 1 is absent and for digit 9 if class 3 is not represented in the training set. Comparing the class-wise F1-scores of our method with the best scores reached on one single node, we observe that the scores for the federated GMLVQ are comparable to the maximum obtainable score of all nodes. For LGMLVQ these scores are actually at least as high as the maximum.

6.5 SUMMARY

We have introduced an approach to obtain GMLVQ and LGMLVQ models from trained ones across multiple decentralized nodes for distributed opti-

Table 6.2: Mean of the class-wise F1-scores over a five-fold cross-validation for each node and class for the second scenario. Cells of missing classes are left blank.

Data set	class	GMLVQ						LGMLVQ					
		Node 1	Node 2	Node 3	Node 4	Node 5	max	Node 1	Node 2	Node 3	Node 4	Node 5	max
Image Segmentation	1		0.9546	0.9716	0.9295	0.9408	0.9716		0.9757	0.9787	0.8783	0.9818	0.9818
	2	0.9895		0.9809	0.9862	0.9910	0.9910	1.0000		1.0000	0.9248	1.0000	1.0000
	3	0.8579	0.7912		0.8662	0.7372	0.8662	0.8709	0.8087		0.8605	0.7395	0.8709
	4	0.6987	0.7150	0.8495		0.7569	0.8495	0.6049	0.6380	0.8388		0.7758	0.8388
	5	0.6521	0.7491	0.6414	0.7061		0.7491	0.8147	0.7446	0.6498	0.7043		0.8147
	6	0.9449	0.9547	0.9425	0.7347	0.9482	0.9547	0.9652	0.9723	0.9794	0.8973	0.9622	0.9794
	7	0.9728	0.8094	0.9909	0.9924	0.9474	0.9924	0.9880	0.9924	0.9939	0.9954	0.9827	0.9954
	avg	0.8526	0.8290	0.8961	0.8692	0.8869	0.8961	0.8739	0.8553	0.9068	0.8768	0.9070	0.9070
Motion Tracking	1		0.7578	0.8742	0.9507	0.9321	0.9507		0.8104	0.8786	0.9705	0.9590	0.9705
	2	0.6912		0.7196	0.9266	0.9093	0.9266	0.7082		0.7330	0.9451	0.9434	0.9451
	3	0.7850	0.7814		0.9385	0.9212	0.9385	0.8052	0.7741		0.9483	0.9510	0.9510
	4	0.7964	0.7855	0.8081		0.6454	0.8081	0.8439	0.8276	0.8448		0.6479	0.8448
	5	0.8493	0.8372	0.8597	0.7005		0.8597	0.8639	0.8604	0.8638	0.6924		0.8639
	6	0.9861	0.9813	0.9876	0.9642	0.9909	0.9909	0.9974	0.9974	0.9946	0.9826	0.9964	0.9974
	avg	0.8216	0.8286	0.8499	0.8961	0.8798	0.8961	0.8437	0.8540	0.8630	0.9078	0.8995	0.9078
Digits	0		0.9768	0.9864	0.9839	0.9504	0.9864		0.9919	0.9769	0.9928	0.9681	0.9928
	1	0.8660		0.7958	0.8845	0.8079	0.8845	0.9500		0.7921	0.9522	0.8750	0.9522
	2	0.9299	0.9286		0.8388	0.9443	0.9443	0.8538	0.9223		0.9501	0.9848	0.9848
	3	0.9356	0.9341	0.7851		0.9397	0.9397	0.9664	0.9511	0.8781		0.9540	0.9664
	4	0.9483	0.9339	0.9384	0.9460		0.9483	0.9617	0.8573	0.9719	0.9742		0.9742
	5	0.8542	0.9435	0.9318	0.8780	0.9347	0.9435	0.9312	0.9583	0.9696	0.8979	0.9523	0.9696
	6	0.8581	0.9349	0.9418	0.9735	0.7859	0.9735	0.9520	0.9717	0.9778	0.9849	0.8692	0.9849
	7	0.9688	0.9185	0.9579	0.9429	0.9403	0.9688	0.9789	0.9780	0.9503	0.9629	0.8977	0.9789
	8	0.8369	0.7266	0.8353	0.8596	0.9008	0.9008	0.9048	0.7807	0.8691	0.8922	0.9326	0.9326
	9	0.7753	0.8452	0.8743	0.7533	0.8697	0.8743	0.7681	0.8991	0.9089	0.7421	0.8614	0.9089
	avg	0.8859	0.9047	0.8941	0.8956	0.8971	0.9047	0.9185	0.9234	0.9216	0.9277	0.9217	0.9277

mization. Having scattered data sets, we can answer **RQ4** with this proposed method by merging the prototypes for each class separately and fusing the relevance matrices.

In our experiments, we showed that for the homogeneous scenario our models perform as well as those using all the training samples. In the second scenario, where classes are removed on some nodes, we show that the performances of the GMLVQ models are comparative. For LGMLVQ, the models are as good as those from the first scenario. Such models are particularly interesting for their efficient realization in hardware, on peripheral devices, or real-time classification models. Further, the methodology enables the training of a model on a decentralized server. It optimizes the models using other ones from different servers trained on new data samples without transmitting and sharing potentially sensitive data.

7 CONCLUSIONS AND OUTLOOK

In this dissertation, I investigated new technologies for prototype-based classifiers, more precisely for GLVQ and its adaptations to metric learning, i. e. GMLVQ and LGMLVQ. Although these models have been used for more than 30 years (Driancourt and Gallinari 1992; Makino et al. 1992; Morris, Rubin, and Tirri 1990) and a lot of research has been conducted, there are still open research questions that I address and investigate in this thesis. The questions introduced in Chapter 1 are summarized and answered in the following.

In Chapter 3, the *privacy* of a trained LVQ model is considered. Using the definition of Differential Privacy (DP) provides the opportunity to analyze whether a function or algorithm holds the notion of privacy. I showed that an LVQ model leaks information from training samples in all likelihood. Especially, the prototype's positions are affected by outliers or in settings where the number of prototypes does not match the inherent modality of the data distribution. I showed that the Global Sensitivity of LVQ is large since, in theory, the prototype's positions can vary heavily for adjacent training sets. Theoretically, one prototype might jump from one corner to the opposite one. Thus, privatizing the model by just perturbing the position of the prototypes is unfeasible. The prerequisite noise renders useless models because adding noise on the prototype to guarantee DP would shift all prototypes randomly far away.

Hence, I presented three novel training schemes for GLVQ models that preserve the privacy of each training sample according to the notion of DP. The statistical behavior of LVQ models and the geometrical property of the prototypes' positions are exploited for two variants. If various models are trained using distinct training sets, a differentially private GLVQ model is obtained by 1) averaging the prototypes belonging to the same class over all models and 2) perturbing the means with well-defined noise, i. e. adding random numbers drawn from the Laplace or Gaussian distribution. These techniques yield acceptable results. In particular, a simple aggregation seems better suited for high-dimensional data sets as a more advanced aggregation based on the COA. In contrast to the fusing approaches, I present a third variant that adapts the training itself, i. e. the initialization of the prototype is substituted by differentially private functions and the optimization by a differentially private mini-batch gradient descent. Here, the results are convincing, and the misclassification rates are not significantly worse than for the vanilla GLVQ already for small values for the privacy budget. Its sensitivity can be controlled concerning hyperparameters as well. In addition, the gradient-based technique enables the training of the metric parameters for GMLVQ that provides even better results.

Using these techniques, it is possible to preserve personal privacy in a published LVQ model (**RQ1**). Further, it can be guaranteed that those algorithms hold the definition of DP. The model does not leak more information from any sample used in the training algorithm, even with present auxiliary data.

In the next chapter, the *robustness* of an LVQ model has been considered. For the reject option given by the Relative Similarity, I proofed strict boundaries: The first one provides a minimal margin for samples close to the decision boundary. The second one gives the maximal distance between a sample and a model's prototype such that it is not rejected by Relative Similarity. The first limit directly gives the required minimum shift of a sample to change the classifier's output. Due to the rejects of points close to the decision boundary, this shift has to be at least twice the magnitude of the margin.

Further, I introduced a probabilistic post-processing that transfers outputs of pairwise LVQ models to a suitable scaling of probabilities. By coupling methods, these probabilities are converted into likelihoods for each class. In contrast to the threshold-based Relative Similarity that only reveals the certainty for the winner class, the rejection mechanism can directly consider these likelihoods that also have a semantic meaning. Furthermore, for a newly classified sample, the model provides greater insight into the classification since not only two classes, i. e. those from the closest prototypes belonging to different classes, are considered but all classes in the training set.

These probabilistic confidences can be combined over time in numerous settings. Examples are systems that monitor the status of an industrial machine, peripheral devices that record sensor data and classify the performed movement, or any other data set where the current class stays the same over a bunch of samples. For such settings, I gave a fusing method that improves the classification's performance by avoiding false predictions.

To summarize, the reject option ensures that small changes or distortions of the input sample do not yield a different output of the classifier. Likewise, the integration of class probabilities prevents misclassification. Thus, these methods increase the model's robustness. This positively answers **RQ2**.

Some settings require that the classification can be performed in real-time. In the two mentioned examples, an error should be detected as soon as possible and the performed movement should be recognized fast. Therefore, the classifier's *complexity* has to be small. In Chapter 5, I add a regularization term to the cost function that penalizes the metric parameters of a model. This penalization shrinks some model coefficients towards zero such that features belonging to those do not affect the dissimilarity and hence the classification. Using this regularization, a sparse model can be obtained and features of the samples can be omitted. Additionally, the models are faster in the predictions, easier to inspect, and easier to interpret for a human. This adaption leads to less complex models and a higher sparsity (**RQ3**).

The last proposed methodology in my thesis fuses parameters from several models trained on *distributed data sets* (**RQ4**). For GMLVQ and LGMLVQ models, I gave a scheme that combines the position of the prototypes and the metric parameters for the dissimilarities. It enables the training of a model from decentralized servers without transmitting and sharing potentially sensitive data.

All approaches enhance the GLVQ models and their variations concerning *interpretability*, *sparseness*, *robustness*, and *privacy* that can directly be used in various utilization. They pave the way for releasing trained models on sensitive data such as, for example, some from the biomedical domain or activity profiles. Likewise, the mechanisms can help to enable the classification

directly on a peripheral device with limited computational power where the data is recorded. Hence, processing the data can be performed on those devices without transferring a lot of data to a server beforehand. They can also deal with applications with restricted access to training or test data because of privacy issues, strategic concerns, or just the amount of data. Further, due to the sparser representation of the data, the models are simpler to interpret, and the predictions are easier to comprehend.

Outlook: However, some additional open questions are only partially mentioned in the previous chapters that I will motivate and raise in the following. As the privacy for all samples in the training set is preserved by adding random noise that typically goes beyond the amount of noise used to regularize ML models, a comparable big data set is required for appropriate results. Hence, one challenging question is how to mediate this problem in practical applications where less data or a rare population is present. Recent approaches propose mechanisms that enable a selection of how much potentially sensitive information can be revealed provided a higher accuracy is required, such as discussed, for example, by Ligett et al. (2017).

Further, on the tested benchmarks, one prototype per class provides already promising results, particularly if the metric is adapted. However, several prototypes per class or even varying amounts of prototypes for each type may be suitable for other data sets. This raises the question of how to fuse the model parameter of several models because relying on the means is not sufficient. Hence, a heuristic finding a convincing matching is necessary.

My last aspect is related to FL. This thesis scratches only the surface of the broad research topic. In contrast to our assumptions in Chapter 6, the local data sets can also differ in the contained features or are influenced by distinct drifts, i. e. samples are drawn from different distributions. The first results are already obtained by V. Vaquet, Hinder, Brinkrolf, et al. (2022) where the proposed LVQ version deals with concept drift and strongly imbalanced classes for each node.

