View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Technische Universität Dresden: Qucosa

Pairwise Classification and Pairwise Support Vector Machines

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

Doctor rerum naturalium

(Dr. rer. nat.)

vorgelegt

der Fakultät Mathematik und Naturwissenschaften
der Technischen Universität Dresden

von

Diplommathematiker Carl Brunner

geboren am 13. März 1982 in Bad Dürkheim

Gutachter:
Prof. Dr. Andreas Fischer
Technische Universität Dresden
Institut für Numerische Mathematik

Prof. Dr. Laura Palagi
Sapienza Università di Roma, Dipartimento di Ingegneria Informatica,
Automatica e Gestionale Antonio Ruberti

Eingereicht am: 19. Dezember 2011

For Denise & Chiara
Sine quibus non

Preface

The first time I heard about machine learning and Support Vector Machines was in the end of 2007, when I was talking with Professor Andreas Fischer about a possible topic for my diploma thesis. While writing this thesis I was asked if I would be interested in working for a research project between the Technical University of Dresden and Cognitec Systems GmbH and meanwhile writing my PhD thesis.

In a classification task one wants to classify the members of a given set of examples into several disjoint classes. For training a set of examples is given and a decision function is determined which is then tested on another set of examples. Any class represented by at least one example used for training or testing belongs to the training classes or test classes, respectivly. The purpose of this dissertation is to provide a way to extend classification tasks to an arbitrarily large number of classes while many of the test classes do not belong to the training classes. To reach that goal the pairwise classification approach is proposed and the extension of non pairwise classification tasks to pairwise classification tasks is analyzed. It is shown that the extension of an ordinary data generating process to a pairwise data generating process is not trivial. However, problems like the needed training time, the enforcement of certain properties of the decision function, or optimal pairwise classifiers are successfully addressed. The unique contributions of this thesis is a in depth analysis of pairwise classification tasks and methods which make pairwise classification practical. This dissertation may be of interest to anybody in the field of machine learning, especially to those which are interested in face recognition or collaborative filtering.

Many of the results presented in this dissertation were found during the research project. I would like to thank Thorsten Thies, Klaus Luig and Frank Weber for introducing me to pairwise SVMs and providing me with their first results on this topic. Additionally, I would like to thank them, Professor Andreas Fischer and Gerd Langensiepen for many discussions an numerous advice on my research.

This research was supported by Cognitec Systems GmbH.

Carl Brunner

December 2011
Dresden, Germany

Contents

List of Figures

List of Tables

1 Introduction

A recent approach for multiclass classification is the pairwise classification which relies on two input examples instead of one and predicts whether the two input examples belong to the same class or to different classes (see [1, 2, 3, 5, 7, 19, 32, 43]). A common pairwise classification task is face recognition. In this area, a set of images is given for training and another set is given for testing. Often, one is interested in a good interclass generalization. The latter means that any person which is represented by an image in the training set is not represented by any image in the test set.

For a pairwise classifier the order of the two examples should not influence the classification result. A common approach to enforce this symmetry is the use of selected kernels. Relations between such kernels and certain projections are provided. It is shown, that those projections can lead to an information loss. For pairwise SVMs another approach for enforcing symmetry is the symmetrization of the training sets. In other words, if the pair (a, b) of examples is a training pair then (b, a) is a training pair, too. It is proven that both approaches do lead to the same decision function for selected parameters. Empirical tests show that the approach using selected kernels is three to four times faster. For a good interclass generalization of pairwise SVMs training sets with several million training pairs are needed. A technique is presented which further speeds up the training time of pairwise SVMs by a factor of up to 130 and thus enables the learning of training sets with several million pairs. Another element affecting time is the need to select several parameters. Even with the applied speed up techniques a grid search over the set of parameters would be very expensive. Therefore, a model selection technique is introduced that is much less computationally expensive.

In machine learning, the training set and the test set are created by using some data generating process. Several pairwise data generating processes are derived from a given non pairwise data generating process. Advantages and disadvantages of the different pairwise data generating processes are evaluated.

Pairwise Bayes' Classifiers are introduced and their properties are discussed. It is shown that pairwise Bayes' Classifiers for interclass generalization tasks can differ from pairwise Bayes' Classifiers for interexample generalization tasks. In face recognition the interexample task implies that each person which is represented by an image in the test set is also represented by at least one image in the training set. Moreover, the set of images of the training set and the set of images of the test set are disjoint.

Pairwise SVMs are applied to four synthetic and to two real world datasets. One of the real world datasets is the Labeled Faces in the Wild (LFW) database while the other one is provided by Cognitec Systems GmbH. Empirical evidence for the presented model selection heuristic, the discussion about the loss of information and the provided speed up techniques is given by the synthetic databases and it is shown that classifiers of pairwise SVMs lead to a similar quality as pairwise Bayes' classifiers. Additionally, a pairwise classifier is identified for the LFW database which leads to an average equal error rate (EER) of 0.0947 with a standard error of the mean (SEM) of 0.0057. This result is better than the result of the current state of the art classifier, namely the combined probabilistic linear discriminant analysis classifier, which leads to an average EER of 0.0993 and a SEM of 0.0051.

This thesis is structured as follows. Chapter 2 presents theoretical background and literature review of optimization theory and of machine learning. Furthermore, several proofs are given which have not been presented in literature to the best of my knowledge. In Chapter 3 pairwise classification is introduced. Several properties and the quality of pairwise decision functions are discussed. Moreover, optimal pairwise classifiers and a new model selection technique are introduced and pairwise data generating processes are analyzed. In Chapter 4 pairwise SVMs and pairwise kernels are introduced and several methods to enforce the pairwise symmetry of a pairwise decision function are discussed. Additionally, the evaluation of pairwise kernel function values is investigated. The efficient implementation of pairwise SVMs is discussed in Chapter 5, followed by performance measurements for five synthetic datasets and for two real world datasets. At the end of this thesis you can find a list of symbols and abbreviations.

2 Preliminaries

Here, an overview over selected topics of optimization theory and over machine learning which are used throughout this dissertation is provided. A more elaborate presentation can be found in [38, 42, 44].

First, relevant topics of optimization theory are presented in Section 2.1. In Section 2.2 an overview about the basic concepts of statistical learning theory is given, followed by a short introduction to Support Vector Machines (SVMs) in Section 2.3. Section 2.4 reviews the Bayes' rule of classification. This is another classification rule which is optimal in some sense. This chapter is concluded in Section 2.5 by discussing ways of evaluating the quality of given classifiers.

2.1 Optimization Theory

Several restrained optimization problems occur in this dissertation. Hence, the optimization theory for these restrained optimization problems is introduced. For further information the interested reader is referred to [9, 20].

Definition 2.1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable. Furthermore, let $g_i : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable for all $i \in \{1, \dots p\}$ and let $h_j : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable for all $j \in \{1 \dots q\}$ and let G be defined by*

$$G := \{x \in \mathbb{R}^n \mid g_i(x) \leq 0 \text{ for all } i \in \{1, \dots, p\}, h_j(x) = 0 \text{ for all } j \in \{1, \dots, q\}\}.$$

*Then, any $x^* \in G$ with $f(x^*) \leq f(x)$ for all $x \in G$ is called **solution** of the **Constrained Optimization Problem**. Note that any constrained optimization problem can be also written as*

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f(x) \\ \text{s.t. } &g_i(x) \leq 0 \text{ for all } i \in \{1, \dots, p\} \\ &h_j(x) = 0 \text{ for all } j \in \{1, \dots, q\}. \end{aligned} \tag{2.1}$$

If one further assumes that f is a convex function and that G is a convex set then (2.1) *is called* ***Convex Optimization Problem****.*

Remark 2.2.

- *According to the Weierstrass-Theorem the following holds: If G is closed and bounded, then for any constrained optimization problem there exists a solution x^* of* (2.1)*.*
- *If f is strictly convex and G is convex, closed, and bounded, then a unique solution exists.*
- *If f and G are convex and x_1^* and x_2^* are solutions of problem* (2.1) *then $(\gamma x_1^* + (1-\gamma)x_2^*)$ is also a solution for all $\gamma \in (0,1)$.*

Now, necessary and sufficient conditions for a point $x \in G$ to be a solution are presented.

Definition 2.3. *Let a constrained optimization problem of the form* (2.1) *be given. Then, the function $L : \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$ defined by*

$$L(x, \lambda, \mu) := f(x) + \sum_{i=1}^{p} \lambda_i g_i(x) + \sum_{j=1}^{q} \mu_j h_j(x).$$

is called ***Lagrange Function*** *of the constrained optimization problem.*

Definition 2.4. *Let the constrained optimization problem* (2.1) *be considered.*

- *Then, the conditions*

$$\begin{aligned} \nabla_x L(x, \lambda, \mu) &= 0 \\ h_j(x) &= 0, \ \textit{for all } j \in \{1, \dots, q\} \\ g_i(x) &\leq 0, \ \textit{for all } i \in \{1, \dots, p\} \\ \lambda_i &\geq 0, \ \textit{for all } i \in \{1, \dots, p\} \\ \sum_{i=1}^{p} g_i(x)\lambda_i &= 0 \end{aligned}$$

  are called ***Karush-Kuhn-Tucker (KKT) Conditions****.*

- *Any point $(x^*, \lambda^*, \mu^*) \in \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^q$ which fulfills the KKT Conditions is called* ***Karush-Kuhn-Tucker Point*** *associated with the constrained optimization problem* (2.1)*. The vectors λ^* and μ^* of such a point are called* ***Lagrange Multipliers****.*

In literature several conditions can be found which imply that for any (local) minimum x^* of (2.1) there exist Lagrange Multipliers λ^* and μ^* such that (x^*, λ^*, μ^*) is a KKT Point. In this thesis any occurring optimization problem is convex and its constraints are linear functions. Therefore, only the following results are presented.

Theorem 2.5 (KKT Conditions under Linear Constraints). *Let x^* be a solution of (2.1). Furthermore, let any $g_i, i \in \{1, \dots, p\}$ and any $h_j, j \in \{1, \dots, q\}$ be an affine function. Then, there exist Lagrange Multipliers $\lambda^* \in \mathbb{R}^p$ and $\mu^* \in \mathbb{R}^q$ such that (x^*, λ^*, μ^*) is a KKT Point of (2.1).*

Proof. A proof is provided by [20]. □

Theorem 2.6. *Let (x^*, λ^*, μ^*) be a KKT Point of a convex optimization problem of the form (2.1). Then, x^* is a solution.*

Proof. A proof is provided by [20]. □

Corollary 2.7. *Let a convex optimization problem of the form (2.1) be given and let all of its constraints be affine functions. Then, x^* is a solution of (2.1) if and only if there exist Lagrange Multipliers $\lambda^* \in \mathbb{R}^p$ and $\mu^* \in \mathbb{R}^q$ such that (x^*, λ^*, μ^*) is a KKT Point of (2.1).*

Proof. The proof follows directly by combining Theorems 2.5 and 2.6. □

2.2 Statistical Learning Theory

Now, a short overview of Statistical Learning Theory is given. Subsections 2.2.1, 2.2.2 and 2.2.3 follows mainly [42, Sections 1-4], while Subsection 2.2.4 is based upon [44].

2.2.1 Model of the Data Generating Process

In machine learning one wants to find a functional dependency between some input space X and some output space Y. Here, only the case $X \subseteq \mathbb{R}^n$ is considered and it is assumed that the underlying learning model can be described in the following way (see Figure 2.1):

The model contains three elements

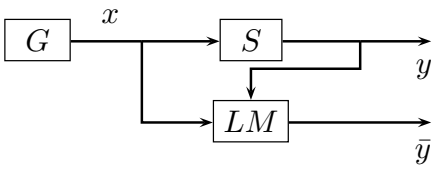


Figure 2.1: Model of learning from examples (see [42, pp. 19-21]). The generator G gives an example x to the supervisor S and the learning machine LM. The supervisor predicts the (correct) y which is given to the LM, too. During training the LM observes pairs (x_i, y_i) (the training set). After training the machine must return for any x an $\bar{y}$ which should be close to y.

1. The generator G of the (input) points.
2. The supervisor.
3. The learning machine.

The generator G creates (input) points $x \in X$ according to (unknown) probability functions $F_t(x)$ with $t \in \mathbb{Z}$. Here, t denotes the time and implies the time dependence of the underlying probability function. Another way to describe G is that it generates the points according to a time discrete stochastic process $Z(t), t \in \mathbb{Z}$. However, it is assumed that $F_t = F_X$ for all $t \in \mathbb{Z}$. Hence, the drawings are identically distributed and the stochastic process is strongly stationary. The interested reader is referred to the paper [39] for a more general setting.

Each created point $x \in X$ is input to the supervisor which returns the true output (label) $y \in Y$ to the input x. Note that one needs to assume that this supervisor exists.

A learning machine works in the following way. Firstly, it observes the **training set** or **set of training points**

$$(x_1, y_1), \ldots, (x_m, y_m) \in \{X \times Y\}^m$$

which contains the input points $x_i \in X$ and the answers of the supervisor $y_i \in Y$ with $i \in M := \{1, \ldots, m\}$. Secondly, the learning machine constructs a predictor that given some $x \in X$ gives an answer $\bar{y}$ which should be similar to the answer y of the supervisor.

Formally speaking, the generator G creates the points $x \in X$ accordingly to a probability distribution function F_X where one additionally assumes that a corresponding density p_X exists. Then, the supervisor returns the output y on the

input x according to a conditional distribution function $F_Y(\cdot|X = x)$ with density $p_Y(\cdot|X = x)$. Hence, the learning machine observes the training set which is drawn randomly according to a joint distribution $F_{X\times Y}(\cdot,\cdot)$ with corresponding density $p_{X,Y}$ with $p_{X,Y}(x, y) := p_X(x)p_Y(y|X = x)$. Consequently, the learning machine constructs an approximation $g : X \to Y$ to the answer of the unknown supervisor. This approximation is called **decision function**. In other words, the learning machine chooses the decision function out of a given set of functions. In the following, it is assumed that the set of decision functions which can be chosen by the learning machine can be written in parametric form as $\{g_\lambda : X \to Y|\lambda \in \Lambda\}$, hence, each $\lambda \in \Lambda$ corresponds to a single decision function denoted by g_λ.

If $Y = \{-1, 1\}$, then $\mathrm{sgn}(g(x))$ is often used as **classifier** or **classification rule**. If not stated otherwise $\mathrm{sgn}(0)$ is set to 1. Additionally, if $Y = \{-1, 1\}$, then a training point (x_i, y_i) is called positive training point if $y_i = 1$ and it is called negative training point if $y_i = -1$.

Now, a list of remarks is presented.