GLOSSARY

adjacent Two data sets are adjacent if they differ in at most one sample, i. e. the Hamming Distance is 1 (see Definition 3 in Section 3.2) 26–30, 36, 42, 93, 121, 122

differentially private An algorithm or mechanism is called differentially private if it fulfills the Definition 4 25–29, 31–33, 35, 37, 40–46, 48, 52–54, 93, 100, 101, 121, 122, 126

edge computing is a computing paradigm where processing data takes place closer to the source of data, e. g. directly on a peripheral device 2, 77

F1-score is the harmonic mean of the precision and the recall 90–92, 101

Gaussian Mechanism is a differentially private mechanism which adds random variables drawn from the Gaussian distribution to ensure DP (see Section 3.2.4) 31, 37, 43, 44

Global Sensitivity gives the maximum difference of a function between the outputs of two adjacent data sets (see Definition 5) 29–31, 35, 39, 40, 43, 44, 49, 93

Group LASSO is a variation of Least Absolute Shrinkage and Selection Operator which allows predefiened groups of model parameters to jointly be selected into or out of a model (see Section 5.2.2) 77–80, 83

HT is a coupling method used to gain class probabilities (see Section 4.4.2) 66, 67, 73

hyperparameter is a parameter controlling the learning process of a ML algorithm 25, 26, 45–48, 54, 79, 81, 83, 93, 122–125

Laplace Mechanism is a differentially private mechanism which adds random variables drawn from the Laplace distribution to ensure DP (see Theorem 3 in Section 3.2) 29–31, 34, 37, 39, 41–43, 121

Least Absolute Shrinkage and Selection Operator is a method that regularize the coefficients of a ML model such that some coefficients are shrinked to zero (see Section 5.2.1) 79

limited-memory BFGS is a quasi-Newton optimization algortihm that approximates the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. It estimate the invers Hessian matrix by only a few vectors representing the approximation 11

Local Sensitivity gives the maximum difference of a function between a given data set and an adjacent one (see Definition 6 in Section 3.2) 26, 29, 30, 35, 40, 44, 45

peripheral device is an auxiliary deviced which is used to put information into or get information out of a computer; typicalle examples are, mouse, keyboard, printer, monitor, speaker but also smartwatch, wearables, or sensors iv, 68, 87, 92, 94, 95

PKPD is a coupling method used to gain class probabilities (see Section 4.4.2) 67, 70–73

Platt Scaling is a method which fits a sigmoid function to turn uncalibrated outputs into posterior probabilities (see Section 4.4.1) 65, 66
Principle Component Analysis is a linear mapping used for dimension reduction. The projection directions are defined as ones that minimizes the average squared distances from the samples to the line 20
privacy budget is a parameter which defined the privacy in a differentially private algorithm (see Definition 4 in Section 3.2.1) 28, 42, 43, 45–54, 93, 100, 122, 126, 128

Relative Similarity is a certainty measure for LVQ models (see Definition 9 in Section 4.2.2) 60, 64, 68, 94, 100

Sum Squared Error is a measurement of the discrepancy between the data and an estimation of a model. It sums up the sqared risiduals (difference between the true value and the estimation), i. e. SSE $= \|y - \hat{y}\|$ 79

winner-takes-all is a principle for prototype-based classifiers where the classification of a sample is determined by the closest prototype (see Equation (2.8)) 10, 12, 58
WLW1 is a coupling method used to gain class probabilities (see Section 4.4.2) 67, 73
WLW2 is a coupling method used to gain class probabilities (see Section 4.4.2) 67, 73

z-score trasforms each sample in a data set such that the mean of each feature equals zero and the variance equals one 70, 82, 90

ACRONYMS

ARC Accuracy-Reject-Curve 70–75, 100, 101

BFGS Broyden-Fletcher-Goldfarb-Shanno 49, 51, 127

CDF Cumulative Distribution Function 31, 32
COA Center of Attention 35–37, 44, 45, 48, 49, 51–54, 93, 100

DP Differential Privacy 23, 25–30, 32, 34–37, 39, 43, 44, 51, 53, 93, 100, 121
DT Decision Tree 25

FL Federated Learning 3, 87, 88, 95, 100

GLVQ Generalized Learning Vector Quantization iv, 7, 9, 11, 12, 14, 16–18, 20, 21, 23, 25, 31, 37–53, 56–60, 70, 78, 93, 94, 100, 122–125, 127, 128
GMLVQ Generalized Matrix Learning Vector Quantization 12–15, 17, 20, 21, 41–43, 45–49, 52, 57, 61, 63, 65, 69–73, 78, 90–94, 100, 122–125, 127

IoT Internet of Things 87

KM k-means 25

LASSO Least Absolute Shrinkage and Selection Operator 79, 80, 83
LBFGS limited-memory BFGS 11, 46, 48, 53, 73
LGMLVQ Local Generalized Matix Learning Vector Quantization 13–15, 17, 20, 21, 69–73, 77, 80, 81, 83, 84, 89–94
LR Logistic Regression 25, 83
LVQ Learning Vector Quantization 1–4, 7, 9–11, 16, 17, 23–26, 37, 38, 40, 41, 46, 49, 51–61, 64–69, 72, 74, 75, 77, 87–90, 93–95, 100, 101, 119, 120
LVQ1 Learning Vector Quantization1 10, 11, 16, 38, 44, 53

ML Machine Learning iv, 1, 2, 8, 9, 25, 29, 88, 95
MP Matching Pursuit 80

NB naïve Bayes 25
NG Neural Gas 1, 42

OMP Orthogonal Matching Pursuit 78, 80, 82–84

PCA Principle Component Analysis 20, 45, 89, 90
PDF Probability Density Function 8, 27, 28, 69

RelSim Relative Similarity 60–65, 68, 70, 71, 74, 75
RF Random Forrest 83, 84
RSLVQ Robust Soft Learning Vector Quantization 56, 64

SAA Subsample and Aggregate 34, 35, 37, 40, 41, 43–45, 48, 49, 51–53, 100
SGD Stochastic Gradient Descent 32, 37, 41, 43–49, 51–53, 100, 101, 126, 127
SLGMLVQ Sparse LGMLVQ 82–84, 129
SNG Supervised Neural Gas 16

SOM Self-Organizing Maps 1, 56
SSE Sum Squared Error 79
SVD Singular Value Decomposition 89
SVM Suport Vector Machine 25, 57, 64, 65, 74

LIST OF FIGURES

LIST OF TABLES

PUBLICATIONS IN THE CONTEXT OF THIS THESIS

Brinkrolf, Johannes, Tobias Mittag, et al. (2016). "Virtual optimisation for improved production planning". In: *Proceedings of the Workshop New Challenges in Neural Computation (NC^2 2016)*. Ed. by Barbara Hammer, Thomas Martinetz, and Thomas Villmann, pp. 67–74. URL: https://www.techfak.uni-bielefeld.de/~fschleif/mlr/mlr_04_2016.pdf#page=67.

Brinkrolf, Johannes, Kolja Berger, and Barbara Hammer (2017). "Differential Privacy for Learning Vector Quantization". In: *Proceedings of the Workshop New Challenges in Neural Computation (NC^2 2017)*. Ed. by Barbara Hammer, Thomas Martinetz, and Thomas Villmann, pp. 17–25. URL: https://www.techfak.uni-bielefeld.de/~fschleif/mlr/mlr_03_2017.pdf#page=17.

Brinkrolf, Johannes and Barbara Hammer (2017). "Probabilistic extension and reject options for pairwise LVQ". In: *12th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization, WSOM 2017, Nancy, France, June 28-30, 2017*, pp. 205–212. DOI: 10.1109/WSOM.2017.8020028. URL: https://doi.org/10.1109/WSOM.2017.8020028.

Schulz, Alexander, Johannes Brinkrolf, and Barbara Hammer (2017). "Efficient kernelisation of discriminative dimensionality reduction". In: *Neurocomputing* 268, pp. 34–41. DOI: 10.1016/j.neucom.2017.01.104. URL: https://doi.org/10.1016/j.neucom.2017.01.104.

Brinkrolf, Johannes, Kolja Berger, and Barbara Hammer (2018). "Differential private relevance learning". In: *26th European Symposium on Artificial Neural Networks, ESANN 2018, Bruges, Belgium, April 25-27, 2018*. URL: https://www.esann.org/sites/default/files/proceedings/legacy/es2018-119.pdf.

Brinkrolf, Johannes and Barbara Hammer (2018). "Interpretable machine learning with reject option". In: *at - Automatisierungstechnik* 66.4, pp. 283–290. DOI: 10.1515/auto-2017-0123. URL: https://doi.org/10.1515/auto-2017-0123.

Brinkrolf, Johannes, Christina Göpfert, and Barbara Hammer (2019). "Differential privacy for learning vector quantization". In: *Neurocomputing* 342, pp. 125–136. DOI: 10.1016/j.neucom.2018.11.095. URL: https://doi.org/10.1016/j.neucom.2018.11.095.

Brinkrolf, Johannes and Barbara Hammer (2020a). "Sparse Metric Learning in Prototype-based Classification". In: *28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2020, Bruges, Belgium, October 2-4, 2020*. Ed. by Michel Verleysen, pp. 375–380. URL: https://www.esann.org/sites/default/files/proceedings/2020/ES2020-138.pdf.

Brinkrolf, Johannes and Barbara Hammer (2020b). "Time integration and reject options for probabilistic output of pairwise LVQ". In: *Neural Computing and Applications* 32.24, pp. 18009–18022. DOI: 10.1007/s00521-018-03966-0. URL: https://doi.org/10.1007/s00521-018-03966-0.

Artelt, André, Valerie Vaquet, et al. (2021). "Evaluating Robustness of Counterfactual Explanations". In: *IEEE Symposium Series on Computational Intelligence, SSCI 2021, Orlando, FL, USA, December 5-7, 2021*, pp. 1–9. DOI: 10.1109/SSCI50451.2021.9660058. URL: https://doi.org/10.1109/SSCI50451.2021.9660058.

Brinkrolf, Johannes and Barbara Hammer (2021). "Federated Learning Vector Quantization". In: *29th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2021, Online event (Bruges, Belgium), October 6-8, 2021*. DOI: 10.14428/esann/2021.ES2021-141. URL: https://doi.org/10.14428/esann/2021.ES2021-141.

Hinder, Fabian, Johannes Brinkrolf, et al. (2021). "A Shape-Based Method for Concept Drift Detection and Signal Denoising". In: *IEEE Symposium Series on Computational Intelligence, SSCI 2021, Orlando, FL, USA, December 5-7, 2021*, pp. 1–8. DOI: 10.1109/SSCI50451.2021.9660111. URL: https://doi.org/10.1109/SSCI50451.2021.9660111.

Hinder, Fabian, Valerie Vaquet, et al. (2021). "Fast Non-Parametric Conditional Density Estimation using Moment Trees". In: *IEEE Symposium Series on Computational Intelligence, SSCI 2021, Orlando, FL, USA, December 5-7, 2021*, pp. 1–7. DOI: 10.1109/SSCI50451.2021.9660031. URL: https://doi.org/10.1109/SSCI50451.2021.9660031.

Vaquet, Valerie, Fabian Hinder, Jonas Vaquet, et al. (2021). "Online Learning on Non-Stationary Data Streams for Image Recognition using Deep Embeddings". In: *IEEE Symposium Series on Computational Intelligence, SSCI 2021, Orlando, FL, USA, December 5-7, 2021*, pp. 1–7. DOI: 10.1109/SSCI50451.2021.9659903. URL: https://doi.org/10.1109/SSCI50451.2021.9659903.

Artelt, André, Johannes Brinkrolf, et al. (2022). "Explaining Reject Options of Learning Vector Quantization Classifiers". In: *NCTA 2022 - Proceedings of the International Conference on Neural Computation Theory and Applications, part of the 14th International Joint Conference on Computational Intelligence IJCCI 2022*. event-place: Valletta, Malta. SciTePress.

Vaquet, Valerie, André Artelt, et al. (2022). "Taking care of our drinking water: Dealing with Sensor Faults in Water Distribution Networks". In: *31st International Conference on Artificial Neural Networks (ICANN 2022)*. event-place: Bristol, United Kingdom. URL: https://pub.uni-bielefeld.de/record/2962650.

Vaquet, Valerie, Fabian Hinder, Johannes Brinkrolf, et al. (2022). "Federated learning vector quantization for dealing with drift between nodes". In: *30th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2022)*. Ed. by Michel Verleysen. i6doc.com. URL: https://pub.uni-bielefeld.de/record/2964534.

REFERENCES

Abadi, Martín et al. (2016a). "Deep Learning with Differential Privacy". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS '16. event-place: Vienna, Austria. New York, NY, USA: Association for Computing Machinery, pp. 308–318. ISBN: 978-1-4503-4139-4. DOI: 10.1145/2976749.2978318. URL: https://doi.org/10.1145/2976749.2978318.

Abadi, Martín et al. (2016b). "Deep Learning with Differential Privacy". In: *CoRR* abs/1607.00133. arXiv: 1607.00133. URL: http://arxiv.org/abs/1607.00133.

Agarwal, Alekh et al. (2014). "Learning Sparsely Used Overcomplete Dictionaries". In: *Proceedings of The 27th Conference on Learning Theory, COLT 2014, Barcelona, Spain, June 13-15, 2014*. Ed. by Maria-Florina Balcan, Vitaly Feldman, and Csaba Szepesvári. Vol. 35. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 123–137. URL: http://proceedings.mlr.press/v35/agarwal14a.html.

Anguita, Davide et al. (2013). "A Public Domain Dataset for Human Activity Recognition using Smartphones". In: *21st European Symposium on Artificial Neural Networks, ESANN 2013, Bruges, Belgium, April 24-26, 2013*. URL: https://www.esann.org/sites/default/files/proceedings/legacy/es2013-84.pdf.

Armknecht, Frederik et al. (2015). "A Guide to Fully Homomorphic Encryption." In: *IACR Cryptology ePrint Archive* 2015, p. 1192. URL: http://dblp.uni-trier.de/db/journals/iacr/iacr2015.html#ArmknechtBCGJRS15.

Athalye, Anish et al. (2017). "Synthesizing Robust Adversarial Examples". In: *CoRR* abs/1707.07397. URL: http://arxiv.org/abs/1707.07397.

Bagnall, Anthony et al. (2017). "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances". In: *Data Min. Knowl. Discov.* 31.3. Number: 3, pp. 606–660. DOI: 10.1007/s10618-016-0483-9. URL: https://doi.org/10.1007/s10618-016-0483-9.

Balle, Borja and Yu-Xiang Wang (2018). "Improving the Gaussian Mechanism for Differential Privacy: Analytical Calibration and Optimal Denoising". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 403–412. URL: http://proceedings.mlr.press/v80/balle18a.html.

Beraha, Mario et al. (2019). "Feature Selection via Mutual Information: New Theoretical Insights". In: *International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019*, pp. 1–9. DOI: 10.1109/IJCNN.2019.8852410. URL: https://doi.org/10.1109/IJCNN.2019.8852410.

Bertolini, Massimo et al. (2021). "Machine Learning for industrial applications: A comprehensive literature review". In: *Expert Systems with Applications* 175, p. 114820. ISSN: 0957-4174. DOI:

https://doi.org/10.1016/j.eswa.2021.114820. URL: https://www.sciencedirect.com/science/article/pii/S095741742100261X.

Biehl, Michael (2017). "Biomedical Applications of Prototype Based Classifiers and Relevance Learning". In: *AlCoB: 4th Int. Conference on Algorithms for Computational Biology*. Ed. by Daniel Figueiredo et al. Cham: Springer International Publishing, pp. 3–23. ISBN: 978-3-319-58163-7.

Biehl, Michael, Kerstin Bunte, et al. (2012). "Large margin linear discriminative visualization by Matrix Relevance Learning". In: *The 2012 International Joint Conference on Neural Networks (IJCNN), Brisbane, Australia, June 10-15, 2012*, pp. 1–8. DOI: 10.1109/IJCNN.2012.6252627. URL: http://dx.doi.org/10.1109/IJCNN.2012.6252627.

Biehl, Michael, Anarta Ghosh, and Barbara Hammer (2007). "Dynamics and Generalization Ability of LVQ Algorithms". In: *Journal of Machine Learning Research* 8, pp. 323–360. URL: http://dl.acm.org/citation.cfm?id=1314511.

Biehl, Michael, Barbara Hammer, Frank-Michael Schleif, et al. (2015). "Stationarity of Matrix Relevance LVQ". In: *2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015*, pp. 1–8. DOI: 10.1109/IJCNN.2015.7280441. URL: https://doi.org/10.1109/IJCNN.2015.7280441.

Biehl, Michael, Barbara Hammer, Petra Schneider, et al. (2009). "Metric Learning for Prototype-Based Classification". In: *Innovations in Neural Information Paradigms and Applications*, pp. 183–199. DOI: 10.1007/978-3-642-04003-0_8. URL: https://doi.org/10.1007/978-3-642-04003-0_8.

Biehl, Michael, Barbara Hammer, and Thomas Villmann (2016). "Prototype-based models in machine learning". In: *Wiley interdisciplinary reviews. Cognitive science* 7.2. Number: 2, pp. 92–111. DOI: 10.1002/wcs.1378.

Bishop, Christopher M. (2007). *Pattern recognition and machine learning, 5th Edition*. Information science and statistics. Springer. ISBN: 978-0-387-31073-2. URL: https://www.worldcat.org/oclc/71008143.

Blum, Avrim and Ronald L. Rivest (1993). "Training a 3-Node Neural Network is NP-Complete". In: *Machine Learning: From Theory to Applications - Cooperative Research at Siemens and MIT*, pp. 9–28. DOI: 10.1007/3-540-56483-7_20. URL: https://doi.org/10.1007/3-540-56483-7_20.

Breiman, Leo (2001). "Random Forests". In: *Machine Learning* 45.1. Number: 1, pp. 5–32. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

Bunte, Kerstin, Elizabeth S. Baranowski, et al. (2016). "Relevance Learning Vector Quantization in Variable Dimensional Spaces". In: *New Challenges in Neural Computation (NC^2)*. Ed. by Barbara Hammer, Thomas Martinetz, and Thomas Villmann. Workshop of the GI-Fachgruppe Neuronale Netze and the German Neural Networks Society in connection to GCPR 2016. Hannover, Germany, pp. 20–23.

Bunte, Kerstin, Petra Schneider, et al. (2012). "Limited Rank Matrix Learning, discriminative dimension reduction and visualization". In:

Neural Networks 26, pp. 159–173. DOI: 10.1016/j.neunet.2011.10.001. URL: https://doi.org/10.1016/j.neunet.2011.10.001.

Carlini, Nicholas et al. (2019). "The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks". In: *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, pp. 267–284. URL: https://www.usenix.org/conference/usenixsecurity19/presentation/carlini.

Chow, C. K. (1970). "On optimum recognition error and reject tradeoff". In: *IEEE Transactions on Information Theory* 16.1. Number: 1, pp. 41–46. ISSN: 1557-9654. DOI: 10.1109/TIT.1970.1054406.

Çınar, Zeki Murat et al. (2020). "Machine Learning in Predictive Maintenance towards Sustainable Smart Manufacturing in Industry 4.0". In: *Sustainability* 12.19. ISSN: 2071-1050. DOI: 10.3390/su12198211. URL: https://www.mdpi.com/2071-1050/12/19/8211.

Crammer, Koby et al. (2002). "Margin Analysis of the LVQ Algorithm". In: *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pp. 462–469. URL: https://proceedings.neurips.cc/paper/2002/hash/bbaa9d6a1445eac881750bea6053f564-Abstract.html.

Davis, Geoffrey M., Stephane G. Mallat, and Zhifeng Zhang (1994). "Adaptive time-frequency decompositions". In: *Optical Engineering* 33.7. Number: 7, pp. 2183–2191. DOI: 10.1117/12.173207. URL: https://doi.org/10.1117/12.173207.

Deadman, Edvin, Nicholas J. Higham, and Rui Ralha (2012). "Blocked Schur Algorithms for Computing the Matrix Square Root". In: *Applied Parallel and Scientific Computing - 11th International Conference, PARA 2012, Helsinki, Finland, June 10-13, 2012, Revised Selected Papers*, pp. 171–182. DOI: 10.1007/978-3-642-36803-5_12. URL: https://doi.org/10.1007/978-3-642-36803-5_12.

Dowell, J. Richard (1977). "An Overview of Privacy and Security Requirements for Data Bases". In: *Proceedings of the 15th Annual Southeast Regional Conference*. ACM-SE 15. New York, NY, USA: ACM, pp. 528–536. DOI: 10.1145/1795396.1795469. URL: http://doi.acm.org/10.1145/1795396.1795469.

Driancourt, Xavier and Patrick Gallinari (1992). "A speech recognizer optimally combining learning vector quantization, dynamic programming and multi-layer perceptron". In: *1992 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '92, San Francisco, California, USA, March 23-26, 1992*, pp. 609–612. DOI: 10.1109/ICASSP.1992.225835. URL: https://doi.org/10.1109/ICASSP.1992.225835.

Du, Wenliang and Mikhail J. Atallah (2001). "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. event-place: Cloudcroft, New Mexico. New York, NY, USA: ACM, pp. 13–22. ISBN: 1-58113-457-6. DOI: 10.1145/508171.508174. URL: http://doi.acm.org/10.1145/508171.508174.

Dua, Dheeru and Casey Graff (2017). *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences. URL: http://archive.ics.uci.edu/ml.

Duch, Wlodzislaw (2006). "Filter Methods". In: *Feature Extraction - Foundations and Applications*, pp. 89–117. DOI: 10.1007/978-3-540-35488-8_4. URL: https://doi.org/10.1007/978-3-540-35488-8_4.

Duda, Richard O., Peter E. Hart, and David G. Stork (2001). *Pattern classification, 2nd Edition*. Wiley. ISBN: 978-0-471-05669-0. URL: https://www.worldcat.org/oclc/41347061.

Dwork, Cynthia (2006). "Differential Privacy". In: *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, pp. 1–12. DOI: 10.1007/11787006_1. URL: https://doi.org/10.1007/11787006_1.

Dwork, Cynthia (2011). "A firm foundation for private data analysis". In: *Commun. ACM* 54.1. Number: 1, pp. 86–95. DOI: 10.1145/1866739.1866758. URL: http://doi.acm.org/10.1145/1866739.1866758.

Dwork, Cynthia, Frank McSherry, et al. (2006). "Calibrating Noise to Sensitivity in Private Data Analysis". In: *Theory of Cryptography, Third Theory of Cryptography Conference*, pp. 265–284. DOI: 10.1007/11681878_14. URL: https://doi.org/10.1007/11681878_14.

Dwork, Cynthia and Aaron Roth (2014). "The Algorithmic Foundations of Differential Privacy". In: *Foundations and Trends in Theoretical Computer Science* 9.3. Number: 3-4, pp. 211–407. DOI: 10.1561/0400000042. URL: https://doi.org/10.1561/0400000042.

Dwork, Cynthia, Adam Smith, et al. (2017). "Exposed! A Survey of Attacks on Private Data". In: *Annual Review of Statistics and Its Application (2017)*, pp. 61–84. DOI: 10.1146/annurev-statistics-060116-054123. URL: https://doi.org/10.1146/annurev-statistics-060116-054123.

European Commission (Feb. 2020). *White Paper on Artificial Intelligence: a European approach to excellence and trust*. White Paper COM(2020) 65 final. Brussels: European Commission. URL: https://ec.europa.eu/info/files/white-paper-artificial-intelligence-european-approach-excellence-and-trust_en.

Ah-Fat, Patrick and Michael Huth (Mar. 2018). "Optimal Accuracy-Privacy Trade-Off for Secure Multi-Party Computations". In: *IEEE Transactions on Information Theory* PP. DOI: 10.1109/TIT.2018.2886458.

Fischer, Lydia, Barbara Hammer, and Heiko Wersing (2015). "Efficient rejection strategies for prototype-based classification". In: *Neurocomputing* 169, pp. 334–342. DOI: 10.1016/j.neucom.2014.10.092. URL: http://dx.doi.org/10.1016/j.neucom.2014.10.092.

Fischer, Lydia, Barbara Hammer, and Heiko Wersing (2016). "Optimal local rejection for classifiers". In: *Neurocomputing* 214, pp. 445–457. DOI: 10.1016/j.neucom.2016.06.038. URL: http://dx.doi.org/10.1016/j.neucom.2016.06.038.

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

Friedman, Jerome H. (1996). *Another approach to polychotomous classification*. Department of Statistics, Stanford University. URL: http://www-stat.stanford.edu/%20jhf/ftp/poly.ps.Z.

Frikken, Keith B. (2010). "Algorithms and Theory of Computation Handbook". In: ed. by Mikhail J. Atallah and Marina Blanton. Chapman & Hall/CRC, pp. 14–14. ISBN: 978-1-58488-820-8. URL: http://dl.acm.org/citation.cfm?id=1882723.1882737.

Gamelas Sousa, Ricardo et al. (2015). "Robust Classification with Reject Option Using the Self-organizing Map". In: *Neural Comput. Appl.* 26.7. Number: 7, pp. 1603–1619. ISSN: 0941-0643. DOI: 10.1007/s00521-015-1822-2. URL: http://dx.doi.org/10.1007/s00521-015-1822-2.

Gawlikowski, Jakob et al. (2021). "A Survey of Uncertainty in Deep Neural Networks". In: *CoRR* abs/2107.03342. arXiv: 2107.03342. URL: https://arxiv.org/abs/2107.03342.

Gisbrecht, Andrej et al. (2012). "Linear Time Relational Prototype Based Learning". In: *International Journal of Neural Systems* 22.5. Number: 5. DOI: 10.1142/S0129065712500219. URL: https://doi.org/10.1142/S0129065712500219.

Gong, Maoguo et al. (2020). "A Survey on Differentially Private Machine Learning [Review Article]". In: *IEEE Computational Intelligence Magazine* 15.2, pp. 49–64. DOI: 10.1109/MCI.2020.2976185. URL: https://doi.org/10.1109/MCI.2020.2976185.

Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy (2014). "Explaining and Harnessing Adversarial Examples". In: *ArXiv e-prints*.