- The presented setting includes the case when the supervisor is deterministic and uses some function $h : X \to Y$ to return the output. In this dissertation the more general setting described above is needed. In other words, the answer of the supervisor might not be unique for any given $x \in X$.

- The presented model is very general. It includes the case of pattern recognition (the input space Y is a discrete set) and the case of regression estimation (Y is a continuous set). In this thesis only the case of pattern recognition is considered.

- In pattern recognition one calls Y the **set of classes** and each $y \in Y$ **class**.

- As already stated it is assumed in this section that the drawings of the points $x \in X$ are identically distributed. In Subsections 2.2.2 and 2.2.3 it is assumed that those drawings are independent, too. A more general setting of identically distributed but dependent drawings is discussed in Subsection 2.2.4.

- In machine learning one wants to find an approximation to the supervisor. This supervisor classifies according to the distribution $F_{X\times Y}$. If not stated otherwise, this distribution is unknown. If the distribution $F_{X\times Y}$ is known, then it is possible to use the Bayes' Classifier (see Section 2.4).

2.2.2 Empirical Risk Minimization

Let a decision function g be given. Then, to determine the quality of g one needs a **loss function**

$$l : Y \times \mathbb{R} \to \mathbb{R}_+.$$

This loss function is used to penalize wrong answers of g. For example:

$$\begin{aligned} \text{0-1-loss: } l(y, \bar{y}) &:= \begin{cases} 0 & \text{if } y = \bar{y} \\ 1 & \text{else} \end{cases} \\ \text{(for } Y = \{-1, 1\}\text{) hinge loss: } l(y, \bar{y}) &:= (1 - y \cdot \bar{y})_+ := \max\{1 - y \cdot \bar{y}, 0\}. \end{aligned} \tag{2.2}$$

Here, y can be seen as the correct answer (or the answer of the supervisor) and $\bar{y}$ would be the answer of the decision function. Now, one can define the **risk** R of a decision function $g_\lambda : X \to \mathbb{R}$ by

$$R(\lambda) := \int_{X \times Y} l(y, g_\lambda(x)) \, \mathrm{d}F_{X \times Y}(x, y), \tag{2.3}$$

where the loss function $l(\cdot, g_\lambda(\cdot))$ is assumed to be integrable for any g_λ with $\lambda \in \Lambda$. Hence, the task of learning is to find a λ^* with

$$\lambda^* \in \operatorname*{argmin}_{\lambda \in \Lambda} \{R(\lambda)\}. \tag{2.4}$$

As $F_{X \times Y}$ is unknown the risk cannot be minimized directly. The basic idea of the **Empirical Risk Minimization** (ERM) method is to choose a decision function which minimizes the **empirical risk**

$$R_{\mathsf{emp}}(\lambda) := \frac{1}{m} \sum_{i=1}^{m} l(y_i, g_\lambda(x_i)) \tag{2.5}$$

instead of the risk. Hence, instead of choosing a g_{λ^*} one chooses a g_{λ^m} by means of

$$\lambda^m \in \operatorname*{argmin}_{\lambda \in \Lambda} \{R_{\mathsf{emp}}(\lambda)\}. \tag{2.6}$$

Without further restrictions the approach of minimizing the empirical risk might not lead to good results. For example, let the 0-1-loss be selected as loss function and

let the following decision function

$$\bar{g}(x) := \begin{cases} y_i & \text{if } x \in \{x_i\}_{i \in M} \\ 1 & \text{otherwise} \end{cases}$$

be defined. Obviously, $\bar{g}$ has zero empirical risk (2.5), whereas its risk (2.3) will not be optimal in general.

Thus, one is looking for a decision function which minimizes the empirical risk and has a good generalization ability, in other words for large values of m the difference between the risk (2.3) and the empirical risk (2.5) is small in some sense. Now, let $(\mathbb{R}, \Sigma, P)$ denote a probability space. Then, one says that a sequence of random variables $a_1, \ldots, a_m, \ldots$ converges to a random variable a_0 in probability if

$$P\{|a_m - a_0| > \delta\} \xrightarrow[m \to \infty]{} 0$$

is valid for any $\delta > 0$. From now on,

$$a_m \xrightarrow[m \to \infty]{P} a_0$$

denotes the convergence in probability of $a_1, \ldots, a_m, \ldots$ to a_0.

Now, conditions when the ERM method leads to the convergences (in probability) of $R_{\text{emp}}(\lambda^m)$ to $R(\lambda^*)$ and of $R(\lambda^m)$ to $R(\lambda^*)$ for $m \to \infty$ are described (see Figure 2.2).

Definition 2.8. *[42, p. 82] Let*

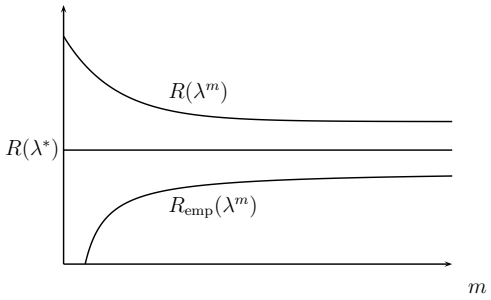
$$\Lambda(c) := \{\lambda \in \Lambda : R(\lambda) \geq c\}$$

be defined. The ERM method is said to be ***strictly consistent*** *for the set of functions* $\{g_\lambda | \lambda \in \Lambda\}$ *and the probability distribution function* $F_{X \times Y}$ *if for any* $c > 0$ *with* $\Lambda(c) \neq \emptyset$ *the convergence in probability*

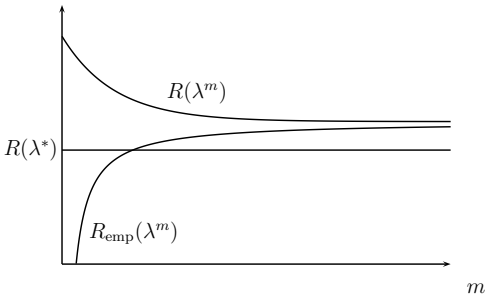
$$\inf_{\lambda \in \Lambda(c)} R_{emp}(\lambda) \xrightarrow[m \to \infty]{P} \inf_{\lambda \in \Lambda(c)} R(\lambda) \tag{2.7}$$

takes place.

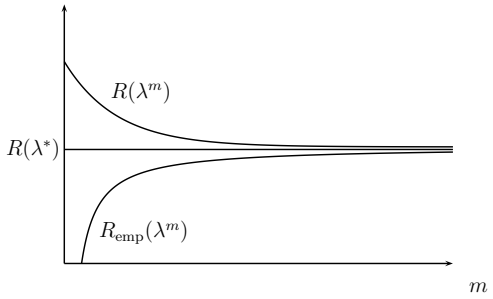
In other words, the ERM method is strictly consistent if convergence (2.7) takes place for the given set of functions and for all subsets $\Lambda(c)$ of functions that remain after the functions with a corresponding risk value smaller than c are excluded from the set.



(a) Although $R(\lambda^m)$ and $R_{\text{emp}}(\lambda^m)$ converge to a fixed value they do not converge to each other.



(b) $R(\lambda^m)$ and $R_{\text{emp}}(\lambda^m)$ converge to each other but they do not converge to the minimal risk.



(c) $R(\lambda^m)$ and $R_{\text{emp}}(\lambda^m)$ converge to $R(\lambda^*)$ (see (2.4)). In this case the learning process is consistent.

Figure 2.2: Here, λ^m is chosen according to (2.6). The convergence of $R(\lambda^m)$ and $R_{\text{emp}}(\lambda^m)$ are analyzed for $m \to \infty$.

In Definition 2.8 the convergence of the empirical risk to the minimal risk is used, while the convergence of the risk to the minimal risk is not used. The following lemma shows that this convergence also holds true for strictly consistent learning methods.

Lemma 2.9. *[42, p. 82] If the ERM method is strictly consistent, then the following convergence in probability holds*

$$R(\lambda^m) \xrightarrow[m\to\infty]{P} \inf_{\lambda\in\Lambda} R(\lambda).$$

In [42] several conditions which lead to the (strict) consistency of the ERM method are given. The main result is the connection between the strict consistency of the ERM method and uniform one sided convergence of an empirical process which is presented in Corollary 2.10.

To define uniform one sided convergence, let the probability distribution function

$F_{X\times Y}(\cdot,\cdot)$ be defined on $\mathbb{R}^n\times\mathbb{R}$, and let $l(\cdot,g_\lambda(\cdot))$ with $\lambda\in\Lambda$ be a set of integrable (with respect to this distribution) loss functions. Let

$$(x_1,y_1),\ldots,(x_m,y_m),\ldots$$

be a sequence of independent identically distributed (i.i.d.) training points and let $(\mathbb{R},\Sigma,P)$ denote a probability space. Now, let the following sequence of random variables be considered

$$\eta^m := \sup_{\lambda\in\Lambda}\left(\int_{X\times Y} l(y,g_\lambda(x))\,\mathrm{d}F_{X\times Y}(x,y) - \frac{1}{m}\sum_{i=1}^{m} l(y,g_\lambda(x_i))\right) \tag{2.8}$$

with $m\in\mathbb{N}$. Then, one looks at the convergence

$$P\left(|\eta^m|>\varepsilon\right)\xrightarrow[m\to\infty]{} 0.$$

If this convergence takes place for any $\varepsilon>0$, then this is called **uniform convergence** of the sequence $\{\eta^m\}$. Additionally, if the convergence

$$P\left(\eta_+^m>\varepsilon\right)\xrightarrow[m\to\infty]{} 0$$

takes place for any $\varepsilon>0$ with

$$\eta_+^m := \sup_{\lambda\in\Lambda}\left(\int_{X\times Y} l(y,g_\lambda(x))\,\mathrm{d}F_{X\times Y}(x,y) - \frac{1}{m}\sum_{i=1}^{m} l(y,g_\lambda(x_i))\right)_+$$

then this convergence is called **uniform one sided convergence** of the sequence $\{\eta_+^m\}$. Here, it is assumed that $l(\cdot,g_\lambda(\cdot))$ is integrable for any $\lambda\in\Lambda$.

Corollary 2.10. *[42, p. 88] Let l be a loss function. Additionally, let any function in the set $\{l(\cdot,g_\lambda(\cdot)),\lambda\in\Lambda\}$ be integrable for all distribution functions $F_{X\times Y}\in\mathcal{P}$. Here, the only condition for $\mathcal{P}$ is that any $F_{X\times Y}\in\mathcal{P}$ is a distribution function with $F_{X\times Y}: X\times Y\to\mathbb{R}$. Moreover, let there be constants $a,A\in\mathbb{R}$ such that for all functions in $\{l(\cdot,g_\lambda(\cdot)),\lambda\in\Lambda\}$ and all $F_{X\times Y}\in\mathcal{P}$ the inequalities*

$$a\le\int_{X\times Y} l(y,g_\lambda(x))\,\mathrm{d}F_{X\times Y}(x,y)\le A$$

hold true. Then, the following statements are equivalent:

i) For any distribution function in the set $\mathcal{P}$, the ERM method is strictly consistent on the set of functions $\{l(\cdot,g_\lambda(\cdot)),\lambda\in\Lambda\}$.

ii) For any distribution function in the set $\mathcal{P}$, the one sided convergence takes

place on the set of functions $\{l(\cdot, g_\lambda(\cdot)), \lambda \in \Lambda\}$.

Note that one can set $a = 0, A = 1$ if the 0-1-loss function is selected.

The last corollary shows that conditions which lead to uniform one sided convergence of the ERM method lead to strict consistency. To derive conditions which lead to uniform one sided convergence some kind of measurement of the entropy of the set of implementable functions $\{l(\cdot, g_\lambda(\cdot)), \lambda \in \Lambda\}$ is needed. To this end, the Vapnik-Chervonenkis (VC) dimension is defined.

Definition 2.11.

- *Let* l *be the 0-1-loss. The* ***VC dimension*** *of the set*

$$\{l(\cdot, g_\lambda(\cdot)), \lambda \in \Lambda\}$$

is defined as the largest number h *of vectors* $x_1, \ldots, x_h$ *such that for any* $\bar{y} \in \{-1, 1\}^h$ *there is a function* $g_\lambda, \lambda \in \Lambda$ *with* $l(\bar{y}_i, g_\lambda(x_i)) = 0$ *for all* $i \in \{1, \ldots, h\}$ *(see [42, p. 147] and Figure 2.3).*

- *Let* l *be an arbitrary loss function. Then, the* ***VC dimension*** *of the set*

$$\{l(\cdot, g_\lambda(\cdot)), \lambda \in \Lambda\}$$

is equal to the VC dimension of the following set of indicator functions (or 0-1-loss functions):

$$\{\theta\left(l(\cdot, g_\lambda(\cdot)) - \beta\right), \lambda \in \Lambda, \beta \in \mathcal{B}\}$$

with

$$\theta(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{else} \end{cases} \quad \text{and} \quad \mathcal{B} := \left(\inf_{\lambda,x,y} \{l(y, g_\lambda(x))\}, \sup_{\lambda,x,y} \{l\left(y, g_\lambda(x)\right)\}\right)$$

(see [42, p.191]).

Remark 2.12. *For example, the VC dimension of a set of functions*

$$\{\langle w, x\rangle + b, (w, b) \in \mathbb{R}^n \times \mathbb{R}\}$$

is $n + 1$ *independently of the chosen loss function (see [42, p. 156 and p. 192]).*

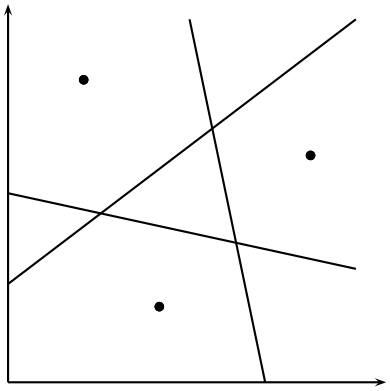


Figure 2.3: The VC dimension of affine linear functions in $\mathbb{R}^2$ is 3. One can label the three points arbitrarily with $+1$ or -1 and then separate the positive point(s) from the negative point(s) by using the provided functions. This works only if there is at least one positive and one negative point. Otherwise, the separation is trivial. For four points such a separation is not possible with affine linear functions in $\mathbb{R}^2$.

Theorem 2.13. *[42, Theorem 3.3] Let l be the 0-1-loss function and η^m be defined according to* (2.8)*. Then, if the VC dimension of the set $\{l(\cdot, g_\lambda(\cdot)), \lambda \in \Lambda\}$ is finite, the sequence $\{\eta^m\}$ convergences uniformly.*

Remark 2.14. *In [42, Theorem 3.8] the last theorem is extended to the case of bounded loss functions, in other words $|l(y, g_\lambda(x))| < A$ holds for some $A \in \mathbb{R}$ and all $(x, y) \in X \times Y$ and all $\lambda \in \Lambda$. However, many more technical details would be needed. Section 2.3 introduces Support Vector Machines (SVMs). Often, the hinge loss is chosen as loss function for SVMs. In general, the hinge loss is unbounded. However, in [8] the authors proof in Lemma 23 that for SVMs the hinge loss is bounded if the input data are bounded.*

Now, the risk is bounded by means of the empirical risk. The presented bounds show, that a finite VC dimension implies uniform one sided convergence which again implies the consistency of the ERM method.

Theorem 2.15. *[42, Theorem 5.3] Let $0 \leq l(y, g_\lambda(x)) \leq A$ be valid for all $(x, y) \in X \times Y$ and for all $\lambda \in \Lambda$ and let h be the (finite) VC dimension. Then, the inequality*

$$P\left(\frac{R(\lambda) - \frac{1}{m}\sum_{i=1}^{m} l(y, g_\lambda(x))}{\sqrt{R(\lambda)}} > \varepsilon\right) < 4\exp\left(m\left(\frac{h}{m}\left(\ln\frac{2m}{h} + 1\right) - \frac{\varepsilon^2}{4A}\right)\right) \tag{2.9}$$

holds for all $\lambda \in \Lambda$ and all $\varepsilon > 0$.

Note that for large values of m this bound is nontrivial. For $\eta \in (0, 1)$ the **Capacity Term** is defined by

$$\mathcal{E}(m, h, \eta) := 4\frac{h\left(\ln\frac{2m}{h} + 1\right) - \ln\frac{\eta}{4}}{m}.$$

Then, by using (2.9) one obtains:

Corollary 2.16. *See [42, p. 193]. Let $A > 0$ with $A \in \mathbb{R}$ be given and let the set $\{l(\cdot, g_\lambda(\cdot)), \lambda \in \Lambda\}$ with $0 \leq l(y, g_\lambda(x)) \leq A$ for all $(x, y) \in X \times Y$, $\lambda \in \Lambda$ be given. Additionally, h denotes the corresponding VC dimension.*

i) Then, for $\eta \in (0, 1)$ one obtains the following result. With probability of at least $1 - \eta$ it holds that

$$R(\lambda) \leq R_{emp}(\lambda) + \frac{A\mathcal{E}(m, h, \eta)}{2} \left(1 + \sqrt{1 + \frac{4R_{emp}(\lambda)}{A\mathcal{E}(m, h, \eta)}}\right). \tag{2.10}$$

ii) Then, for $\eta \in (0, 0.5)$ one obtains the following result. With probability of at least $1 - 2\eta$ it holds that

$$R(\lambda^m) - R(\lambda^*) \leq A \left(\sqrt{\frac{-\ln \eta}{2m}} + \mathcal{E}(m, h, \eta) \left(1 + \sqrt{1 + \frac{4R_{emp}(\lambda^m)}{A\mathcal{E}(m, h, \eta)}}\right)\right). \tag{2.11}$$

2.2.3 Structural Risk Minimization

Now, the bounds presented in Corollary 2.16 are reviewed. Let the 0-1-loss be selected. Hence, $A = 1$ can be selected in Corollary 2.16. Here, it is further assumed that the VC dimension h is finite. Obviously, for any given decision function g_λ with $\lambda \in \Lambda$ the risk is smaller than 1. Let $R_{\text{emp}}(\lambda^m)$ be 0 for all m. Then, the right hand side of (2.10) is $\mathcal{E}(m, h, \eta)$ and the right hand side of (2.11) is larger than $2\mathcal{E}(m, h, \eta)$. Now, values of selected $\mathcal{E}(m, h, 0.05)$ are approximated.

m	h	$10h$	$36h$	$100h$	$1000h$
$\mathcal{E}(m, h, 0.05)$	≈ 24.3	≈ 3.1	≈ 1	≈ 0.4	≈ 0.05

For $m < 36h$ (2.10) and (2.11) are trivial. In practice h is usually much larger than $1{,}000$. Hence, around $1{,}000{,}000$ training points are needed to get useful bounds from (2.10) and (2.11). Often, one has less training points available. Regardless of this, such a large training set is computationally expensive. Hence, if one is interested in reasonable bounds for $R(\lambda)$ and $(R(\lambda^m) - R(\lambda^*))$ for small training sets, then another approach is needed. It is known that a smaller VC dimension leads to smaller bounds. Thus, the **Structural Risk Minimization** (SRM) method

does not only minimize the empirical risk, but also (a bound of) the VC dimension simultaneously.

It is assumed that there exists a structure within the set $S = \{l(\cdot, g_\lambda(\cdot))|\lambda \in \Lambda\}$. Let there be a sequence $\{\Lambda_i\}$ with $\Lambda_i \subseteq \Lambda$ for all $i \in \mathbb{N}$. Furthermore, let $S_i := \{l(\cdot, g_\lambda(\cdot))|\lambda \in \Lambda_i\}$ and $S^* := \bigcup_{i=1}^{\infty} S_i$ be defined. This structure should have the following properties:

1. $S_1 \subset S_2 \subset \ldots \subset S_p \subset \ldots$

2. For any i there is $A_i > 0$ such that $0 \leq l(y, g_\lambda(x)) \leq A_i$ for all $\lambda \in \Lambda_i$, $(x, y) \in X \times Y$.

3. Any set S_i has a finite VC dimension denoted by h_i.

4. The set S^* is dense in the set S in the $L_1(F)$ metric, that is to say for all $\delta > 0$ and any $l(\cdot, g_\lambda(\cdot)) \in S$ there exists a function $l(\cdot, g_{\bar{\lambda}}(\cdot)) \in S^*$ such that

$$\int_{X \times Y} |l(y, g_{\bar{\lambda}}(x)) - l(y, g_\lambda(x))| \, \mathrm{d}F_{X \times Y}(x, y) < \delta.$$

Obviously, if such a sequence exists, the following holds:

$$\begin{aligned} h_1 \leq h_2 \leq \ldots \leq h_p \leq \ldots \\ A_1 \leq A_2 \leq \ldots \leq A_p \leq \ldots \\ \mathcal{E}(m, h_1, \eta) \leq \mathcal{E}(m, h_2, \eta) \leq \ldots \leq \mathcal{E}(m, h_p, \eta) \leq \ldots. \end{aligned}$$

Let $g_{\lambda_p^m}$ denote the decision function that minimizes the empirical risk for a given training set of size m in the set of functions S_p. Then, (2.10) can be rewritten as

$$R(\lambda_p^m) \leq R_{\mathsf{emp}}(\lambda_p^m) + \frac{A_p \mathcal{E}(m, h_p, \eta)}{2} \left(1 + \sqrt{1 + \frac{4 R_{\mathsf{emp}}(\lambda_p^m)}{A_p \mathcal{E}(m, h_p, \eta)}}\right). \tag{2.12}$$

Now, for a given set of training points $(x_1, y_1), \ldots, (x_m, y_m)$ the SRM method minimizes the right hand side of (2.12) by choosing the element S_p of the structure for which the smallest right hand side can be achieved and determining a $\lambda_p^m \in \Lambda_p$ with

$$R_{\mathsf{emp}}(\lambda_p^m) = \min_{\lambda \in \Lambda_p} R_{\mathsf{emp}}(\lambda).$$

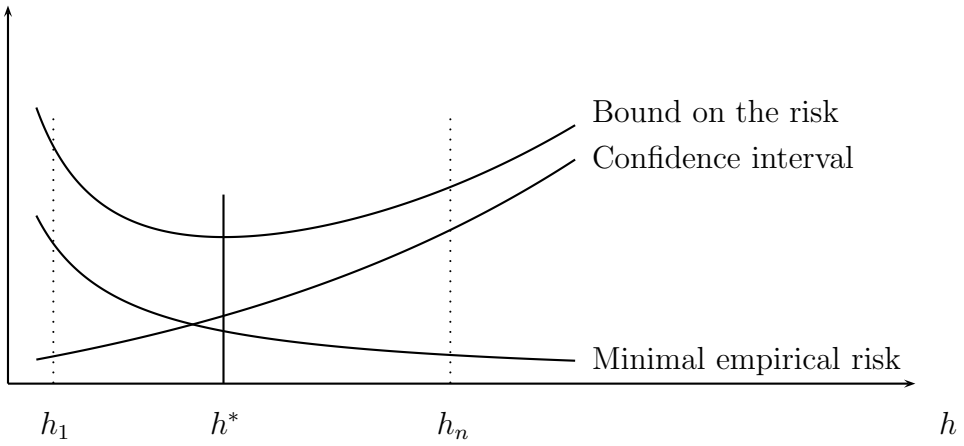


Figure 2.4: The right hand side of (2.12) is the sum of the empirical risk and the structural risk. If the VC dimension h is increased, then the capacity term increases while the minimal empirical risk decreases (see [42, p. 223]).

In literature the right hand side of (2.12) is decomposed into the Empirical Risk (the first summand) and the confidence interval or structural risk (the second summand) (see Figure 2.4).

One can show that the risks of the functions chosen according to the SRM method converge to the smallest possible risk for the set S with increasing number of observations. ([42, Subsections 6.3+6.4]).

Remark 2.17. *Section 2.3 introduces Support Vector Machines (SVMs). SVMs implement the SRM principle and the hinge loss is often chosen as loss function for SVMs. In general, the hinge loss is unbounded and the bound* (2.12) *would be trivial. In the [8] the authors proof in Lemma 23 that for SVMs the hinge loss is bounded if the input data are bounded (see Remark 2.14).*

2.2.4 Learning from Dependent Identically Distributed Data

So far i.i.d. data generating processes were considered (see the beginning of this section). Here the findings of Subsections 2.2.2 and 2.2.3 are extended to a more general class of processes.

Vidyagasar writes in [44, Section 2.5]: "...independence is a very restrictive concept, in several ways. First, it is often an assumption, rather than a deduction on the basis of observations. Second, it is an all or nothing property, in the sense that two random variables are either independent or they are not - the definition does not permit

an intermediate notion of being nearly independent." Furthermore, it is shown in this thesis that in a certain learning task dependent identically distributed data generating process occur. Notions of mixing processes are introduced which enables the handling of some situations of dependent identically distributed data generating processes. Additionally, the question could arise whether one can learn from non identically distributed data. This setting is not analyzed in this dissertation. The interested reader is referred to [39] as an anchor on this topic.

In literature there are several mixing conditions for stochastic processes. Here, three of them are considered, namely α-mixing, β-mixing, and ϕ-mixing. This subsection follows [44, Section 2.5] and first defines mixing coefficients and provides several properties of these coefficients. More information on this topic can be found in [10, 40].

Let Z be a stationary stochastic process, where $Z(t), t \in \mathbb{Z}$ is X valued. Here, X is equipped with a σ-algebra Σ of subsets of X. Furthermore, let the underlying probability space of the stochastic process Z be the canonical space $(X^\infty, \Sigma^\infty, P)$. Here, X^∞ is the Cartesian product $\Pi_{t=-\infty}^{\infty} X$, while Σ^∞ is the corresponding product σ-algebra and P is a shift invariant probability measure on $(X^\infty, \Sigma^\infty)$. For each index p, let $\Sigma_{-\infty}^{p}$ denote the σ-algebra generated by the coordinate random variables $Z(t), t \leq p$, and similarly let Σ_{p}^{∞} denote the σ-algebra generated by the coordinate random variables $Z(t), t \geq p$. Furthermore, let $\overline{\sum_{1}^{p-1}}$ denote the σ-algebra generated by the random variables $Z(t), t \leq 0$ as well as $Z(t), t \geq p$. Finally, let $P_{-\infty}^{p}$ and P_{p+1}^{∞} denote the marginal probability measures corresponding to $\Sigma_{-\infty}^{p}$ and Σ_{p+1}^{∞}, respectively. Then, by Kolmogorov Extension Theorem, there is a unique probability measure on $(X^\infty, \Sigma^\infty)$, denoted by $P_{-\infty}^{p} \times P_{p+1}^{\infty}$, with

1. $P(A) = \left(P_{-\infty}^{0} \times P_{1}^{\infty}\right)(A)$,
2. $P(B) = \left(P_{-\infty}^{0} \times P_{1}^{\infty}\right)(B)$,
3. $\left(P_{-\infty}^{0} \times P_{1}^{\infty}\right)(A \cap B) = \left(P_{-\infty}^{0} \times P_{1}^{\infty}\right)(A) \cdot \left(P_{-\infty}^{0} \times P_{1}^{\infty}\right)(B)$,

for all $A \in \Sigma_{-\infty}^{0}$ and $B \in \Sigma_{1}^{\infty}$.

Definition 2.18. *The α-**mixing coefficient** of the stochastic process Z is defined as*

$$\alpha(p) := \sup_{A \in \sum_{-\infty}^{0},\ B \in \sum_{p}^{\infty}} \left| P(A \cap B) - P(A)P(B) \right|.$$

*The β-**mixing coefficient** of the stochastic process is defined as*

$$\beta(p) := \sup_{C \in \sum_1^{p-1}} \left| P(C) - \left(P_{-\infty}^0 \times P_1^\infty \right)(C) \right|.$$

*The ϕ-**mixing coefficient** of the stochastic process is defined as*

$$\phi(p) := \sup_{A \in \sum_{-\infty}^0,\ B \in \sum_p^\infty} |P(B|A) - P(B)|.$$

Now, a few remarks are given and several properties of these coefficients are discussed.

- If $P(A) = 0$, then one sets the conditional probability $P(B|A) = P(B)$ in the definition of the ϕ-mixing coefficient.
- Since $\Sigma_{p+1}^\infty \subset \Sigma_p^\infty$ holds for all $p \in \mathbb{N}$, it is obvious that the α-, β-, and ϕ-mixing coefficients are all non-increasing. Thus,

$$\alpha(p+1) \leq \alpha(p),\ \beta(p+1) \leq \beta(p),\ \phi(p+1) \leq \phi(p),\ \text{for all } p \in \mathbb{N}.$$

- It can be shown that (see [10, 40, 44])

$$\alpha(p) \leq \beta(p) \leq \phi(p),\ \text{for all } p \geq 1. \tag{2.13}$$

- The stochastic process Z is i.i.d. if and only if $\phi(p) = 0$ for all $p \in \mathbb{N}$.

Definition 2.19. *The stochastic process Z is said to be α-**mixing** if $\alpha(p) \to 0$ as $p \to \infty$. The stochastic process Z is said to be β-**mixing** if $\beta(p) \to 0$ as $p \to \infty$. Finally, the stochastic process Z is said to be ϕ-**mixing** if $\phi(p) \to 0$ as $p \to \infty$.*

Note that due to (2.13) ϕ-mixing implies β-mixing and the latter implies α-mixing.