Göpfert, Christina, Lukas Pfannschmidt, and Barbara Hammer (2017). "Feature Relevance Bounds for Linear Classification". In: *25th European Symposium on Artificial Neural Networks, ESANN 2017, Bruges, Belgium, April 26-28*. URL: http://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2017-67.pdf.

Grammenos, Andreas et al. (2020). "Federated Principal Component Analysis". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. URL: https://proceedings.neurips.cc/paper/2020/hash/47a658229eb2368a99f1d032c8848542-Abstract.html.

Guyon, Isabelle et al. (2004). "Result Analysis of the NIPS 2003 Feature Selection Challenge". In: *Advances in Neural Information Processing Systems*. Vol. 17.

Hammer, Barbara, Frank-Michael Schleif, and Thomas Villmann (2005). *On the generalization ability of prototype-based classifiers with local relevance determination*. Ifi technical report series. ISSN: 1860-8477. Clausthal-Zellerfeld: Institut für Informatik. URL: http://uri.gbv.de/document/gvk:ppn:515122092.

Hammer, Barbara, Frank-Michael Schleif, and Xibin Zhu (2011). "Relational Extensions of Learning Vector Quantization". In: *Neural Information Processing - 18th International Conference, ICONIP 2011, Shanghai, China, November 13-17, 2011, Proceedings, Part II*. Ed. by Bao-Liang Lu, Liqing Zhang, and James T. Kwok. Vol. 7063. Lecture Notes in Computer

Science. Springer, pp. 481–489. DOI: 10.1007/978-3-642-24958-7_56. URL: https://doi.org/10.1007/978-3-642-24958-7%5C_56.

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

Hammer, Barbara, Marc Strickert, and Thomas Villmann (2005). "Supervised Neural Gas with General Similarity Measure". In: *Neural Processing Letters* 21.1. Number: 1, pp. 21–44. DOI: 10.1007/s11063-004-3255-2. URL: https://doi.org/10.1007/s11063-004-3255-2.

Hammer, Barbara and Thomas Villmann (2002a). "Batch-RLVQ". In: *ESANN 2002, 10th Eurorean Symposium on Artificial Neural Networks, Bruges, Belgium, April 24-26, 2002, Proceedings*, pp. 295–300. ISBN: 2-930307-02-1. URL: https://www.esann.org/sites/default/files/proceedings/legacy/es2002-1.pdf.

Hammer, Barbara and Thomas Villmann (2002b). "Generalized Relevance Learning Vector Quantization". In: *Neural Networks* 15.8. Number: 8-9, pp. 1059–1068. DOI: 10.1016/S0893-6080(02)00079-5. URL: https://doi.org/10.1016/S0893-6080(02)00079-5.

Hastie, Trevor and Robert Tibshirani (1997). "Classification by Pairwise Coupling". In: *Advances in Neural Information Processing Systems 10, [NIPS Conference, Denver, Colorado, USA, 1997]*, pp. 507–513. URL: http://papers.nips.cc/paper/1375-classification-by-pairwise-coupling.

He, Xiaofei, Deng Cai, and Partha Niyogi (2005). "Laplacian Score for Feature Selection". In: *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*, pp. 507–514. URL: https://proceedings.neurips.cc/paper/2005/hash/b5b03f06271f8917685d14cea7c6c50a-Abstract.html.

Hendrickx, Kilian et al. (2021). "Machine Learning with a Reject Option: A survey". In: *CoRR* abs/2107.11277. arXiv: 2107.11277. URL: https://arxiv.org/abs/2107.11277.

Herbei, Radu and Marten H. Wegkamp (2006). "Classification with reject option". In: *Canadian Journal of Statistics* 34.4. Number: 4, pp. 709–721. DOI: 10.1002/cjs.5550340410.

Hitaj, Briland, Giuseppe Ateniese, and Fernando Pérez-Cruz (2017). "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*, pp. 603–618. DOI: 10.1145/3133956.3134012. URL: http://doi.acm.org/10.1145/3133956.3134012.

Höffgen, Klaus-Uwe, Hans Ulrich Simon, and Kevin S. Van Horn (1995). "Robust Trainability of Single Neurons". In: *Journal of Computer and System Sciences* 50.1. Number: 1, pp. 114–125. DOI: 10.1006/jcss.1995.1011. URL: https://doi.org/10.1006/jcss.1995.1011.

Hofmann, Daniela, Andrej Gisbrecht, and Barbara Hammer (2012). "Efficient Approximations of Kernel Robust Soft LVQ". In: *Advances in Self-Organizing Maps - 9th International Workshop, WSOM 2012, Santiago, Chile, December 12-14, 2012, Proceedings*. Ed. by Pablo A. Estévez, José Carlos Príncipe, and Pablo Zegers. Vol. 198. Advances in Intelligent Systems and Computing. Springer, pp. 183–192. DOI: 10.1007/978-3-642-35230-0_19. URL: https://doi.org/10.1007/978-3-642-35230-0%5C_19.

Hull, Jonathan J. (1994). "A Database for Handwritten Text Recognition Research". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16.5. Number: 5, pp. 550–554. DOI: 10.1109/34.291440. URL: http://dx.doi.org/10.1109/34.291440.

Ji, Zhanglong, Zachary Chase Lipton, and Charles Elkan (2014). "Differential Privacy and Machine Learning: a Survey and Review". In: *CoRR* abs/1412.7584. URL: http://arxiv.org/abs/1412.7584.

Jiménez, Javier González, Javier Gonzalez Monroy, and Jose-Luis Blanco (2011). "The Multi-Chamber Electronic Nose - An Improved Olfaction Sensor for Mobile Robotics". In: *Sensors* 11.6. Number: 6, pp. 6145–6164. DOI: 10.3390/s110606145. URL: http://dx.doi.org/10.3390/s110606145.

Jirayusakul, A. and Surapong Auwatanamongkol (2007). "A supervised growing neural gas algorithm for cluster analysis". In: *International Journal of Hybrid Intelligent Systems* 4.4, pp. 217–229. URL: http://content.iospress.com/articles/international-journal-of-hybrid-intelligent-systems/his00052.

Kaden, Marika et al. (2015). "Border-sensitive learning in generalized learning vector quantization: an alternative to support vector machines". In: *Soft Comput.* 19.9. Number: 9, pp. 2423–2434. DOI: 10.1007/s00500-014-1496-1. URL: https://doi.org/10.1007/s00500-014-1496-1.

Kairouz, Peter et al. (2021). "Advances and Open Problems in Federated Learning". In: *Found. Trends Mach. Learn.* 14.1, pp. 1–210. DOI: 10.1561/2200000083. URL: https://doi.org/10.1561/2200000083.

Katz, Jonathan and Yehuda Lindell (2007). *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC. ISBN: 1-58488-551-3.

Khan, Minhaj Ahmad and Khaled Salah (2018). "IoT security: Review, blockchain solutions, and open challenges". In: *Future Gener. Comput. Syst.* 82, pp. 395–411. DOI: 10.1016/j.future.2017.11.022. URL: https://doi.org/10.1016/j.future.2017.11.022.

Kohavi, Ron and George H. John (1997). "Wrappers for Feature Subset Selection". In: *Artif. Intell.* 97.1. Number: 1-2, pp. 273–324. DOI: 10.1016/S0004-3702(97)00043-X. URL: https://doi.org/10.1016/S0004-3702(97)00043-X.

Kohonen, Teuvo (1990). "Improved versions of learning vector quantization". In: *IJCNN 1990, International Joint Conference on Neural Networks, San Diego, CA, USA, June 17-21, 1990*. IEEE, pp. 545–550. DOI: 10.1109/IJCNN.1990.137622. URL: https://doi.org/10.1109/IJCNN.1990.137622.

Kohonen, Teuvo (1997). *Self-Organizing Maps, Second Edition*. Vol. 30. Springer Series in Information Sciences. Springer. ISBN: 978-3-540-62017-4. DOI: 10.1007/978-3-642-97966-8. URL: https://doi.org/10.1007/978-3-642-97966-8.

Lal, Thomas Navin et al. (2006). "Embedded Methods". In: *Feature Extraction - Foundations and Applications*, pp. 137–165. DOI: 10.1007/978-3-540-35488-8_6. URL: https://doi.org/10.1007/978-3-540-35488-8%5C_6.

Lecun, Yann et al. (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE*, pp. 2278–2324.

Ligett, Katrina et al. (2017). "Accuracy First: Selecting a Differential Privacy Level for Accuracy-Constrained ERM". In: *CoRR* abs/1705.10829. URL: http://arxiv.org/abs/1705.10829.

Liu, Fang (2019). "Generalized Gaussian Mechanism for Differential Privacy". In: *IEEE Transactions on Knowledge and Data Engineering* 31.4. Number: 4, pp. 747–756. DOI: 10.1109/TKDE.2018.2845388.

Losing, Viktor, Barbara Hammer, and Heiko Wersing (2018). "Incremental on-line learning: A review and comparison of state of the art algorithms". In: *Neurocomputing* 275, pp. 1261–1274. DOI: 10.1016/j.neucom.2017.06.084. URL: https://doi.org/10.1016/j.neucom.2017.06.084.

Makino, Shozo et al. (1992). "Continuous speech recognition with modified learning vector quantization algorithm and two-level DP-matching". In: *1992 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '92, San Francisco, California, USA, March 23-26, 1992*, pp. 597–600. DOI: 10.1109/ICASSP.1992.225838. URL: https://doi.org/10.1109/ICASSP.1992.225838.

Mallat, S.G. and Zhifeng Zhang (1993). "Matching pursuits with time-frequency dictionaries". In: *IEEE Transactions on Signal Processing* 41.12. Number: 12, pp. 3397–3415. DOI: 10.1109/78.258082.

Martins, Paulo, Leonel Sousa, and Artur Mariano (2017). "A Survey on Fully Homomorphic Encryption: An Engineering Perspective". In: *ACM Computing Surveys* 50.6. Number: 6, 83:1–83:33. ISSN: 0360-0300. DOI: 10.1145/3124441. URL: http://doi.acm.org/10.1145/3124441.

McSherry, Frank and Kunal Talwar (2007). "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*, pp. 94–103. DOI: 10.1109/FOCS.2007.41. URL: https://doi.org/10.1109/FOCS.2007.41.

Mori, Usue et al. (2017). "Reliable early classification of time series based on discriminating the classes over time". In: *Data Mining and Knowledge Discovery* 31.1. Number: 1, pp. 233–263. ISSN: 1573-756X. DOI: 10.1007/s10618-016-0462-1. URL: https://doi.org/10.1007/s10618-016-0462-1.

Morris, Robert J. T., Larry D. Rubin, and Henry Tirri (1990). "Neural Network Techniques for Object Orientation Detection: Solution by Optimal Feedforward Network and Learning Vector Quantization Approaches". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12.11. Number: 11, pp. 1107–1115. DOI: 10.1109/34.61712. URL: https://doi.org/10.1109/34.61712.

Nadeem, Malik Sajjad Ahmed, Jean-Daniel Zucker, and Blaise Hanczar (2010). "Accuracy-Rejection Curves (ARCs) for Comparing Classification Methods with a Reject Option". In: *Proceedings of the third International Workshop on Machine Learning in Systems Biology, MLSB 2009, Ljubljana, Slovenia, September 5-6, 2009*, pp. 65–81. URL: http://www.jmlr.org/proceedings/papers/v8/nadeem10a.html.

Narayanan, Arvind and Vitaly Shmatikov (2008). "Robust De-anonymization of Large Sparse Datasets". In: *2008 IEEE Symposium on Security and Privacy (S&P 2008), 18-21 May 2008, Oakland, California, USA*, pp. 111–125.

Nebel, David, Barbara Hammer, Kathleen Frohberg, et al. (2015). "Median variants of learning vector quantization for learning of dissimilarity data". In: *Neurocomputing* 169, pp. 295–305. DOI: 10.1016/j.neucom.2014.12.096. URL: https://doi.org/10.1016/j.neucom.2014.12.096.

Nebel, David, Barbara Hammer, and Thomas Villmann (2013). "A Median Variant of Generalized Learning Vector Quantization". In: *Neural Information Processing - 20th International Conference, ICONIP 2013, Daegu, Korea, November 3-7, 2013. Proceedings, Part II*. Ed. by Minho Lee et al. Vol. 8227. Lecture Notes in Computer Science. Springer, pp. 19–26. DOI: 10.1007/978-3-642-42042-9_3. URL: https://doi.org/10.1007/978-3-642-42042-9%5C_3.

Nene, Sameer A., Shree K. Nayar, and Hiroshi Murase (1996). *Columbia Object Image Library (COIL-20)*.

Nissim, Kobbi, Sofya Raskhodnikova, and Adam D. Smith (2007). "Smooth sensitivity and sampling in private data analysis". In: *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pp. 75–84. DOI: 10.1145/1250790.1250803. URL: http://doi.acm.org/10.1145/1250790.1250803.