Some of the presented results in Subsections 2.2.2 and 2.2.3 for i.i.d. data generating processes can be extended to β- and ϕ-mixing data generating processes. To keep this introduction short only the following results are stated. Let a learning algorithm be working for an i.i.d. data generating process. In other words the difference between the minimal risk and the risk of the decision function returned by the learning algorithm becomes arbitrary small for a sufficiently large number of training points. Vidyagasar gives in [44] several conditions under which the same algorithm

works for a β-mixing data generating process. Roughly speaking, one can replace an i.i.d. by an β-mixing data generating process in a learning algorithm when increasing numbers of training points improve the performance of this algorithm. This holds for the ERM principle as the capacity term $\mathcal{E}(m, h, \eta)$ decreases when the number of training points m increases. However, there are some drawbacks if a mixing data generating process is used. In [29, 44, 49] several bounds on the risk can be found for this setting. Many more technical details would be necessary to derive those bounds. Therefore, those bounds are not presented here. In general, one gets weaker bounds for a mixing data generating process than for an i.i.d. data generating process when the same number of training points is used.

2.3 Support Vector Machines

This section presents a learning method which is based on the Structural Risk Minimization (SRM) principle which was presented in Subsection 2.2.3. The basic idea of this method is to separate the data using a hyperplane. In this section it is assumed that $Y = \{-1, 1\}$.

Theorem 2.20. *[42, Theorem 10.3] Let the set*

$$\Lambda := \{(w, b) \in \mathbb{R}^n \times \mathbb{R} | \langle w, w \rangle = 1\}$$

be defined. Then, let the set of functions $\{g_{w,b} : X \to \mathbb{R} | (w, b) \in \Lambda\}$ *be defined for a given* $\Delta > 0$ *by*

$$g_{w,b}(x) := \begin{cases} 1 & \text{if } \langle w, x \rangle + b \geq \Delta \\ -1 & \text{if } \langle w, x \rangle + b \leq -\Delta \\ 0 & \text{else.} \end{cases}$$

Now, let $X \subseteq \mathbb{R}^n$ *hold and for some* $D > 0$ *let* $\langle x, x \rangle \leq D^2$ *hold for all* $x \in X$. *Then, the VC dimension* h *of this set of functions is bounded by*

$$h \leq \min \left\{ \frac{D^2}{\Delta^2}, n \right\} + 1. \tag{2.14}$$

In this theorem all points $x \in X$ with $|g_{w,b}(x)| < \Delta$ lead to a loss of 1 if the 0-1-loss is used. Hence, there is a **margin** between the positive and negative labeled input points if the empirical risk is zero. The width of this margin is 2Δ.

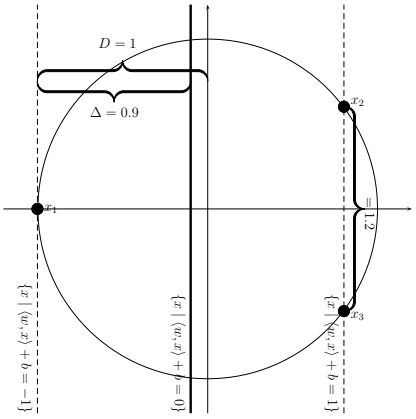


Figure 2.5: In this example all training points belong to the unit ball. According to Theorem 2.20 one wants to classify with hyperplanes which have a margin larger than $2\Delta = 1.8$. Now, three points which can be classified correctly independent of their label should be found. W.l.o.g. it is assumed that x_1 belongs to the positive class. Then, the maximal distance between x_2 and x_3 is 1.2 in this setting. Hence, one cannot label x_2 and x_3 arbitrarily. Thus, the VC dimension is 2 instead of 3.

If one assumes that the set X is bounded by D, then the bound (2.14) on the VC dimension is minimized when the margin 2Δ between the positive and negative training points is maximized. Let a set of training points $\{(x_i, y_i)\}, i \in M := \{1, \ldots, m\}$ be considered. Then, the following optimization problems are equivalent in some sense.

$$\max_{\Delta, w, b} \Delta \quad \text{s.t. } y_i \left(\langle w, x_i \rangle + b\right) \geq \Delta \text{ for all } i \in M;\ \langle w, w \rangle = 1;\ \Delta > 0$$

$$\max_{\Delta, w, b} \Delta \quad \text{s.t. } y_i \left(\left\langle \frac{w}{\Delta}, x_i \right\rangle + \frac{b}{\Delta}\right) \geq 1 \text{ for all } i \in M;\ \langle w, w \rangle = 1;\ \Delta > 0$$

$$\text{set } \tilde{w} := \frac{w}{\Delta};\ \tilde{b} := \frac{b}{\Delta}$$

$$\max_{\Delta, \tilde{w}, \tilde{b}} \Delta \quad \text{s.t. } y_i \left(\langle \tilde{w}, x_i \rangle + \tilde{b}\right) \geq 1 \text{ for all } i \in M;\ \Delta^2 \langle \tilde{w}, \tilde{w} \rangle = 1;\ \Delta > 0$$

$$\text{set } \Delta := \sqrt{\frac{1}{\langle \tilde{w}, \tilde{w} \rangle}}$$

$$\min_{\tilde{w}, \tilde{b}} \langle \tilde{w}, \tilde{w} \rangle \quad \text{s.t. } y_i \left(\langle \tilde{w}, x_i \rangle + \tilde{b}\right) \geq 1 \text{ for all } i \in M$$

with $w, \tilde{w} \in \mathbb{R}^n$ and $b, \tilde{b} \in \mathbb{R}$. Hence, by minimizing $\langle \tilde{w}, \tilde{w} \rangle$ one maximizes the margin Δ and therefore minimizes the VC dimension. Note that w is written for $\tilde{w}$ and b is written for $\tilde{b}$ from now on.

According to the SRM principle one wants to minimize the bound on the risk (2.12) by minimizing the capacity term $\mathcal{E}(m, h, \eta)$ and the empirical risk $R_{\text{emp}}(\lambda)$ (see Figure 2.4). In particular, to minimize the capacity term (structural risk) one can minimize the VC dimension. It was discussed that for a small $\langle w, w \rangle$ a small VC

dimension is expected. The empirical risk (and its minimization) is based on some loss function. The hinge loss (2.2) is often chosen as loss function for SVMs.

Hence, **Support Vector Machines** (SVMs) are based on the following optimization problem

$$\min_{w,b} \ \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{m}(1 - y_i(\langle w, x_i\rangle + b))_+ . \tag{2.15}$$

The parameter $C > 0$ is called **Penalty Parameter**. A large C emphasizes the minimization of the empirical risk while a small C emphasizes the minimization of the structural risk (capacity term).

Let (w^*, b^*) be a solution of (2.15). Then, an SVM leads to the decision function

$$g(x) := \langle w^*, x\rangle + b^*,$$

and uses $\operatorname{sgn}(g(x))$ as **classification rule**. Note that a positive training point (x, y) with $g(x) \in [0, 1)$ would be classified correctly, but already increases the empirical risk. The same applies for a negative point (x, y) with $g(x) \in (-1, 0)$.

2.3.1 Reformulated Optimization Problems

Problem (2.15) is an unconstrained non-differentiable optimization problem. This motivates the following reformulated optimization problem which is constrained but differentiable.

$$\begin{aligned} \min_{w,b,\xi} \ & \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{m}\xi_i \\ \text{s.t. } & y_i\left(\langle w, x_i\rangle + b\right) \geq 1 - \xi_i \quad \text{for all } i \in M \\ & \xi_i \geq 0 \quad \text{for all } i \in M. \end{aligned} \tag{2.16}$$

It can be shown easily that if (w^*, b^*) is a solution of (2.15) then there exists ξ^* such that (w^*, b^*, ξ^*) is a solution of (2.16). Vice versa, if (w^*, b^*, ξ^*) is a solution of (2.16) then (w^*, b^*) is a solution of (2.15). Problem (2.16) is called **linear primal SVM**.

The following assertion can be found in literature. However, to the best of my knowledge, there does not exist a formal proof for it.

Lemma 2.21. *Optimization problem* (2.16) *always has a solution.*

Proof. If there are only positive training points or only negative training points, then $(w^*, b^*, \xi^*) = (0, 1, 0)$ ord $(w^*, b^*, \xi^*) = (0, -1, 0)$ is a solution, respectively. Thus, it is assumed that there is at least one positive and one negative training point. Now, the point $(\bar{w}, \bar{b}, \bar{\xi})$ with $\bar{w} = 0$, $\bar{b} = 0$, $\bar{\xi}_i = 1$ for all $i \in M$ is a feasible point. Therefore, any solution (w^*, b^*, ξ^*) must satisfy

$$0 \leq \xi_i^* \leq Cm \text{ for all } i \in M \text{ and } \|w^*\| \leq \sqrt{2Cm}.$$

Furthermore, by setting $D := \max_{i \in M}\{\|x_i\|\}$ one obtains

$$\begin{aligned} b^* &\geq 1 - \xi_i^* - \langle w^*, x_i \rangle \geq -Cm - D\sqrt{2Cm} && \text{for } y_i = 1 \\ b^* &\leq \xi_i^* - 1 - \langle w^*, x_i \rangle \leq Cm + D\sqrt{2Cm} && \text{for } y_i = -1. \end{aligned}$$

Hence, one can add the following constraints to (2.16) without changing the solution set.

$$\begin{aligned} 0 \leq \xi_i &\leq Cm \quad \text{for all } i \in M \\ \langle w, w \rangle &\leq 2Cm \\ |b| &\leq Cm + D\sqrt{2Cm} \end{aligned}$$

Then, the Weierstrass-Theorem can be applied to (2.16) if those constraints are added. □

According to Corollary 2.7 any (w^*, b^*, ξ^*) is a solution of (2.16) if and only if there exist Lagrange Multipliers $(u^*, v^*) \in \mathbb{R}^m \times \mathbb{R}^m$ such that $(w^*, b^*, \xi^*, u^*, v^*)$ is a KKT Point of (2.16). The Lagrange Function of (2.16) is

$$\mathcal{L}(w, b, \xi, u, v) = \frac{1}{2}\langle w, w \rangle + C\sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} u_i \left(y_i \left(\langle w, x_i \rangle + b\right) - 1 + \xi_i\right) - \sum_{i=1}^{m} \xi_i v_i.$$

The KKT Conditions of (2.16) are

$$\begin{aligned} w = \sum_{i=1}^{m} u_i y_i x_i \qquad & 0 = \sum_{i=1}^{m} y_i u_i \\ C - u_i - v_i = 0 \text{ for all } i \in M \qquad & \xi_i \geq 0 \text{ for all } i \in M \\ 1 - \xi_i - y_i \left(\langle w, x_i \rangle + b\right) \leq 0 \text{ for all } i \in M \qquad & u_i \geq 0 \text{ for all } i \in M \\ v_i \geq 0 \text{ for all } i \in M \qquad & v_i \xi_i = 0 \text{ for all } i \in M \\ u_i \left(1 - \xi_i - y_i \left(\langle w, x_i \rangle + b\right)\right) = 0 \text{ for all } i \in M. & \end{aligned} \tag{2.17}$$

Now, another optimization problem is presented which is connected by duality theory with problem (2.16).

$$\begin{aligned} \min_{\alpha} \ & \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle - \sum_{i=1}^{m} \alpha_i \\ \text{s.t. } & 0 \leq \alpha_i \leq C \text{ for all } i \in M \\ & \sum_{i=1}^{m} \alpha_i y_i = 0. \end{aligned} \tag{2.18}$$

Again, by using Corollary 2.7, α^* is a solution of (2.18) if and only if there are Lagrange Multipliers $(\zeta^*, \eta^*, \theta^*) \in \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}$ such that $(\alpha^*, \zeta^*, \eta^*, \theta^*)$ is a KKT Point.

The Lagrange Function of (2.18) is

$$\mathcal{L}(\alpha, \zeta, \eta, \theta) = \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle - \sum_{i=1}^{m} \alpha_i - \sum_{i=1}^{m} \zeta_i \alpha_i - \sum_{i=1}^{m} \eta_i (C - \alpha_i) + \theta \sum_{i=1}^{m} \alpha_i y_i.$$

The corresponding KKT Conditions are

$$\begin{aligned} & \sum_{j=1}^{m} \alpha_j y_i y_j \langle x_i, x_j \rangle - 1 - \zeta_i + \eta_i + \theta y_i = 0 \text{ for all } i \in M \\ & \alpha_i \geq 0 \text{ for all } i \in M \qquad \alpha_i \leq C \text{ for all } i \in M \\ & \zeta_i \geq 0 \text{ for all } i \in M \qquad \eta_i \geq 0 \text{ for all } i \in M \\ & \zeta_i \alpha_i = 0 \text{ for all } i \in M \qquad \eta_i (C - \alpha_i) = 0 \text{ for all } i \in M \\ & \sum_{i=1}^{m} y_i \alpha_i = 0 \end{aligned} \tag{2.19}$$

The following result is well known. However, the provided proof is new to the best of my knowledge.

Lemma 2.22. *Problem* (2.16) *has a solution if and only if problem* (2.18) *has a solution.*

Proof. Let (w^*, b^*, ξ^*) be a solution of (2.16). Then, there are Lagrange Multipliers u^*, v^* such that $(w^*, b^*, \xi^*, u^*, v^*)$ is a corresponding KKT Point. Now, one sets

$$\alpha^* := u^* \quad \eta^* := \xi^* \quad \theta^* := b^* \quad \zeta_i^* := y_i \left(\langle w^*, x_i \rangle + b^* \right) + \xi_i^* - 1 \text{ for all } i \in M.$$

Now, by using (2.17) one gets that $(\alpha^*, \zeta^*, \eta^*, \theta^*)$ solves (2.19). Thus, problem (2.18) has a solution if (2.16) has a solution.

Now, it is assumed that (2.18) has a solution and that $(\alpha^*, \zeta^*, \eta^*, \theta^*)$ is a corresponding KKT Point. By setting

$$u^* := \alpha^* \quad \xi^* := \eta^* \quad b^* := \theta^* \quad w^* := \sum_{i=1}^{m} \alpha_i^* y_i x_i \quad v_i^* := C - \alpha_i^* \text{ for all } i \in M$$

and using (2.19) one gets that $(w^*, b^*, \xi^*, u^*, v^*)$ solves (2.17). Thus problem (2.16) has a solution if (2.18) has a solution. □

Remark 2.23. *Lemmas 2.21 and 2.22 imply that problem* (2.18) *always has a solution.*

Problem (2.18) is called **dual linear SVM**. Obviously, problem (2.16) and problem (2.18) are quadratic programs. Problem (2.16) has $n + m + 1$ variables and $2m$ inequalities where m are box constraints. Problem (2.18) has m variables, $2m$ box constraints, and one linear equality. Note that for many algorithms it is easier to handle box constraints than (general) linear inequalities.