Nova, David and Pablo A. Estévez (2014). "A review of learning vector quantization classifiers". In: *Neural Computing and Applications* 25.3. Number: 3-4, pp. 511–524. DOI: 10.1007/s00521-013-1535-3. URL: https://doi.org/10.1007/s00521-013-1535-3.

Paninski, Liam (2003). "Estimation of Entropy and Mutual Information". In: *Neural Comput.* 15.6. Number: 6, pp. 1191–1253. DOI: 10.1162/089976603321780272. URL: https://doi.org/10.1162/089976603321780272.

Pedregosa, Fabian et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Pekalska, Elzbieta and Robert P. W. Duin (2005). *The Dissimilarity Representation for Pattern Recognition - Foundations and Applications*. Vol. 64. Series in Machine Perception and Artificial Intelligence. WorldScientific. ISBN: 978-981-256-530-3. DOI: 10.1142/5965. URL: https://doi.org/10.1142/5965.

Pillai, Ignazio, Giorgio Fumera, and Fabio Roli (2013). "Multi-label classification with a reject option". In: *Pattern Recognition* 46.8. Number: 8, pp. 2256–2266. ISSN: 0031-3203. DOI: http://dx.doi.org/10.1016/j.patcog.2013.01.035. URL: http://www.sciencedirect.com/science/article/pii/S0031320313000745.

Platt, John C. (1999). "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods". In: *Advances in Large Margin Classifiers*. MIT Press, pp. 61–74.

Price, David et al. (1994). "Pairwise Neural Network Classifiers with Probabilistic Outputs". In: *Advances in Neural Information Processing Systems 7, [NIPS Conference, Denver, Colorado, USA, 1994]*, pp. 1109–1116. URL: http://papers.nips.cc/paper/883-pairwise-neural-network-classifiers-with-probabilistic-outputs.

Qin, A. Kai and Ponnuthurai N. Suganthan (2004). "A Novel Kernel Prototype-Based Learning Algorithm". In: *17th International Conference on Pattern Recognition, ICPR 2004, Cambridge, UK, August 23-26, 2004*. IEEE Computer Society, pp. 621–624. DOI: 10.1109/ICPR.2004.1333849. URL: https://doi.org/10.1109/ICPR.2004.1333849.

Qin, A. Kai, Ponnuthurai N. Suganthan, and Jing J. Liang (2004). "A new generalized LVQ algorithm via harmonic to minimum distance measure transition". In: *Proceedings of the IEEE International Conference on Systems, Man & Cybernetics: The Hague, Netherlands, 10-13 October 2004*. IEEE, pp. 4821–4825. DOI: 10.1109/ICSMC.2004.1401294. URL: https://doi.org/10.1109/ICSMC.2004.1401294.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin (2016). ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. event-place: San Francisco, California, USA. New York, NY, USA: ACM, pp. 1135–1144. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939778. URL: http://doi.acm.org/10.1145/2939672.2939778.

Riedel, Martin et al. (2013). "Regularization in relevance learning vector quantization using l1-norms". In: *21st European Symposium on Artificial Neural Networks, ESANN 2013, Bruges, Belgium, April 24-26, 2013*. URL: http://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2013-54.pdf.

Rodríguez-Fdez, Ismael et al. (2015). "STAC: a web platform for the comparison of algorithms using statistical tests". In: *Proceedings of the 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*.

Saralajew, Sascha, Lars Holdijk, and Thomas Villmann (2020). "Fast Adversarial Robustness Certification of Nearest Prototype Classifiers for Arbitrary Seminorms". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. URL: https://proceedings.neurips.cc/paper/2020/hash/9da187a7a191431db943a9a5a6fec6f4-Abstract.html.

Saralajew, Sascha and Thomas Villmann (2016). "Adaptive tangent distances in generalized learning vector quantization for transformation and distortion invariant classification learning". In: *2016 International Joint Conference on Neural Networks, IJCNN 2016, Vancouver, BC, Canada, July 24-29, 2016*, pp. 2672–2679. DOI: 10.1109/IJCNN.2016.7727534. URL: https://doi.org/10.1109/IJCNN.2016.7727534.

Sato, Atsushi and Keiji Yamada (1995). "Generalized Learning Vector Quantization". In: *Advances in Neural Information Processing Systems 8, NIPS, Denver, CO, November 27-30, 1995*, pp. 423–429. URL:

http://papers.nips.cc/paper/1113-generalized-learning-vector-quantization.

Schleif, Frank-Michael et al. (2011). "Efficient Kernelized Prototype Based Classification". In: *International Journal of Neural Systems* 21.6. Number: 6, pp. 443–457. DOI: 10.1142/S012906571100295X. URL: https://doi.org/10.1142/S012906571100295X.

Schneider, Petra, Michael Biehl, and Barbara Hammer (2009a). "Adaptive Relevance Matrices in Learning Vector Quantization". In: *Neural Computation* 21.12. Number: 12, pp. 3532–3561. DOI: 10.1162/neco.2009.11-08-908. URL: http://dx.doi.org/10.1162/neco.2009.11-08-908.

Schneider, Petra, Michael Biehl, and Barbara Hammer (2009b). "Distance Learning in Discriminative Vector Quantization". In: *Neural Comput.* 21.10, pp. 2942–2969. DOI: 10.1162/neco.2009.10-08-892. URL: https://doi.org/10.1162/neco.2009.10-08-892.

Schneider, Petra, Michael Biehl, and Barbara Hammer (2010). "Hyperparameter learning in probabilistic prototype-based models". In: *Neurocomputing* 73, pp. 1117–1124. DOI: 10.1016/j.neucom.2009.11.021.

Schulz, Alexander, Bassam Mokbel, et al. (2015). "Inferring Feature Relevances From Metric Learning". In: *IEEE Symposium Series on Computational Intelligence, SSCI 2015, Cape Town, South Africa, December 7-10, 2015*, pp. 1599–1606. DOI: 10.1109/SSCI.2015.225. URL: http://dx.doi.org/10.1109/SSCI.2015.225.

Seo, Sambu and Klaus Obermayer (2003). "Soft Learning Vector Quantization". In: *Neural Computation* 15.7. Number: 7, pp. 1589–1604. DOI: 10.1162/089976603321891819. URL: https://doi.org/10.1162/089976603321891819.

Simon, Noah et al. (2013). "A Sparse-Group Lasso". In: *Journal of Computational and Graphical Statistics* 22.2. Number: 2, pp. 231–245. DOI: 10.1080/10618600.2012.681250. URL: https://doi.org/10.1080/10618600.2012.681250.

Snell, Jake, Kevin Swersky, and Richard S. Zemel (2017). "Prototypical Networks for Few-shot Learning". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 4077–4087. URL: http://papers.nips.cc/paper/6996-prototypical-networks-for-few-shot-learning.

Su, Jiawei, Danilo Vasconcellos Vargas, and Kouichi Sakurai (2019). "One Pixel Attack for Fooling Deep Neural Networks". In: *IEEE Transactions on Evolutionary Computation* 23.5. Number: 5, pp. 828–841. ISSN: 1941-0026. DOI: 10.1109/tevc.2019.2890858. URL: http://dx.doi.org/10.1109/TEVC.2019.2890858.

Tibshirani, Robert (1996). "Regression Shrinkage and Selection Via the Lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1. Number: 1, pp. 267–288. DOI: https://doi.org/10.1111/j.2517-6161.1996.tb02080.x. URL: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1996.tb02080.x.

Torkkola, Kari and William M. Campbell (2000). "Mutual Information in Learning Feature Transformations". In: *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*. Ed. by Pat Langley. Morgan Kaufmann, pp. 1015–1022.

Varshney, Kush R. and Homa Alemzadeh (2016). "On the Safety of Machine Learning: Cyber-Physical Systems, Decision Sciences, and Data Products". In: *CoRR* abs/1610.01256. URL: http://arxiv.org/abs/1610.01256.

Villmann, Thomas and Sven Haase (2011). "Divergence-Based Vector Quantization". In: *Neural Comput.* 23.5, pp. 1343–1392. DOI: 10.1162/NECO_a_00110. URL: https://doi.org/10.1162/NECO%5C_a%5C_00110.

Villmann, Thomas, Barbara Hammer, et al. (2008). "Fuzzy classification using information theoretic learning vector quantization". In: *Neurocomputing* 71.16, pp. 3070–3076. DOI: 10.1016/j.neucom.2008.04.048. URL: https://doi.org/10.1016/j.neucom.2008.04.048.

Villmann, Thomas, Marika Kaden, et al. (2016). "Self-Adjusting Reject Options in Prototype Based Classification". In: *Advances in Self-Organizing Maps and Learning Vector Quantization - Proceedings of the 11th International Workshop WSOM 2016, Houston, Texas, USA, January 6-8, 2016*, pp. 269–279. DOI: 10.1007/978-3-319-28518-4_24. URL: http://dx.doi.org/10.1007/978-3-319-28518-4_24.

Vorácek, Václav and Matthias Hein (2022). "Provably Adversarially Robust Nearest Prototype Classifiers". In: *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, pp. 22361–22383. URL: https://proceedings.mlr.press/v162/voracek22a.html.

Wu, Ting-Fan, Chih-Jen Lin, and Ruby C. Weng (2004). "Probability Estimates for Multi-class Classification by Pairwise Coupling". In: *Journal of Machine Learning Research* 5, pp. 975–1005. URL: http://www.ai.mit.edu/projects/jmlr/papers/volume5/wu04a/wu04a.pdf.

Xu, Jiawen et al. (2022). "A Review on AI for Smart Manufacturing: Deep Learning Challenges and Solutions". In: *Applied Sciences* 12.16. ISSN: 2076-3417. DOI: 10.3390/app12168239. URL: https://www.mdpi.com/2076-3417/12/16/8239.

Yang, Hong-Ming et al. (2018). "Robust Classification With Convolutional Prototype Learning". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 3474–3482. DOI: 10.1109/CVPR.2018.00366. URL: http://openaccess.thecvf.com/content_cvpr_2018/html/Yang_Robust_Classification_With_CVPR_2018_paper.html.

Yang, Qiang et al. (2019). "Federated Machine Learning: Concept and Applications". In: *ACM Trans. Intell. Syst. Technol.* 10.2. Number: 2, 12:1–12:19. DOI: 10.1145/3298981. URL: https://doi.org/10.1145/3298981.

Yuan, Ming and Yi Lin (2006). "Model Selection and Estimation in Regression With Grouped Variables". In: *Journal of the Royal Statistical Society Series B* 68, pp. 49–67. DOI: 10.1111/j.1467-9868.2005.00532.x.

Yuan, Ming and Marten H. Wegkamp (2010). "Classification Methods with Reject Option Based on Convex Risk Minimization". In: *Journal of Machine Learning Research* 11, pp. 111–130. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1756006.1756011.

Zhang, Jun et al. (2013). "PrivGene: differentially private model fitting using genetic algorithms". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, pp. 665–676. DOI: 10.1145/2463676.2465330. URL: http://doi.acm.org/10.1145/2463676.2465330.

Zhu, Xibin, Frank-Michael Schleif, and Barbara Hammer (2012). "Patch Processing for Relational Learning Vector Quantization". In: *Advances in Neural Networks - ISNN 2012 - 9th International Symposium on Neural Networks, Shenyang, China, July 11-14, 2012. Proceedings, Part I*, pp. 55–63. DOI: 10.1007/978-3-642-31346-2_7. URL: https://doi.org/10.1007/978-3-642-31346-2%5C_7.

APPENDIX — TAXONOMY OF LVQ CLASSIFIERS

Table A.1: Full list of all LVQ classifiers from Figure 2.2 including their properties and references.

Method	Characteristics	Distance	Reference
AKGLVQ	Margin maximization, kernelized	Euclidean distance in feature space	Schleif et al. (2011)
ALVQ	Margin maximization	Angle-based dissimilarity	Bunte, Baranowski, et al. (2016)
GLVQ	Margin maximization	squared Euclidean distance	Sato and Yamada (1995)
GMLVQ	Margin maximization	squared generalized distance	Schneider, Biehl, and Hammer (2009a)
GRLVQ	Margin maximization	scaled squared Euclidean distance	Hammer, Strickert, and Villmann (2004)
GTLVQ	Margin maximization	Tangent Distance	Saralajew and Villmann (2016)
IT-LVQ	Margin maximization	Information Theoretic Learning	Torkkola and Campbell (2000)
H2MLVQ	Margin maximization	Harmonic average distances	Qin, Suganthan, and Liang (2004)
KGLVQ	Margin maximization, kernelized	Euclidean distance in feature space	Qin and Suganthan (2004)
KRSLVQ	likelihood-ratio maximization, kernelized	Euclidean distance in feature space	Hofmann, Gisbrecht, and Hammer (2012)
LGMLVQ	Margin maximization	squared generalized distance	Schneider, Biehl, and Hammer (2009a)
LGRLVQ	Margin maximization	scaled squared Euclidean distance	Hammer, Strickert, and Villmann (2004)
LMRSLVQ	likelihood-ratio maximization	squared generalized distance	Schneider, Biehl, and Hammer (2009b)
LVQ1	Heuristic	Euclidean distance	Kohonen (1990)
LVQ2.1	Heuristic	Euclidean distance	Kohonen (1990)
LVQ3	Heuristic	Euclidean distance	Kohonen (1990)
LVQ-CSD	Margin maximization	information theoretic learning	Villmann and Haase (2011)
MGLVQ	Margin maximization	Euclidean distance	Nebel, Hammer, and Villmann (2013)
MRSLVQ	likelihood-ratio maximization	squared generalized distance	Schneider, Biehl, and Hammer (2009b)

Table A.1: Continue list of all LVQ classifiers from Figure 2.2

Method	Characteristics	Distance	Reference
OLVQ	Heuristic	Euclidean distance	Kohonen (1990)
RGLVQ	Margin maximization	Dissimilarities	Gisbrecht et al. (2012)
RMGLVQ	Margin maximization	Dissimilarities	Nebel, Hammer, Frohberg, et al. (2015)
RRSLVQ	likelihood-ratio maximization	Dissimilarities	Hammer, Schleif, and Zhu (2011)
RSLVQ	likelihood-ratio maximization	Euclidean distance	Seo and Obermayer (2003)
SGNG	Margin maximization	Euclidean distance	Jirayusakul and Auwatanamongkol (2007)
SNG	Margin maximization	Euclidean distance	Hammer, Strickert, and Villmann (2005)

APPENDIX — DIFFERENTIAL PRIVACY

B

B.1 PROOFS

In this section, we will restate and prove Theorem 1 and provide a proof that the Laplace Mechanism is differentially private.

B.1.1 Composition Theorem

We restate Theorem 1 and provide a proof adapted from Dwork and Roth (2014, Theorem 3.14) to our notation.

Theorem 1. *Let $\mathcal{A}_1 \colon \mathcal{D} \to \mathbb{R}^k$ and $\mathcal{A}_2 \colon \mathcal{D} \to \mathbb{R}^{k'}$ be an ε_1- and ε_2-differentially private algorithms. Then, their combination, defined to be $\mathcal{A}_{1,2} \colon \mathcal{D} \to \mathbb{R}^k \times \mathbb{R}^{k'}$ by the mapping: $\mathcal{A}_{1,2}(\boldsymbol{x}) = (\mathcal{A}_1(\boldsymbol{x}), \mathcal{A}_2(\boldsymbol{x}))$ is $(\varepsilon_1 + \varepsilon_2)$-differentially private.*

Proof. Let $D, D' \subset \mathcal{D}$ be adjacent data sets. Fix any $(\boldsymbol{r}_1, \boldsymbol{r}_2) \in \mathbb{R}^k \times \mathbb{R}^{k'}$. Then:

$$\begin{aligned}
\frac{\mathbb{P}[\mathcal{A}_{1,2}(D) = (\boldsymbol{r}_1, \boldsymbol{r}_2)]}{\mathbb{P}[\mathcal{A}_{1,2}(D') = (\boldsymbol{r}_1, \boldsymbol{r}_2)]} &= \frac{\mathbb{P}[\mathcal{A}_1(D) = \boldsymbol{r}_1]\,\mathbb{P}[\mathcal{A}_2(D) = \boldsymbol{r}_2]}{\mathbb{P}[\mathcal{A}_1(D') = \boldsymbol{r}_1]\,\mathbb{P}[\mathcal{A}_2(D') = \boldsymbol{r}_2]} \\
&= \left(\frac{\mathbb{P}[\mathcal{A}_1(D) = \boldsymbol{r}_1]}{\mathbb{P}[\mathcal{A}_1(D') = \boldsymbol{r}_1]}\right)\left(\frac{\mathbb{P}[\mathcal{A}_2(D) = \boldsymbol{r}_2]}{\mathbb{P}[\mathcal{A}_2(D') = \boldsymbol{r}_2]}\right) \\
&\leq \exp(\varepsilon_1)\exp(\varepsilon_2) \\
&= \exp(\varepsilon_1 + \varepsilon_2)
\end{aligned}$$

By symmetry, $\mathbb{P}[\mathcal{A}_{1,2}(D)=(\boldsymbol{r}_1,\boldsymbol{r}_2)]/\mathbb{P}[\mathcal{A}_{1,2}(D')=(\boldsymbol{r}_1,\boldsymbol{r}_2)] \geq \exp(-(\varepsilon_1 + \varepsilon_2))$. □

B.1.2 Laplace Mechanism

We give the theorem for the Laplace Mechanism (Definition 7) and prove that it holds the definition of DP. This proof is adapted from Dwork and Roth (2014, Theorem 3.6) to our notation.

Theorem 6. *For $f \colon \mathcal{D} \to \mathbb{R}^k$, the Laplace Mechanism $\mathcal{A}_f$ gives $(\Delta_{\mathrm{GS}}(f)/\beta)$-Differential Privacy.*

Proof. Let $D, D' \subset \mathcal{D}$ be adjacent data sets, and f be some function $f \colon \mathcal{D} \to \mathbb{R}^k$. Let p_D and $p_{D'}$ denote the probability density functions of $\mathcal{A}_f(D)$ and

$\mathcal{A}_f(D')$, respectively. We compare the two at some arbitrary point $\boldsymbol{r} \in \mathbb{R}^k$:

$$\begin{aligned}
\frac{p_D(\boldsymbol{r})}{p_{D'}(\boldsymbol{r})} &= \prod_{i=1}^{k} \left(\frac{\exp\left(-\frac{\varepsilon|f(D)_i - \boldsymbol{r}_i|}{\Delta_{\mathrm{GS}}(f)}\right)}{\exp\left(-\frac{\varepsilon|f(D')_i - \boldsymbol{r}_i|}{\Delta_{\mathrm{GS}}(f)}\right)} \right) \\
&= \prod_{i=1}^{k} \exp\left(\frac{\varepsilon(|f(D')_i - \boldsymbol{r}_i| - |f(D)_i - \boldsymbol{r}_i|)}{\Delta_{\mathrm{GS}}(f)} \right) \\
&\leq \prod_{i=1}^{k} \exp\left(\frac{\varepsilon(|f(D)_i - f(D')_i|)}{\Delta_{\mathrm{GS}}(f)} \right) \\
&= \exp\left(\frac{\varepsilon \, \|f(D) - f(D')\|_1}{\Delta_{\mathrm{GS}}(f)} \right) \\
&\leq \exp(\varepsilon)
\end{aligned}$$

where the first inequality follows from the triangle inequality, and the second one follows from the definition of the sensitivity and the fact that D and D' are adjacent. That $p_D(\boldsymbol{r})/p_{D'}(\boldsymbol{r}) \geq \exp(-\varepsilon)$ follows by symmetry. □

B.2 FURTHER RESULTS

We will give further results of our differentially private GLVQ and GMLVQ versions for varying hyperparameters and selected privacy budgets for the three benchmarks MNIST, Motion Tracking, and Image Segmentation in Figures B.1 to B.3, respectively.

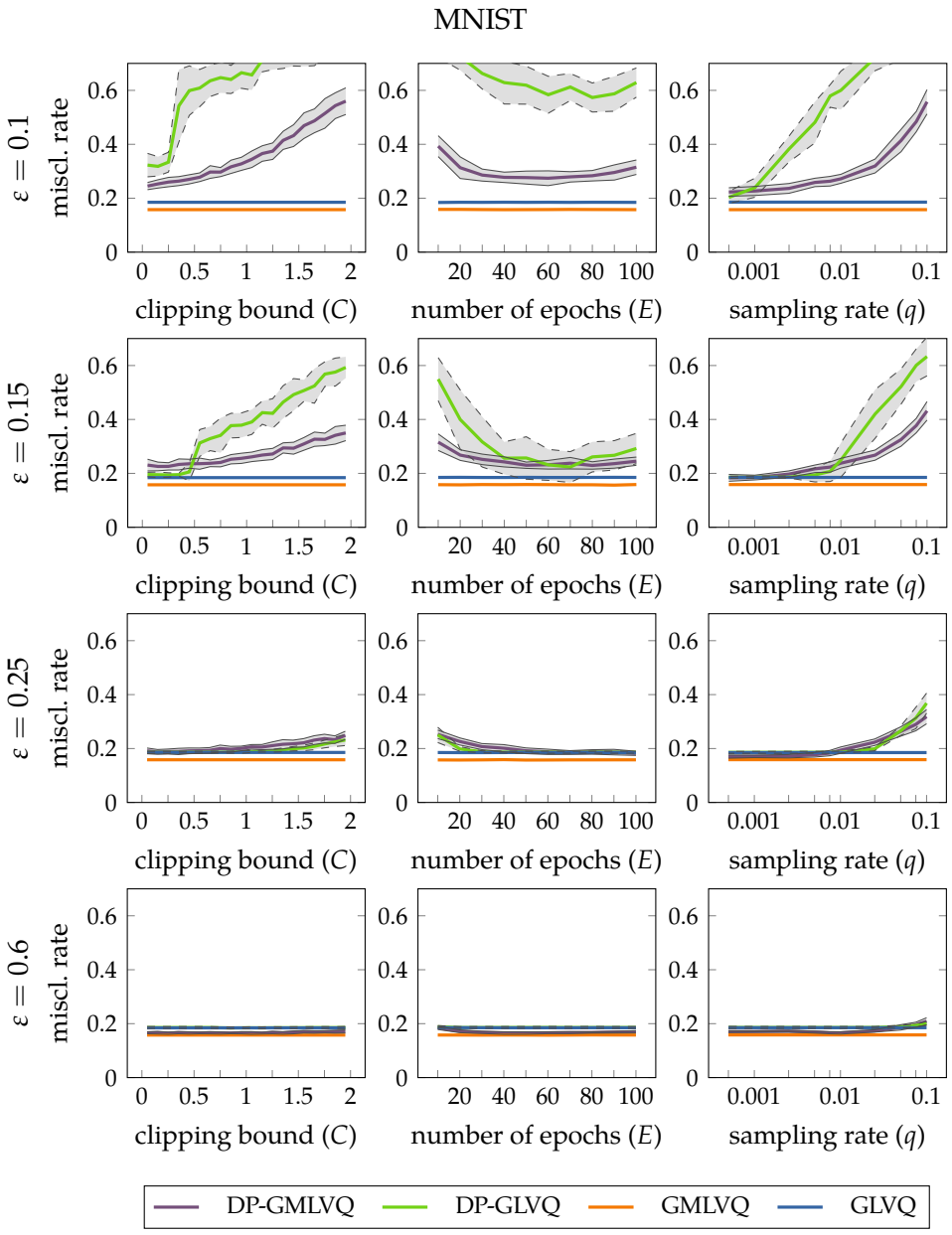


Figure B.1: Averaged GLVQ and GMLVQ misclassification rates for MNIST and different hyperparameters C, E, and q with $\varepsilon \in \{0.1, 0.15, 0.25, 0.6, 0.75\}$