Remark 2.24. *Additionally, the last proof provides under certain assumptions a way to obtain* (w^*, b^*) *by a solution* α^**, in other words one does not need the Lagrange Multipliers of* (2.18) *to obtain* w^* *and* b^**. By using the first equality of* (2.17) *one can always obtain* w^**. Additionally, using* (2.17) *one obtains:* $0 < \alpha_i^* = u_i^* < C \Rightarrow\ v_i^* > 0 \Rightarrow \xi_i^* = 0$ *Then, by* $u_i^* > 0$ *and* $\xi_i^* = 0$ *it follows that* $1 - y_i(\langle w^*, x_i \rangle + b^*) = 0$*. Hence,* $b^* = y_i - \langle w^*, x_i \rangle$*. Therefore, for any solution* α^* *with* $0 < \alpha_i^* < C$ *for some* $i \in M$ *one can easily obtain* (w^*, b^*)*.*

The following assertion is well known but is sometimes wrongly stated by claiming that the solution of (2.16) is unique. To the best of my knowledge there does not exist a proof for this assertion. A similar assertion can be found in [42, Theorem 10.1].

Theorem 2.25. *Let* (w^*, b^*, ξ^*) *and* $(w^{**}, b^{**}, \xi^{**})$ *be solutions of* (2.16)*. Then,* $w^* = w^{**}$*.*

Proof. According to Remark 2.2 $(\bar{w}, \bar{b}, \bar{\xi}) := \frac{1}{2}(w^*, b^*, \xi^*) + \frac{1}{2}(w^{**}, b^{**}, \xi^{**})$ is also a feasible point and a solution. Now, it is assumed that $w^* \neq w^{**}$. Then,

$$0 < \langle w^* - w^{**}, w^* - w^{**} \rangle = \langle w^*, w^* \rangle + \langle w^{**}, w^{**} \rangle - 2\langle w^*, w^{**} \rangle$$

implies

$$2\langle w^*, w^{**} \rangle < \langle w^*, w^* \rangle + \langle w^{**}, w^{**} \rangle.$$

Now, one gets

$$\begin{aligned}\frac{1}{2}\langle\bar{w},\bar{w}\rangle + C\sum_{i=1}^{m}\bar{\xi}_i =& \frac{1}{8}\langle w^* + w^{**}, w^* + w^{**}\rangle + \frac{C}{2}\sum_{i=1}^{m}(\xi_i^* + \xi_i^{**}) \\ =& \frac{1}{8}\langle w^*, w^*\rangle + \frac{2}{8}\langle w^*, w^{**}\rangle + \frac{1}{8}\langle w^{**}, w^{**}\rangle + \frac{C}{2}\sum_{i=1}^{m}(\xi_i^* + \xi_i^{**}) \\ <& \frac{1}{2}\left(\frac{1}{2}\langle w^*, w^*\rangle + C\sum_{i=1}^{m}\xi_i^*\right) + \frac{1}{2}\left(\frac{1}{2}\langle w^{**}, w^{**}\rangle + C\sum_{i=1}^{m}\xi_i^{**}\right) \\ =& \frac{1}{2}\langle w^*, w^*\rangle + C\sum_{i=1}^{m}\xi_i^*\end{aligned}$$

This is a contradiction to the optimality of w^*. Thus, $w^* = w^{**}$. □

The following example shows that b^* and ξ^* may not be unique. Let problem (2.16) with

$$\begin{aligned}&m = 4, C = 0.25, y = (-1,-1,1,1)^\top, \text{ and} \\ &x_1 = (0,1)^\top, x_2 = (1,0)^\top, x_3 = (1,2)^\top, x_4 = (2,1)^\top\end{aligned}$$

be considered. Then, for any $\lambda \in [0,1]$

$$(w^*, b^*(\lambda), \xi^*(\lambda)) = \left(\begin{pmatrix}0.5\\0.5\end{pmatrix}, -1.5+\lambda, \begin{pmatrix}\lambda\\ \lambda\\ 1-\lambda\\ 1-\lambda\end{pmatrix}\right)$$

is a solution.

Note that the solution α^* of the dual linear SVM might not by unique, although w^* is uniquely determined and it holds that

$$w^* = \sum_{i=1}^{m}\alpha_i^* y_i x_i.$$

2.3.2 Nonlinear SVMs

In the last subsection linear SVMs which separate the training points by a hyperplane in $\mathbb{R}^n$ were introduced. In many cases this will lead to a bad performance of the classifier since the classes cannot be separated by a hyperplane (see Figure 2.6).

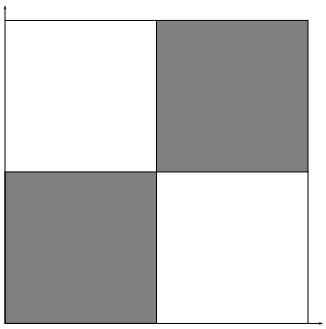


Figure 2.6: All positive input points belong to the white squares while all negative input points belong to the black squares. Obviously, any linear classifier has a bad performance.

In order to overcome this issue linear SVMs can be extended to nonlinear SVMs by the following idea. Let ψ be a mapping with $\psi : X \to \mathcal{H}$ where $\mathcal{H}$ is a real Hilbert space. Now, problem (2.15) is extended to

$$\min_{w,b} \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{m} \left(y_i \left(\langle w, \psi(x_i)\rangle + b\right) - 1\right)_+ \tag{2.20}$$

with $(w, b) \in \mathcal{H} \times \mathbb{R}$. Again, this problem is reformulated to

$$\begin{aligned} \min_{w,b,\xi} \ & \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{m} \xi_i \\ \text{s.t. } & y_i \left(\langle w, \psi(x_i)\rangle + b\right) \geq 1 - \xi_i \quad \text{for all } i \in M \\ & \xi_i \geq 0 \quad \text{for all } i \in M \end{aligned} \tag{2.21}$$

which is called primal SVM. The difference between problems (2.16) and (2.21) is that in the latter w is element of an arbitrary real Hilbert space $\mathcal{H}$ while in the first problem w belongs to $\mathbb{R}^n$. Note that one can generalize Theorem 2.25. This means that any solution of (2.21) has the same $w^* \in \mathcal{H}$. Now, let $(w, b) \in \mathcal{H} \times \mathbb{R}$ be given and let the set

$$\{x \in \mathbb{R}^n \mid \langle w, \psi(x)\rangle + b = 0\}$$

be considered. This set is a hyperplane in $\mathcal{H}$ but it will not be a hyperplane in $\mathbb{R}^n$ in general. Hence, by choosing an appropriate mapping ψ one can obtain other classifiers than those defined by a hyperplane in $\mathbb{R}^n$.

For the moment it is assumed that $\mathcal{H}$ has finite dimension. Then, similar as in

Subsection 2.3.1 one obtains the following optimization problem

$$\begin{aligned} \min_{\alpha} \ & \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \psi(x_i), \psi(x_j) \rangle - \sum_{i=1}^{m} \alpha_i \\ \text{s.t. } & 0 \leq \alpha_i \leq C \text{ for all } i \in M \\ & \sum_{i=1}^{m} \alpha_i y_i = 0. \end{aligned} \tag{2.22}$$

Now, Lemma 2.22 can be extended to connect (2.21) and (2.22). However, if $\mathcal{H}$ has infinite dimension, then Lemma 2.22 cannot be applied directly.

Theorem 2.26 (Semiparametric Representer Theorem). *Let (w^*, b^*) denote a solution of problem* (2.20)*. Then, there is $\mu^* \in \mathbb{R}^m$ such that*

$$w^* = \sum_{i=1}^{m} \mu_i^* \psi(x_i).$$

Proof. See [38, Theorem 4.3]. □

Using this theorem one can reformulate (2.21) to

$$\begin{aligned} \min_{\mu,\xi,b} & \frac{1}{2} \sum_{i,j=1}^{m} \mu_i \mu_j \left\langle \psi(x_i), \psi(x_j) \right\rangle + C \sum_{i=1}^{m} \xi_i \\ \text{s.t. } & y_i \left(\sum_{j=1}^{m} \mu_j \left\langle \psi(x_j), \psi(x_i) \right\rangle + b \right) \geq 1 - \xi_i \text{ for all } i \in M \\ & \xi_i \geq 0 \text{ for all } i \in M \end{aligned} \tag{2.23}$$

Now, by comparing the KKT Systems of (2.22) and (2.23) one can solve (2.22) to obtain a solution of (2.23) and vice versa. Then, by using Theorem 2.26 a solution of (2.21) can be obtained.

Often, it is computationally very expensive or even impossible to evaluate the mapping ψ explicitly. Therefore, by a closer look of problem (2.22) or (2.23) it follows that the mapping ψ is used only within scalar products. Thus, one can define **kernels** $k : X \times X \to \mathbb{R}$ as

$$k(s,t) := \langle \psi(s), \psi(t) \rangle \text{ for all } (s,t) \in X \times X. \tag{2.24}$$

Formally speaking, a function $k : X \times X \to \mathbb{R}$ is called kernel function or just kernel if there exist some Hilbert space $\mathcal{H}$ and some mapping $\psi : X \to \mathcal{H}$ so that (2.24) holds.

Note that there exist kernels which can be evaluated without explicitly computing or even knowing the function ψ. For example, by

$$k(s,t) := \langle s,t \rangle^p \,,\ p \in \mathbb{N}$$

a kernel function $k : X \times X \to \mathbb{R}$ is defined for any $p \in \mathbb{N}$. The corresponding ψ can be found in [15, p. 31].

The method of replacing a linear scalar product by a kernel function is often called **kernel trick**.

Using kernels, (2.22) can be written as

$$\begin{aligned} \min_{\alpha} \ & \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j k\left(x_i, x_j\right) - \sum_{i=1}^{m} \alpha_i \\ \text{s.t. } & 0 \le \alpha_i \le C \text{ for all } i \in M \\ & \sum_{i=1}^{m} \alpha_i y_i = 0. \end{aligned} \tag{2.25}$$

Problem (2.25) is called **dual SVM** or just **SVM**. Furthermore, any x_i with $\alpha_i > 0$ is called **Support Vector** (SV).

Note that the decision function $g : X \to \mathbb{R}$ obtained by an SVM can be written as

$$g(x) := \sum_{i=1}^{m} \alpha_i y_i k(x_i, x) + b^*.$$

Here, b^* is obtained by a solution of the primal SVM (see also Remark 2.24).

Let the matrix $G \in \mathbb{R}^{m \times m}$ be defined by $G_{ij} := k(x_i, x_j)$. Then, G is a Gram matrix. It is well known that any Gram matrix is positive semidefinite. This yields that (2.25) has a quadratic target function which is convex. Hence, (2.25) is a convex optimization problem.

In the following, several possibilities to construct new kernels are presented.

Proposition 2.27. *Let $k_1 : X \times X \to \mathbb{R}$ and $k_2 : X \times X \to \mathbb{R}$ be kernel functions*

and let $x_i, x_j \in X$ be given. Then, the functions

$$
\begin{aligned}
&k(x_i, x_j) = k_1(x_i, x_j) + k_2(x_i, x_j) \\
&k(x_i, x_j) = c \cdot k_1(x_i, x_j) \text{ for all } c > 0 \\
&k(x_i, x_j) = k_1(x_i, x_j) + c \text{ for all } c > 0 \\
&k(x_i, x_j) = k_1(x_i, x_j) \cdot k_2(x_i, x_j) \\
&k(x_i, x_j) = f(x_i) \cdot f(x_j) \text{ for any continuous function } f : X \to \mathbb{R}
\end{aligned}
$$

are kernel functions, too.

Proof. See [22, Theorem 2.20]. □

Corollary 2.28. *Let $k_1 : X \times X \to \mathbb{R}$ be a kernel function and let $x_i, x_j \in X$ be given. Then, the functions*

$$
\begin{aligned}
&k(x_i, x_j) = (k_1(x_i, x_j) + c)^d, \ c > 0, d \in \mathbb{N} \\
&k(x_i, x_j) = \exp\left(\frac{k_1(x_i, x_j)}{\sigma^2}\right) \text{ for all } \sigma > 0 \\
&k(x_i, x_j) = \exp\left(-\frac{k_1(x_i, x_i) - 2k_1(x_i, x_j) + k_1(x_j, x_j)}{\sigma^2}\right) \text{ for all } \sigma > 0 \\
&k(x_i, x_j) = \frac{k_1(x_i, x_j)}{\sqrt{k_1(x_i, x_i) \cdot k_1(x_j, x_j)}}
\end{aligned}
$$

are kernel functions, too.

Proof. See [22, Corollary 2.21]. □

Remark 2.29. *Let $x_i, x_j \in X \subseteq \mathbb{R}^n$ be given. Then,*

- *$k(x_i, x_j) := \langle x_i, x_j \rangle$ is called **linear kernel**.*
- *$k(x_i, x_j) := (\langle x_i, x_j \rangle + c)^d$ is called **polynomial kernel of degree** d. Moreover, if $c = 0$ then it is called **homogeneous** polynomial kernel of degree d.*
- *$k(x_i, x_j) := \exp\left(-\sigma^2 \|x_i - x_j\|^2\right)$ for $\sigma > 0$ is called **radial basis function (RBF) kernel**.*

Remark 2.30. *Many numerical methods for efficiently solving SVMs have been proposed. As an anchor to this topic the reader is referred to the papers [17, 27, 31, 33]. The numerical results presented later on are based on a modified version of the LIBSVM library [13]. Those modifications are presented in Chapter 5. The LIBSVM library solves the dual SVM to obtain a decision function.*

2.4 Bayes' Rule of Classification

Up to now it was assumed that the data generating distribution function $F_{X\times Y}$ is unknown. In this section, it is assumed that it is known and that there is a corresponding density function $p_{X,Y}$ which is called joint density function of X and Y. Under those assumptions the **Bayes' Rule of Classification** is known and can be used as decision function. Note that the Bayes' Rule of Classification is also called **Bayes' Classifier**. In this dissertation, any classifier which leads to the same risk as the Bayes' Rule of Classification for a given loss function is also called Bayes' Classifier.

It is well known that a Bayes' Classifier is optimal. In other words for a given pattern recognition problem there is no other classifier with a smaller risk than the Bayes' Classifier [16]. Often, the Bayes' Classifier is only derived for a discrete density function p_X. Here, it is provided for a continuous density function p_X.

Let p_X and p_Y be the marginal probability density functions of X and Y, respectively. The marginal probability density function of X is

$$p_X(x) := \int_Y p_{X,Y}(x,y)\,\mathrm{d}y,$$

while the marginal probability density function of Y is

$$p_Y(y) := \int_X p_{X,Y}(x,y)\,\mathrm{d}x.$$

Then, using Bayes' Theorem for probability densities, one gets the a posterior probability density function of Y given $X = x$ by

$$p_Y(y|X=x) = \frac{p_{X,Y}(x,y)}{p_X(x)} = \frac{p_X\left(x|Y=y\right)p_Y(y)}{p_X(x)}.$$

Hence, the density $p_Y(y|X=x)$ can be evaluated for any $(x,y) \in X \times Y$. As Y is discrete, p_Y is a discrete density function. Thus, $\sum_{y\in Y} p_Y(y|X=x) = 1$ for all x and

$$P(Y=y|X=x) = p_Y(y|X=x).$$

Now, it is shown how the Bayes' Classifier can be obtained. Firstly, let the 0-1-loss be selected. Obviously, to minimize the risk, one has to classify any $x \in X$ to the class with largest probability $P(y|X=x)$. For instance, if $Y = \{-1,1\}$, then one

classifies x to the positive class if and only if

$$P(Y = 1|X = x) > P(Y = -1|X = x),$$

which is equivalent to $P(Y = 1|X = x) > 0.5$.

Secondly, for $Y = \{-1, 1\}$ a way is presented how to obtain a Bayes' Classifier which is based on the **weighted loss function**

$$l_d(y, \bar{y}) := \begin{cases} d & \text{if } y = 1 \text{ and } \bar{y} = -1, \\ 1 & \text{if } y = -1 \text{ and } \bar{y} = 1, \\ 0 & \text{else} \end{cases} \tag{2.26}$$

for some $d > 0$. Here, y is the correct class while $\bar{y}$ is the predicted class.

Let x be given input point. Then, the expected loss l_d is

- $l_d(-1, 1)P(Y = -1|X = x) = 1 \cdot P(Y = -1|X = x)$ if x is classified to 1
- $l_d(1, -1)P(Y = 1|X = x) = d \cdot P(Y = 1|X = x)$ otherwise.

Hence, to minimize the loss x is classified to the positive class if an only if

$$d \cdot P(Y = 1|X = x) > P(Y = -1|X = x).$$

Remark 2.31. *In [28] Lin presents an interesting connection between SVMs and the Bayes' Rule in Classification. It is shown, that the classification rules derived by the decision function of SVMs converge for large numbers of training points to the Bayes' Rule of classification if the data generating process is i.i.d.*

2.5 Quality of a Classifier

In the last subsections different techniques to obtain a decision function were described. Let g_1 and g_2 be decision functions for a given binary pattern recognition task. The question arises which of the given functions is more suited for the task, in other words which one has the smaller risk (2.3). In the last subsections bounds for the risk were presented. However, those bounds are in many cases computational expensive, trivial, or weak. Therefore, other techniques to measure the performance of a classifier will be introduced.

Here, let a training set $\{(x_i, y_i)\}$ with $i \in \{1, \ldots, m\}$ be given for a given pattern recognition task. Then, one can split up this training set into two disjoint sets, where the set $\{(x_i, y_i)\}_{i=1}^{m_1}$ is still called training set and the set $\{(x_i, y_i)\}_{i=m_1+1}^{m}$ is called test set. Let g_1 and g_2 be obtained by two different learning methods which use $\{(x_i, y_i)\}_{i=1}^{m_1}$ as training set. Then, g_1 is said to be better than g_2 if and only if

$$\frac{1}{m - m_1} \sum_{i=m_1+1}^{m} l(y_i, g_1(x_i)) < \frac{1}{m - m_1} \sum_{i=m_1+1}^{m} l(y_i, g_2(x_i))$$

for a given loss function l. Note that this loss function and the loss function used by the learning method may be different. Hence, one uses the empirical risk of the test set as measure for the quality of a decision function.

However, using the method mentioned above could lead to an overfitting on the given test set. In other words, the selected model leads to very good result for the given test set, but leads to bad results for other test sets. In order to overcome this issue one could use cross validation or the leave one out method [38]. Another possibility would be to split the training set $\{(x_i, y_i)\}$ with $i \in \{1, \ldots, m\}$ into three sets. Then, $\{(x_i, y_i)\}_{i=1}^{m_1}$ is still called training set, while $\{(x_i, y_i)\}_{i=m_1+1}^{m_2}$ is still called test set. Now, $\{(x_i, y_i)\}_{i=m_2+1}^{m}$ is called validation set. For training, only the training set is used. Now, the model selection (for SVMs the selection of the used kernel k and penalty parameter C) is done by using the training set and minimizing the empirical risk on the test set. Finally, one says that the decision function g_1 is better than g_2 if and only if g_1 leads to a smaller empirical risk than g_2 on the validation set.

3 Pairwise Classification

Many machine learning algorithms are based on binary classifiers, in other words they can only be applied if $Y = \{-1, 1\}$. To extend binary classifiers to multiclass classifiers many methods have been suggested, for example the One Against One technique, the One Against All technique, Directed Acyclic Graphs or Multiclass SVMs. Further information on this topic can be found in [36].

Now, let Y be a finite subset of $\mathbb{N}$ and let a training set $\{(x_i, y_i)\}_{i=1}^{m_1}$ and a test set $\{(x_i, y_i)\}_{i=m_1+1}^{m}$ be given. Additionally, it is assumed that $k \in \{m_1 + 1, \ldots, m\}$ exists such that $y_k \neq y_i$ for all $i \in \{1, \ldots, m_1\}$. In other words, there is at least one point in the test set whose class is not represented by any point in the training set. Under this assumption none of the techniques mentioned above will provide meaningful results, which means that they either classify x_k to any of the classes represented in the training set, or they may reject the classification of x_k.

For instance, the described setting occurs in face recognition. If one wants to provide an access control base on face recognition it would be very impractical if one uses any of the techniques mentioned above. For instance, if one person should be added to the database, then a new training would be necessary.

In this work a learning method which is able to handle classes not represented in the training set is analyzed. The generalization to never seen classes is called **interclass generalization**. In literature, interclass generalization is also called interclass transfer or learning to learn [4].

In order to handle interclass generalization it is proposed in this dissertation to use the pairwise classification technique [7, 23, 24, 4, 6, 32, 43]. Note that pairwise classification is a special case of collaborative filtering [1]. Now, to introduce the pairwise classification technique let an ordinary training set

$$\left\{\left(x_i, y_i^{cl}\right)\right\}_{i=1}^{m} \subseteq X \times Y_{cl} \tag{3.1}$$

be given. If not stated otherwise, it is assumed that c is the number of existing classes and that $Y_{cl} := \{1, \ldots, c\}$. Then, instead of using such a training set for

predicting the class $y^{cl} \in Y_{cl}$ for any $x \in X$ a pairwise training set is used. In order to define such a training set, firstly set

$$y_{ij} := \begin{cases} 1 & \text{if } y_i^{cl} = y_j^{cl} \\ -1 & \text{otherwise} \end{cases}$$

for any $i, j \in M := \{1, \ldots, m\}$. In other words, y_{ij} is set to 1 if x_i and x_j belong to the same class and y_{ij} is set to -1 if x_i and x_j do not belong to the same class. Secondly, a set of the following form is used as pairwise training set

$$\{((x_i, x_j), y_{ij})\}_{(i,j)\in I} \subseteq (X \times X) \times Y \tag{3.2}$$

with $Y = \{-1, 1\}$ and $I \subseteq M \times M$. In pairwise classification $u \in X$ is called **example** and $(u, v) \in X \times X$ is called **pair of examples** or just **pair**. Furthermore, $(u, v) \in X \times X$ is called positive (negative) pair if the corresponding $y = 1$ $(y = -1)$. Let a set of the form (3.1) be given. Then, the set $\{x_i\}_{i=1}^m \subseteq X$ is called **set of training examples**. Additionally, the set $\{y_i^{cl}\}_{i=1}^m \subseteq Y^{cl}$ is called **set of training classes**. Finally, a **pairwise test set**, a **set of test examples**, and a **set of test classes**, are defined analogous to a pairwise training set, a set of training examples, and a set of training classes, respectively.

In pairwise classification one wants to determine a decision function $g : X \times X \to \mathbb{R}$ which predicts for a pair $(u, v) \in X \times X$ whether u belongs to the same class as v $(g(u, v) > 0)$ or not $(g(u, v) < 0)$. Note that u, v do not need to belong to the set of training examples and that the classes corresponding to u and v do not need to belong to the set of training classes.

Remark 3.1. *Above, it is assumed that for each example x_i its corresponding class y_i^c is known. However, this assumption can be weakened. It suffices that for any training pair (x_i, x_j) it is known whether this pair is positive or negative. Moreover, it was assumed that Y is a finite subset of $\mathbb{N}$. This assumption can be weakened by assuming that Y is a finite set.*

This chapter is structured as follows. Section 3.1 analyzes pairwise decision functions and discusses certain properties of such functions. Those results were already presented in my papers [11, 12]. Then, in Section 3.2 several ways to model a pairwise data generating process are discussed. The presented results are new to the best of my knowledge. Section 3.3 deals with the quality of pairwise decision functions. Afterwards, in Section 3.4 a model selection approach is presented. This approach was also presented in my papers [11, 12]. Section 3.5 concludes

this chapter by a new discussion about optimal classifiers for pairwise classification problems.

3.1 Properties of a Pairwise Decision Function

In this section certain properties of a pairwise decision function are analyzed.

A first idea for a pairwise decision function could be the use of a given metric $d : X \times X \to \mathbb{R}$ in the following way

$$g_d(u, v) := b - d(u, v)$$

where $b > 0$ denotes some threshold parameter. Now, certain properties of g_d are discussed.

At first, $g_d(u, v) = g_d(v, u)$ holds for all $u, v \in X$. This raises the question whether a pairwise decision function $g : X \times X \to \mathbb{R}$ should be symmetric in the sense that

$$g(u, v) = g(v, u) \qquad \text{for all } u, v \in X.$$

In the following this property is called **pairwise symmetry** and seems intrinsic and desirable for any pairwise decision function.

Secondly, let g be a pairwise decision function. Obviously, for any $u \in X$ it holds that $g_d(u, u) = b > 0$. This leads to the question whether a pairwise decision function should be reflexive in the sense that

$$g(u, u) > 0 \quad \text{for all } u \in X$$

holds.

At first glance, this property seems desirable for a pairwise decision function. However, let an example $u \in X$ be given. Additionally, is is assumed that u may come from many different classes which all have the same likelihood. In this case it is very unlikely that $g(u, u) > 0$. Therefore, a pairwise decision function may not be reflexive. Later, in Subsection 3.5.2 a more detailed discussion on this topic is presented.

Another property of a metric is the triangle inequality. Let d be the Euclidean metric, $b = 1.1$, $X = \mathbb{R}^2$, and let $u, v, w \in \mathbb{R}^2$

$$u = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad v = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad w = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

be given. This implies

$$g_d(u, v) > 0 \qquad g_d(v, w) > 0 \quad \text{and} \quad g_d(u, w) < 0.$$

In other words g_d would not lead to transitivity in the sense that

$$g(u, v) > 0 \quad \text{and} \quad g(v, w) > 0 \quad \Rightarrow \quad g(u, w) > 0.$$

Again, transitivity seems desirable at first glance. However, it stays in some tension with the non reflexivity of the decision function. A more detailed analysis of this property can be found in Subsection 3.5.2.

Chapter 4 presents several approaches of how a **symmetric** pairwise decision function can be obtained by a pairwise Support Vector Machine. However, the described reflexivity and transitivity will not be enforced as both properties seem not to be desirable in some cases. The results of Sections 4.1 to 4.3 ensure the symmetry of a pairwise decision function and can be easily transferred to other pairwise learning methods.

3.2 Pairwise Data Generating Process

In this section several data generating processes are analyzed which can be used to model the drawing of a pairwise training set or a pairwise classification task. A pairwise classification task denotes a learning problem which should generalize to unknown pairs.

In Subsection 3.2.1 several possibilities to create pairwise training sets out of given ordinary training sets are introduced and several drawbacks of this approach are discussed. In this dissertation an ordinary training or test set means a non pairwise training or test set, respectively. In Subsection 3.2.2 the direct drawing of training pairs is analyzed, and it is shown that this approach has other drawbacks. Finally, Subsection 3.2.3 introduces a new data generating process which seems more suited for pairwise data generating processes.

3.2.1 Using a Subset of All Existing Pairs

Throughout this subsection it is assumed that there is an i.i.d. data generating process for an ordinary training set of the form (3.1). Now, some approaches to obtain a pairwise training set by a given ordinary training set are analyzed.

In order to start the discussion the following technique which generates a pairwise training set is introduced. At first, one draws an ordinary training set of the form (3.1) consisting of $m \in \mathbb{N}$ training points. Then, one creates the set $\mathcal{T}$

$$\mathcal{T} := \{((x_i, x_j), y_{ij})\}_{i,j=1}^{m} \subseteq (X \times X) \times Y. \tag{3.3}$$

In other words, the set $\mathcal{T}$ consists of all possible training pairs and corresponding labels. Finally, one draws the training pairs out of $\mathcal{T}$. Now, the drawing of the training pairs will be further investigated. To this end, $\mathcal{T}$ is equipped with a suitable σ-algebra Σ and a suitable probability measure P. Hence, $(\mathcal{T}, \Sigma, P)$ is a probability space. Now, let an i.i.d. time discrete stochastic process Z be given. Furthermore, in each step of this process a point is drawn accordingly to $(\mathcal{T}, \Sigma, P)$.

A first idea would be to use the whole set $\mathcal{T}$ as training set. In other words each point is used once. Then, if m^2 points are generated using Z it is very unlikely that each point is drawn once, independent of the chosen P. Hence, P needs to be modified after each drawing in such a way that the drawn pair cannot be drawn another time. This leads to a dependent identically distributed data generating process. Moreover, the data generating process would be finite in this setting. Hence, it cannot be a time invariant process and one cannot determine any mixing coefficient.

A second possibility would be to use Z in the i.i.d. setting. However, for large numbers of training points the obtained training set will contain a lot of redundancy as some pairs will be drawn more than once.