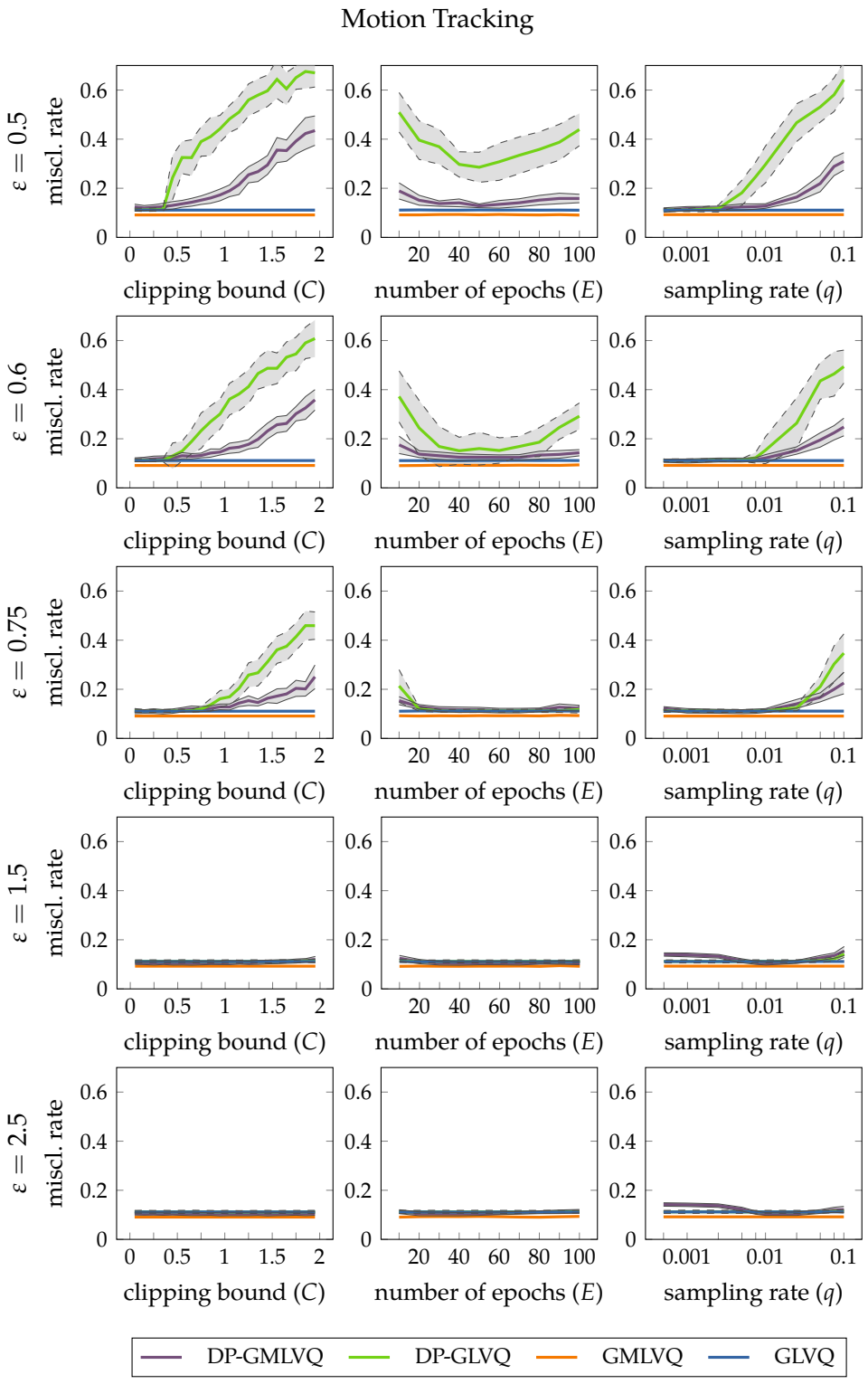


Figure B.2: Averaged GLVQ and GMLVQ misclassification rates for Motion Tracking and different hyperparameters C, E, and q with $\varepsilon \in \{0.5, 0.6, 0.75, 1.5, 2.5\}$

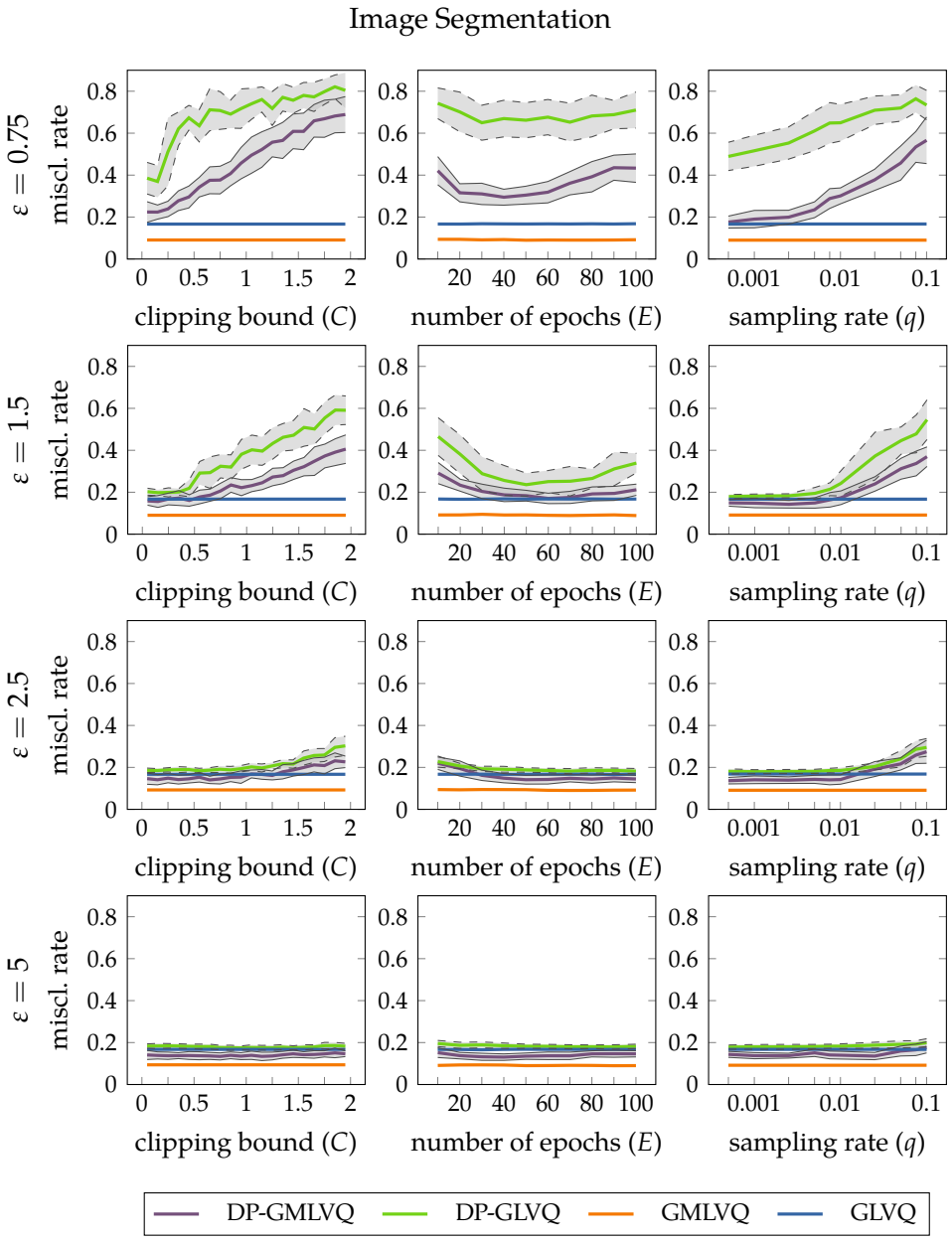


Figure B.3: Averaged GLVQ and GMLVQ misclassification rates for Image Segmentation and different hyperparameters C, E, and q with $\varepsilon \in \{0.75, 1, 1.5, 2.5, 5\}$

B.3 FURTHER RESULTS FOR SGD

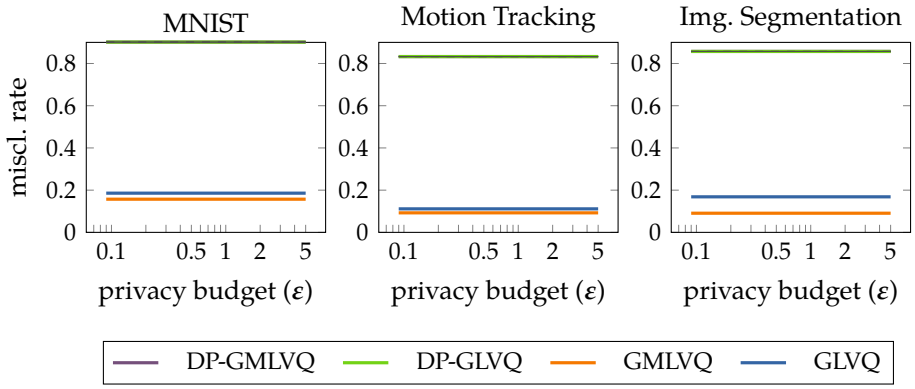


Figure B.4: Results for differentially private SGD without initialization of prototypes to class means (i. e. $\varepsilon_1 = 0$ in Section 3.4.1). Even setting the privacy budget to 5 does not lead to a viable model. The required number of epochs for finding reliable prototype positions is much too big.

Data set	$\varepsilon = 0.25$	$\varepsilon = 0.75$	$\varepsilon = 1$	$\varepsilon = 1.5$	$\varepsilon = 2.5$	$\varepsilon = 5$	non DP SGD	non DP BFGS
MNIST	0.186 (0.0038)	0.186 (0.0038)	0.186 (0.0037)	0.186 (0.0036)	0.186 (0.0037)	0.186 (0.0036)	0.185 (0.0033)	0.184 (0.0037)
	0.191 (0.0096)	0.165 (0.0038)	0.165 (0.0033)	0.165 (0.0033)	0.165 (0.0036)	0.165 (0.0033)	0.158 (0.0030)	0.148 (0.0029)
Mot. Track.	0.703 (0.0705)	0.113 (0.0058)	0.112 (0.0064)	0.112 (0.0059)	0.112 (0.0058)	0.112 (0.0056)	0.111 (0.0065)	0.111 (0.0063)
	0.273 (0.0324)	0.113 (0.0072)	0.108 (0.0072)	0.104 (0.0058)	0.104 (0.0054)	0.104 (0.0053)	0.094 (0.0080)	0.090 (0.0087)
Img. Seg.	0.823 (0.0658)	0.646 (0.0918)	0.497 (0.0942)	0.251 (0.0541)	0.188 (0.0126)	0.183 (0.0099)	0.167 (0.0103)	0.167 (0.0105)
	0.691 (0.0621)	0.321 (0.0592)	0.237 (0.0251)	0.169 (0.0224)	0.144 (0.0224)	0.134 (0.0168)	0.092 (0.0136)	0.089 (0.0131)

Table B.1: Mean and standard deviation (in parenthesis) of the misclassification rates for all benchmarks. As a baseline, the results of a non-private training with SGD and a BFGS optimizer are given. The first rows for each data set are results for GLVQ the second for GMLVQ.

B.4 ARTIFICIAL DATA SET

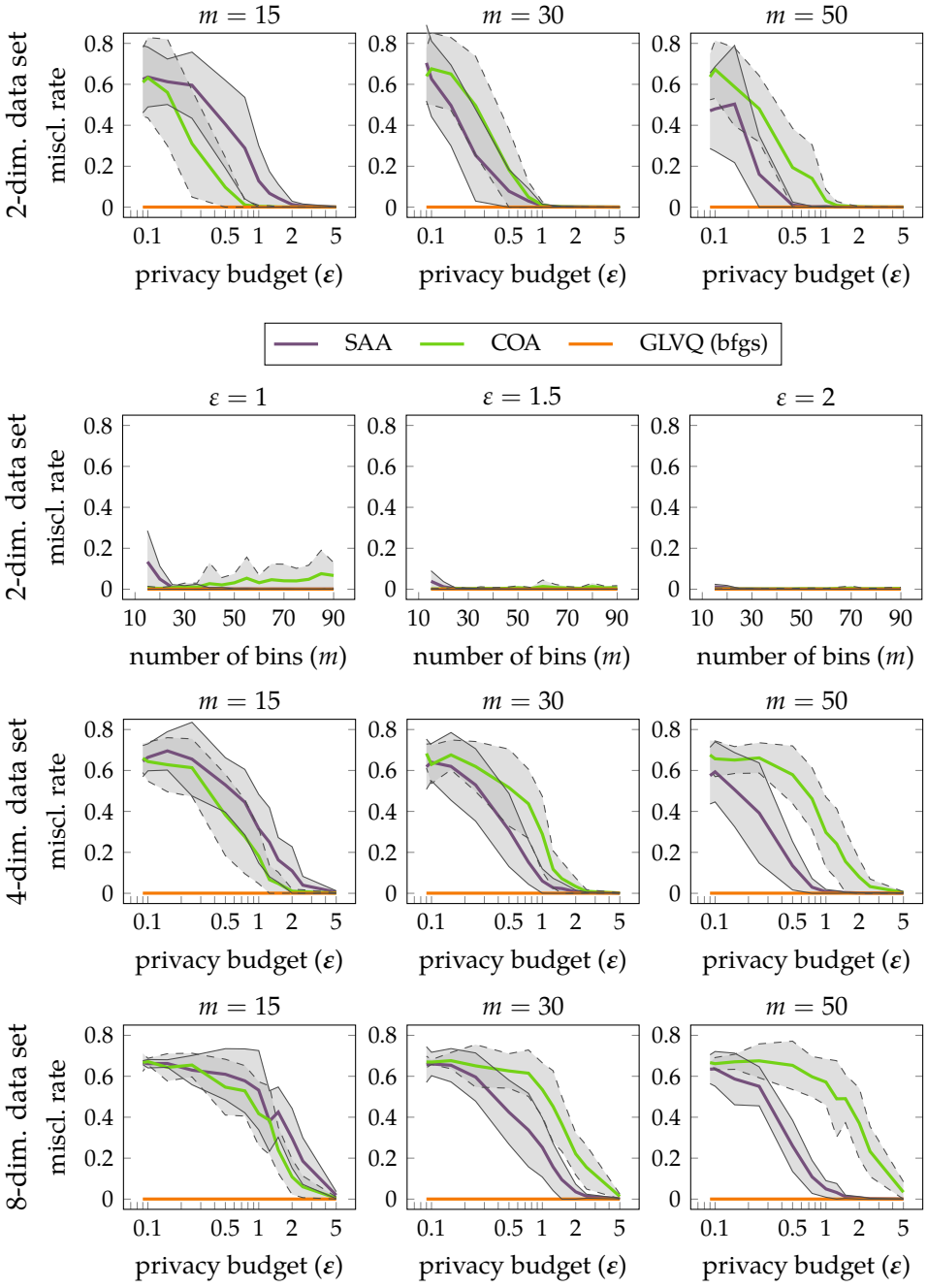


Figure B.5: Averaged GLVQ misclassification rates for the big artificial data sets (same properties as the artificial data set in Section 3.5.2 but with 10 000 samples per class, 30 000 in total) with different dimensions (2 in the first, 4 in the third and 8 in the forth row) and different numbers of bins ($m \in \{15, 30, 50\}$). In the second row, the misclassification rate is plotted against m for three privacy budgets ($\varepsilon \in \{1, 1.5, 2\}$).