Above, two possibilities of how a pairwise training set can be obtained were discussed. Now, it is shown that the approach of first creating an ordinary training set and then using a subset of the corresponding set $\mathcal{T}$ has another drawback. In machine learning one assumes that the drawing of the test set is based on the same data generating process as the drawing of the training set. Now, let an ordinary training set and an ordinary test set be given and let $\overline{\mathcal{T}}$ be the set consisting of all possible test pairs and corresponding labels, that is to say $\overline{\mathcal{T}}$ is defined analogous to (3.3). Moreover, $(\overline{\mathcal{T}}, \overline{\Sigma}, \overline{P})$ is defined analogous to $(\mathcal{T}, \Sigma, P)$, too. Then, the data generating process of the training set and the data generating process of the

test set are different unless $\mathcal{T} = \overline{\mathcal{T}}$. In other words, the theory of Section 2.2 can only be applied if the set of training examples and test examples are the same. Hence, such a data generating process seems not to be suitable for the interclass generalization setting.

There exist another approach for drawing a pairwise training set and a pairwise training set out of $\mathcal{T}$. At first, one selects $1 < m_1 < m$. At second, one uses the set $\{(x_i, x_j), y_{ij}\}$ with $i, j \in \{1, \ldots, m_1\}$ as training set. At third, one uses the set $\{(x_i, x_j), y_{ij}\}$ with $i, j \in \{m_1 + 1, \ldots, m\}$ as test set. However, this approach suffers from similar problems as presented above.

3.2.2 Drawing the Pairs Directly

This subsection presents another pairwise data generating process which can be used to apply the results of Section 2.2 to pairwise classification tasks in the interclass setting. However, it is shown that this data generating process has other drawbacks.

Like in Subsection 2.2.1 it is assumed that an i.i.d. time discrete ordinary data generating process Z is given. Here, an ordinary data generating process denotes a non pairwise data generating process. At each time step an example $x \in X$ is generated by Z. Additionally, let X be equipped with a σ-algebra Σ and a probability measure P_X with density p_X. Hence, (X, Σ, P_X) is a probability space.

Now, let $X^2 := X \times X$ be defined and let Σ^2 denote the σ-algebra on X^2 generated by subsets of the form $A \times B$ where $A, B \in \Sigma$. By P_X^2 the product measure is denoted. Hence, it is defined by

$$P_X^2(A \times B) := P_X(A) P_X(B)$$

for all $A, B \in \Sigma$. Now, one can assume that the pairs are drawn accordingly to a stochastic process $\tilde{Z}(t)$ with $t \in \mathbb{Z}$. In each step of this process a pair is drawn according to (X^2, Σ^2, P_X^2) with density p_{X^2}. Then, the supervisor returns the output y for any $(u, v) \in X \times X$ according to the distribution function $F_Y(\,\cdot\,|X \times X = (u, v))$ with density $p_Y(\cdot|X \times X = (u, v))$. Like in Subsection 2.2.1 one assumes that this supervisor exists. Thus, any training point and its corresponding label is drawn according to $F_{X \times X, Y}((\cdot, \cdot), \cdot)$ with density

$$p_{X^2,Y}((u, v), y) := p_X(u) p_X(v) p_Y(y|X \times X = (u, v)).$$

Note that $\tilde{Z}$ is i.i.d. as Z is i.i.d.

From a theoretical point of view $\tilde{Z}$ is suitable for pairwise classification tasks and any result for a non pairwise learning tasks with i.i.d. input data can be transferred to this setting directly. Furthermore, the training set and the test set come from the same data generating process in this setting. Now, drawbacks of the described data generating process are discussed.

Firstly, it is often very expensive to obtain examples or the number of examples is limited. Now, let a set of m i.i.d. drawn examples be given. Then, one could obtain $\frac{m}{2}$ training pairs by using the data generating process $\tilde{Z}$. In other words, the cardinality of the pairwise training set is halved compared to the cardinality of the ordinary training set.

Secondly, a training set generated by $\tilde{Z}$ contains less information than a training set consisting of all possible pairs. Let m be even and let

$$\{((x_i, x_{i+1}), y_{i(i+1)})\}_{i \in \{1,3,\ldots,m-1\}} \subseteq (X \times X) \times Y$$

be a pairwise training set generated by $\tilde{Z}$. Now, one tries to determine for any pair (x_i, x_j) with $i, j \in M = \{1, \ldots, m\}$ whether it is positive or negative. Unfortunately, by using only the set generated by $\tilde{Z}$ this cannot be correctly determined in general. In other words, such a training set contains less information than a training set consisting of all possible pairs with corresponding labels.

In order to analyze this problem further, let a pairwise training set be given. Additionally, one assumes that the supervisor uses a reflexive, transitive, and symmetric decision function (see Section 3.1). Now, one tries to construct an ordinary training set out of the pairwise training set by only using the assumption on the supervisor. Obviously, the obtained ordinary training set will be unique except a permutation of the class numbers. Note that Subsection 3.2.1 presents possibilities to construct a pairwise training set out of an ordinary training set. Here, one is interested in a way to construct an ordinary training set out of a pairwise training set.

The following lemma shows that one cannot construct such an ordinary training set from a pairwise training set if the number of training pairs is too small. Before this lemma is presented, please note that the transitivity of the decision function implies

$$g(u, v) > 0 \quad \text{and} \quad g(v, w) > 0 \quad \Rightarrow \quad g(u, w) > 0$$

but

$$\begin{aligned} g(u,v) < 0 \quad \text{and} \quad g(v,w) < 0 \quad &\not\Rightarrow \quad g(u,w) < 0 \\ g(u,v) < 0 \quad \text{and} \quad g(v,w) < 0 \quad &\not\Rightarrow \quad g(u,w) > 0. \end{aligned} \tag{3.4}$$

Lemma 3.2. *Let the pairwise training set*

$$\{(x_i, x_j), y_{ij}\}_{(i,j)\in I} \subseteq (X \times X) \times Y$$

with $I \subseteq M \times M$ and $x_i \neq x_j$ for $i \neq j$ be given. Additionally, let the supervisor have access to the ordinary training set

$$\left\{\left(x_i, y_i^{cl}\right)\right\}_{i=1}^{m} \subseteq X \times Y_{cl}$$

and let it use

$$y_{ij} := \begin{cases} 1 & \text{if } y_i^{cl} = y_j^{cl} \\ -1 & \text{otherwise} \end{cases}$$

as classifier. Furthermore, let c be set by

$$c = |\{y_i^{cl}\}_{i=1}^{m}|.$$

In other words c is the number of different classes in the ordinary training set. Then, if $|I| < m - c + \frac{c(c-1)}{2}$ it is not possible to correctly decide for every $i, j \in M$ whether

$$y_i^{cl} = y_j^{cl}$$

or not, by only using the pairwise training set.

Proof. Note that the supervisor uses a reflexive, transitive, and symmetric decision function due to the given assumptions.

W.l.o.g. one can assume that the set of training classes Y^{cl} is $\{1, \ldots, c\}$. Here, $M_k \subseteq M$ denotes all such indices whose corresponding examples belong to class k, in other words for all $k \in \{1, \ldots, c\}$ it holds that $y_i^{cl} = k$ for all $i \in M_k$.

Firstly, it is shown that there must be $m - c$ positive pairs in the training set to correctly determine whether $y_{ij} = 1$ or not for all $i, j \in M$. Additionally, it is shown that this number is independent of the used negative pairs.

For each class $k \in \{1, \ldots, c\}$ the following graph H_k is constructed. The vertices of H_k are the examples belonging to class k. Hence, H_k consists of $|M_k|$ vertices. Then, x_i and x_j with $i, j \in M_k$ are connected if and only if $((x_i, x_j), 1)$ is part of the pairwise training set. Obviously, if H_k is connected, then the transitivity implies

for any $i, j \in M_k$ that $y_{ij} = 1$ holds. However, if H_k is disconnected, then there are at least two connected subgraphs H_{k_1} and H_{k_2}. Now, let x_i be a vertex of H_{k_1} and let x_j be a vertex of H_{k_2}. Then, due to (3.4) one cannot determine whether $y_{ij} = 1$ or $y_{ij} = -1$, independent of the negative pairs in the training set. Hence, if H_k is disconnected, then there is at least one pair (x_i, x_j) for which one cannot correctly determine whether $y_{ij} = 1$ or $y_{ij} = -1$. Any Graph with $|M_k|$ vertices and less then $|M_k| - 1$ edges is disconnected. Furthermore, there is always a connected graph with $|M_k|$ vertices and $|M_k| - 1$ edges. Hence, one can conclude that there must be

$$\sum_{k=1}^{c} (|M_k| - 1) = m - c$$

positive pairs in the training set and that this number of positive pairs is sufficient. Additionally, this number is independent of the used negative pairs.

Secondly, one shows that there must be $\frac{c(c-1)}{2}$ negative pairs to correctly determine whether $y_{ij} = 1$ or $y_{ij} = -1$ for $i, j \in M$. Furthermore, one shows that this number is sufficient.

Let $\mathcal{T} \subseteq (X \times X) \times Y$ consist of $m - c$ positive pairs such that all corresponding graphs H_k are connected. For each class k a representing example x_{c_k} is selected. Now, let the following set be added to $\mathcal{T}$

$$\{((x_{c_k}, x_{c_l}), -1) | k, l \in \{1, \ldots, c\}, k < l\}.$$

Note that the pairwise transitivity and symmetry imply

$$g(u, v) < 0 \quad \text{and} \quad g(v, w) > 0 \quad \Rightarrow \quad g(u, w) < 0.$$

Therefore, by $\mathcal{T}$ one can correctly determine for any $i, j \in M$ whether $y_{ij} = 1$ or $y_{ij} = -1$. Obviously, $\mathcal{T}$ consists of $\frac{c(c-1)}{2}$ negative pairs. Now, it is shown that this number of negative pairs is necessary. Therefore, let a training set with less than $\frac{c(c-1)}{2}$ negative pairs be used. Then, there are $k, l \in \{1, \ldots, c\}$ such that

$$((x_i, x_j), -1)$$

is not part of the training set for all $i \in M_k, j \in M_l$. Then, as (3.4) holds one cannot determine whether $y_{ij} = -1$ or $y_{ij} = -1$ for any $i \in M_k, j \in M_l$.

Hence, there must be at least

$$m - c + \frac{c(c-1)}{2}$$

pairs in the training set to correctly determine whether $y_{ij} = 1$ or $y_{ij} = -1$ for all $i, j \in M$. Additionally, it was shown that this number is sufficient. □

Note that for $m \in \{2, 4, \dots\}$ it holds that

$$\frac{m}{2} \leq m - c + \frac{c(c-1)}{2}.$$

This inequality becomes strict for $m \geq 4$ and $c \in \{1, \dots, m\}$. Hence, the data generating process $\tilde{Z}$ looses some information if two or more training pairs are used.

3.2.3 A New Pairwise Data Generating Process

Here, an algorithm is presented which can be used to generate a training set and a test set which both come from the same data generating process. However, it seems that the underlying process is not stationary. Nevertheless, it might be possible to use the block technique presented in [48, Corollary 2.7] and [29, Lemma 3] to show that the data generating process has some mixing properties.

Before the algorithm is presented the basic idea of Algorithm 3.3 is described. Let some kind of memory be available. Let this memory consist of a specific number of drawn examples. In line 1 of Algorithm 3.3 p is selected as the **example per class ratio** (EPCR) while t_1 is selected as the number of classes in the memory. The EPCR denotes the (average) number of examples per class. The memory stores the last $t_1 p$ drawn examples while older examples are removed. To initialize the memory it is filled with examples of t_1 classes with p examples per class (lines 3–9, Figure 3.1a). Note that after this step the training set $\mathcal{T}$ is still empty. Lines 10–26 can be described in the following way: At first, the algorithm draws p examples from one new class. At second, all (positive) pairs and corresponding labels from those p examples are added to the training set $\mathcal{T}$. Additionally, all those pairs and corresponding labels are added to the training set whose first member belongs to the newly drawn examples and whose second member belongs to the memory (Figure 3.1b). At third, the memory is updated by deleting the p oldest examples and adding the p new examples to the memory (Figure 3.1c). Now, the steps of lines 10-26 are repeated t times.

In Algorithm 3.3 it is assumed that an i.i.d. generator of the classes is given. Additionally, for each class k let an i.i.d. generator G_k of examples be given.

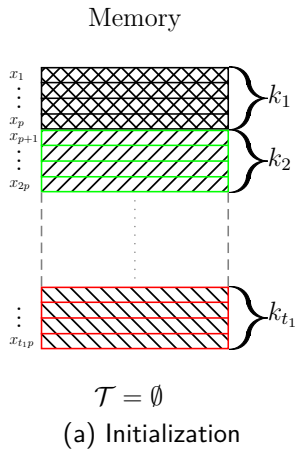
Algorithm 3.3. *A new Pairwise Data Generating Process*

Select $p, t_1, t \in \mathbb{N}$
Set $\mathcal{T} := \emptyset, e := 1$
for $i = 1, \ldots, t_1$ ***do***
  draw class k_i
  for $j = 1, \ldots, p$ ***do***
    draw example x_e *using* G_{k_i}
    Set $y_e^{cl} := k_i,\ e := e + 1$
  end for
end for
for $i = 1, \ldots, t$ ***do***
  Set $h := e$
  draw class k_i
  for $j = 1, \ldots, p$ ***do***
    draw example x_e *using* G_{k_i}
    Set $y_e^{cl} := k_i$
    for $t = h - t_1 p, \ldots, h + j - 1$ ***do***
      if $(y_e^{cl} == y_t^{cl})$ ***then***
        $y_{et} := 1$
      else
        $y_{et} := -1$
      end if
      $\mathcal{T} := \mathcal{T} \cup \{((x_e, x_t), y_{et})\}$
    end for
    Set $e := e + 1$
  end for
end for

Obviously, Algorithm 3.3 is well defined. Note that the underlying distribution of the pairs seems to depend on the time. Hence, the process seems not to be stationary. For instance, right after the initialization of the memory a number of positive pairs is added. Afterwards, a number of negative pairs added and so on.

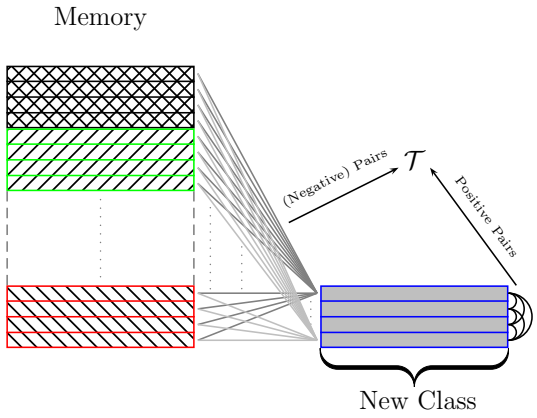
In the following some comments concerning Algorithm 3.3 are given.

- In general, a pairwise training set obtained by the algorithm cannot be used to determine whether any two examples of the corresponding set of training examples belong to the same class or not (cf. Lemma 3.2).