APPENDIX — SPARSE LVQ AND HYPERPARAMETER FOR SPARSENESS

C

We provide further results for SLGMLVQ varying the parameter to limit the projection dimensionality m. In addition, we also give the plots for the sparsity parameter S for FRI, Motion Tracking, and Gisette in Figures C.1 to C.3, respectively.

C.1 FRI

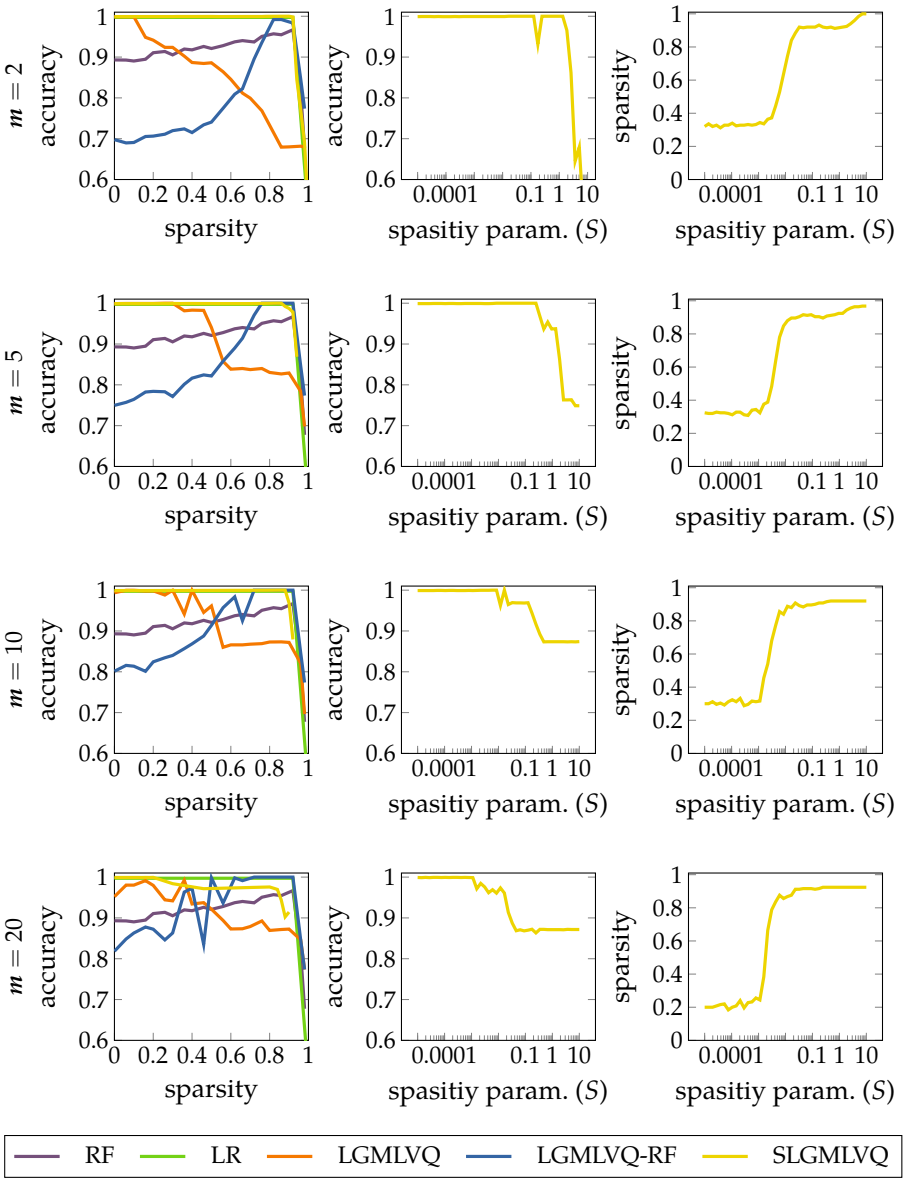


Figure C.1: Results of SLGMLVQ for FRI and four projection dimensions $m \in \{2, 5, 10, 20\}$

C.2 MOTION TRACKING

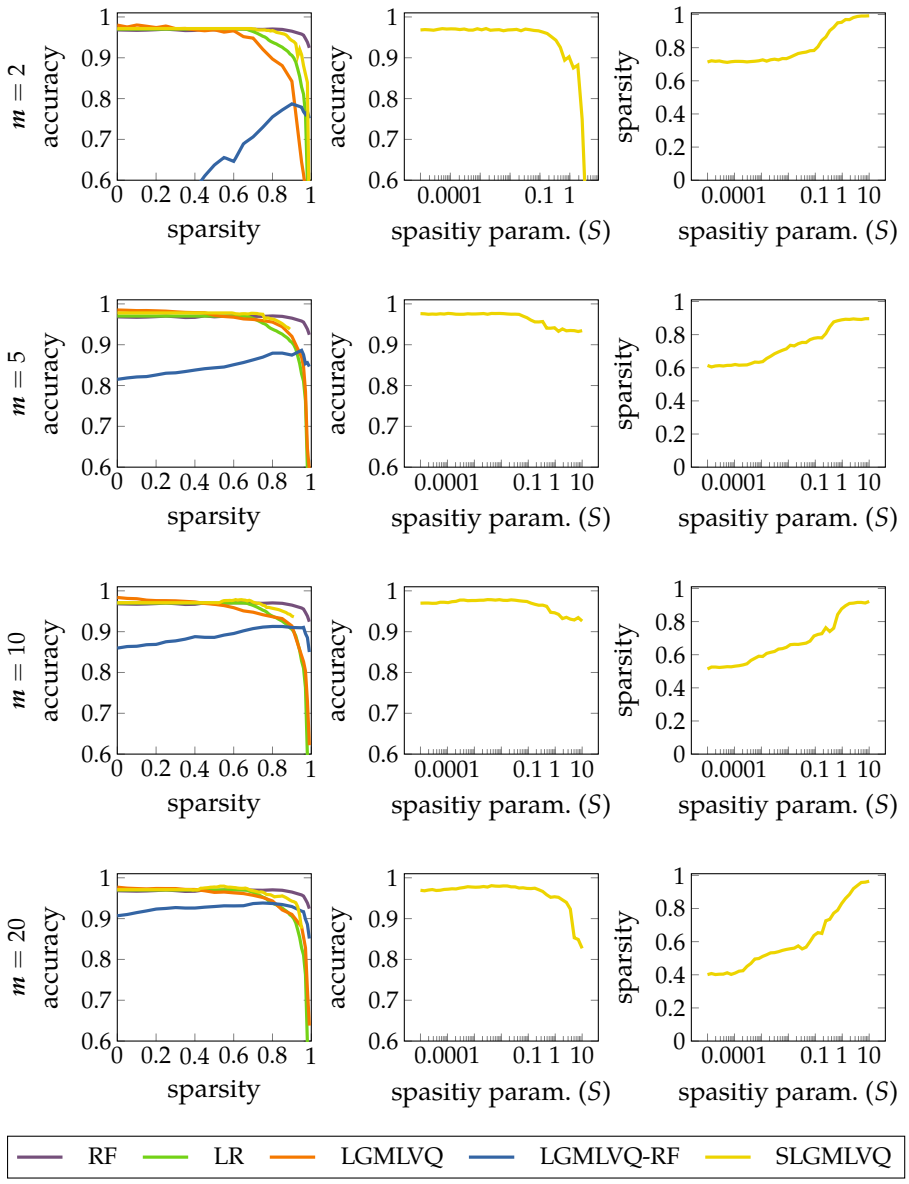


Figure C.2: Results of SLGMLVQ for Motion Tracking and four projection dimensions $m \in \{2, 5, 10, 20\}$

C.3 GISETTE

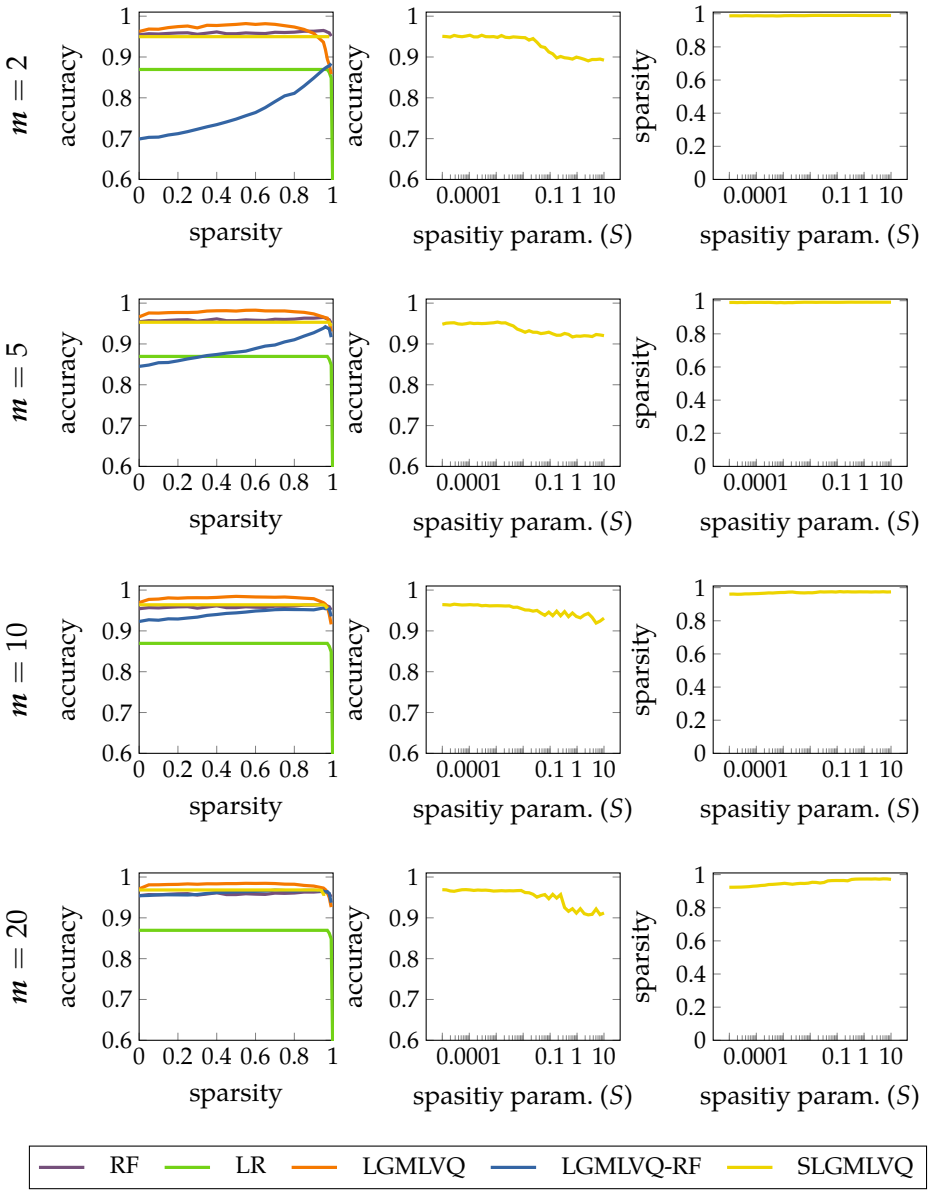


Figure C.3: Results of SLGMLVQvarying m for Gisette and four projection dimensions $m \in \{2, 5, 10, 20\}$