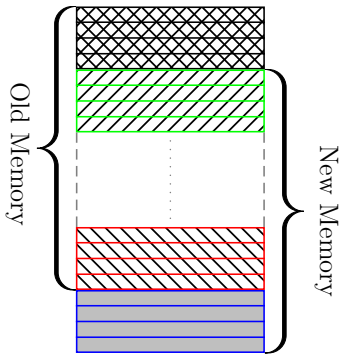


(a) Initialization

(a) The memory is initialized using t_1 classes with p examples per class.



(b) Adding Pairs
A new class and p corresponding examples are drawn. All positive pairs of those p examples are added to $\mathcal{T}$. Additionally, all (negative) pairs are added to $\mathcal{T}$ whose first member belongs to the newly drawn examples and whose second member belongs to the current memory.



(c) Update of the Memory

(c) The oldest class and its corresponding examples are deleted from the memory. The new class and its corresponding examples are added to the memory.

Figure 3.1: Description of Algorithm 3.3

- Let $\mathcal{T}$ denote a training set obtained by Algorithm 3.3. Then, $i \leq j$ holds for any $((x_i, x_j), y_{ij}) \in \mathcal{T}$. Chapter 4 shows that it suffices to use such pairwise training sets in pairwise SVMs.

- Note that this algorithm may draw the same class two times. Therefore, a training set and a test set obtained by this algorithm might not lead to the interclass setting which was described in the beginning of this chapter. However, if the number of existing classes is sufficiently large, then this should be a rare event.

- In order to conclude that the test set and the training set are generated by the same process in the interclass setting, let a sufficiently large pairwise training set be obtained by Algorithm 3.3. Furthermore, let the elements of the training set be numbered from 1 to m in the ordering obtained by thealgorithm. Then, the elements $\{1, \ldots, m_1\}$ are used as training set and the elements $\{m_1 + (p(p+1)/2 + p^2 t_1)t_1 + 1, \ldots, m\}$ are used as test set for an appropriate m_1. This guarantees that the set of training classes and the set of test classes do not intersect unless a specific class is drawn two times.

- Note that one implicitly selects the a priory probability of a positive pair of the obtained training set $\mathcal{T}$ by selecting p and t_1 if the number of existing classes is sufficiently large. This raises the question which a priory probability is reasonable.

- One could modify this algorithm in the following way. For each class a random number of examples is drawn, in other words p is drawn randomly for each class.

- This algorithm leads to training sets which are smaller than the approach of including all existing pairs to the training set. This significantly shortens the needed training time but may decrease the accuracy. In Subsection 5.2.2 the impact of using training sets which do not use all possible training pairs is analyzed.

- Algorithm 3.3 is not used to generate the data in Section 5.2. Instead, two other data generating processes are used which are now presented. The first one includes all possible pairs to the training set. The second one includes all positive pairs to the training set similar as Algorithm 3.3. In contrast to Algorithm 3.3 the negative pairs are selected randomly out of the existing negative pairs.

3.3 Evaluating the Quality of a Pairwise Decision Function

Section 2.5 discussed how a test set can be used to evaluate the quality of a non pairwise decision function by means of comparing empirical risks of given test sets. Now, the empirical risk of two pairwise test sets and certain classifiers are calculated. Let the first test set consist of all pairs of $1{,}000$ examples. It is assumed that there are 200 classes in this set and that there is a **constant EPCR** of 5, in other words each class consists of 5 examples. If a symmetric pairwise decision function is used, then there are $2{,}000$ positive points and $498,500$ negative points. Now, let the second test set consist of $10{,}000$ examples of $2{,}000$ classes. Again, each class has a constant EPCR of 5. Then, there are $20{,}000$ positive pairs and $49{,}985{,}000$ negative pairs. Obviously, the number of positive pairs increases linearly with the number of classes while the number of negative pairs increases quadratically. Now, let the decision function $g : X \times X \to \mathbb{R}$ be defined by $g(u, v) := -1$ for all $u, v \in X$. In other words, it always predicts that u and v do not belong to the same class. If the 0-1-loss is used then the second training set leads to a smaller empirical risk (2.5) than the first training set, as the a priory probability of a positive pair is smaller. Moreover, the empirical risk would converge to zero if the number of used classes is further increased. Hence, another method to measure the quality of a pairwise decision function is needed.

The basic idea of the presented measure (cf. [18]) is to look at the empirical risk on the positive pairs and the empirical risk on the negative pairs separately. If the 0-1-loss is used, then the empirical risk on the positive pairs is called (empirical) **false non match rate (FNMR)** while the empirical risk on the negative pairs is called (empirical) **false match rate (FMR)**.

Now, let a decision function $g : X \times X \to \mathbb{R}$ be given and let $g^b : X \times X \to \mathbb{R}$ be defined by

$$g^b(u, v) := g(u, v) + b \tag{3.5}$$

with $b \in \mathbb{R}$. Any pair $(u, v) \in X \times X$ is classified according to $\operatorname{sgn}(g^b(u, v))$.

Now, the FMR and the FNMR are regarded as functions depending on b. For instance, in order to calculate FMR(b) one uses g^b and calculates the corresponding FMR. Note that the function FMR$(\cdot)$ and FNMR$(\cdot)$ are step functions and therefore do not continuously depend on b. However, the FNMR decreases monotonously with b and the FMR increases monotonously with b. If there is $\hat{b}$ with FMR($\hat{b}$)=FNMR($\hat{b}$) then this function value is called **equal error rate (EER)**. Due to the discontinuity

of FMR$(\cdot)$ and FNMR$(\cdot)$ this value does not necessarily need to exist. To get a substitute for the EER one can linearly connect the discontinuities of the FMR function and of the FNMR function, respectively. Then, one can use any intersection point of the obtained curves as substitute.

The EER or its substitute seems to be a good indicator for the quality of a pairwise decision function at a first glance. However, it might be insufficient within real world applications. For example, in security applications one should use a very low FMR and therefore has to accept a higher FNMR. Hence, the EER could lead to wrong conclusions in this setting. For this reason it is proposed to use **detection error trade-off (DET)** curves

$$\{(\mathrm{FMR}(b), \mathrm{FNMR}(b)) \mid b \in \mathbb{R}\}$$

to determine the quality of a pairwise decision function (see Section 5.2 for examples). For a given pairwise test set a pairwise decision function g_A is called (strictly) better than a pairwise decision function g_B if the DET curve of g_A is (strictly) below the DET of g_B.

There are several other measures which can be used for determining the quality of a pairwise classifier (see [18]). For example, one could use receiving operating curves instead. Many of those measures are closely connected to the DET curve. Furthermore, like the EER there are other measurements for the quality of a decision function based on scalars like the F-score. However, all such measurements suffer from the problem that there are specific applications where they would lead to wrong conclusions.

3.4 A Heuristic Model Selection Technique

Most learning algorithms have a large number of parameters. For instance, in a SVM one has to select the penalty parameter C or the used kernel. The selection of those parameters is called model selection. A common way to select such parameters is a grid search of the set of parameters. In a grid search many models must be calculated which is often computationally very expensive.

For pairwise classification this problem becomes even worse. For instance, for pairwise SVMs (see Chapter 4) one has to select two different kernels instead of one kernel. Additionally, it may be possible to select the EPCR or the number of classes

in a pairwise training set. Therefore, a new heuristic model selection technique is presented which is based on tasks of increasing levels of difficulty.

In this section let an ordinary training set of the form (3.1) with corresponding pairwise training sets of the form (3.2) be given. Additionally, let an ordinary test set with a corresponding pairwise test set be given. Now, three tasks which can be used for model selection are introduced. Firstly, the **interclass task** is introduced. In this task the intersection of the set of training classes and the set of test classes (see beginning of Chapter 3) is empty. For instance, let face recognition be considered in which the interclass task is to classify pairs of unknown images of unknown persons. However, if the interclass task is used to measure the quality of a pairwise decision function, one cannot determine whether a bad result is caused by badly chosen parameters of the learning machine, or by a bad EPCR, or by an undersized number of classes in the training set.

In addition to the interclass task it will turn out that the next two tasks can be used for model selection, too. In the **interexample task** the intersection of the set of training examples and the set of test examples is empty while the set of training classes and the set of test classes are equal. Thus, in face recognition the interexample task is to classify pairs of unknown images of known persons. In the **pair task**, the set of training examples and the set of test examples are equal while the intersection of the pairwise training set and the pairwise test set is empty. Therefore, the pairwise training set is a real subset of all pairs of the training examples for this task. Hence, in face recognition the pair task is to classify unknown pairs of known images.

Assuming that a pairwise classification task is given. Then, the interclass task seems harder than the interexample task which again seems harder than the pair task. Thus, if a bad result on the pair task or interexample task is achieved, then one cannot expect a good result on the interclass task. Therefore, it is suggested to use the pair task to find sufficiently good parameters of the learning machine like the used kernel of the used penalty parameters of a pairwise SVM, the interexample task to find a sufficient EPCR in the training set, and the interclass task to find a sufficiently large set of training classes. In other words, the model selection heuristic is based on the assumption that a bad performance in the pair task leads to a bad performance in the interexample task which again leads to a bad performance in the interclass task. However, it is not based on the assumption that a good performance on the pair task leads to a good performance in the interexample task which again leads to a good performance in the interclass task. In general, the needed training time is much shorter in the pair task and interexample task than in the class task

as smaller training sets can be used. Empirical evidence for using the described technique is given in Section 5.2. Moreover, it seems important to specify the used task for any measurement of the quality of a pairwise classification task.

Above, it was proposed to use the interexample task to find a sufficient large EPCR. Now, let a training set with an EPCR of e_1 and a training set with an EPCR of e_2 with $e_1 < e_2$ be given. Both training sets should be used in the interexample task. Then, to minimize side effects, it is proposed that the set of classes is equal for both training sets. Furthermore, the set of examples and the set of pairs corresponding to the first training set should be a subset of the set of examples and the set of pairs corresponding to the second training set, respectively. The same subset relations should hold for two training sets in the interclass task, where the first training set consists of fewer classes than the second training set. Additionally, the set of training classes of the first training set should be a subset of the training classes of the second training set in the interclass task.

There is a difference between the interclass task and the interexample task by means of optimal classifiers. Later, in Section 3.5 pairwise Bayes' Classifiers and Bayes' DET curves are introduced. Additionally, Section 5.2 shows that the Bayes' DET curve of the interexample task and the Bayes' DET curve of the interclass task may be different.

3.5 Pairwise Bayes' Classifiers

For non pairwise classification tasks Bayes' Classifiers are reviewed in Section 2.4. It is recalled that Bayes' Classifiers are optimal, in other words for a given loss function there does not exist a classifier with a smaller risk than the corresponding Bayes' Classifier.

In Section 2.4 it was assumed for non pairwise classification tasks that the data generating distribution function $F_{X,Y}$ is known. Similarly, it is assumed in this section that the pairwise data generating distribution function $F_{X\times X,Y}$ is known and that a corresponding density $p_{X\times X,Y}$ exists. Then, one can follow the same procedure as in Section 2.4 to obtain a pairwise Bayes' Classifier. Now, let a pairwise Bayes' Classifier based on the 0-1-loss be given. Then, such a classifier leads to a specific FMR and a specific FNMR. In Section 3.3 it was discussed that one should use DET curves of classifiers to measure their quality. Therefore, Bayes' DET curves are introduced. A Bayes' DET curve is a DET curve which is obtained by a set of Bayes' Classifiers.

3.5.1 Bayes' DET Curves

At first, a way to calculate the FMR and the FNMR for a given classifier is presented. Let l be the 0-1-loss. Moreover, let $f : X \times X \to Y$ be an arbitrary pairwise decision function. Then, the risk of f is

$$\begin{aligned} R(f) &= \int_{X\times X\times Y} l(y, f(u,v)) \,\mathrm{d}F_{X\times X,Y}((u,v),y) \\ &= \int_{X\times X\times Y} l(y, f(u,v)) p_{X\times X,Y}((u,v),y) \,\mathrm{d}((u,v),y). \end{aligned}$$

Then, the law of total probability yields

$$\begin{aligned} R(f) = &\int\limits_{X\times X\times Y} l(y, f(u,v)) \left[p_{X\times X}(u,v|Y=1)P(Y=1) + \right. \\ &\left. p_{X\times X}(u,v|Y=-1)P(Y=-1)\right] \mathrm{d}((u,v),y) \\ =&P(Y=1)\int_{X\times X} l(1, f(u,v)) p_{X\times X}(u,v|Y=1)\,\mathrm{d}(u,v) \\ &+ P(Y=-1)\int_{X\times X} l(-1, f(u,v)) p_{X\times X}(u,v|Y=-1)\,\mathrm{d}(u,v). \end{aligned}$$

Hence, the FNMR can be obtained by

$$\mathsf{FNMR}(f) := \int_{X\times X} l(1, f(u,v)) p_{X\times X}(u,v|Y=1)\,\mathrm{d}(u,v)$$

while the FMR can be obtained by

$$\mathsf{FMR}(f) := \int_{X\times X} l(-1, f(u,v)) p_{X\times X}(u,v|Y=-1)\,\mathrm{d}(u,v).$$

This yields

$$R(f) = P(Y=1)\mathsf{FMR}(f) + P(Y=-1)\mathsf{FNMR}(f). \tag{3.6}$$

A pairwise Bayes' Classifier based on the 0-1-loss function, would classify according to the signum of

$$g^0(u,v) := P(Y=1|X\times X=(u,v)) - P(Y=-1|X\times X=(u,v)) \tag{3.7}$$

with $g^0 : X \times X \to \mathbb{R}$. Now, one could apply the technique described in Section 3.3 to (3.7) to obtain a Bayes' DET curve. Similar to (3.5), this yields

$$g^b(u,v) := g^0(u,v) + b. \tag{3.8}$$

However, for $b \neq 0$ it is not obvious that there is a loss function such that $\operatorname{sgn}(g^b)$ is a corresponding Bayes' Classifier. Therefore, another way to obtain a Bayes' DET curve is now presented. Later, in Lemma 3.4 it is shown that $\operatorname{sgn}(g^b)$ is a Bayes' Classifier for a specific weighted loss function for all $b \in (0, \infty)$.

Now, let d be an element of $(0, \infty)$ and let Bayes$^d : X \times X \to Y$ denote the pairwise Bayes' Classifier which uses the weighted loss function l_d (2.26), that is Bayes^d classifies the pair (u, v) according to the signum of $\hat{g}^d : X \times X \to \mathbb{R}$ with

$$\hat{g}^d(u, v) := d \cdot P(Y = 1 | X \times X = (u, v)) - P(Y = -1 | X \times X = (u, v)). \quad (3.9)$$

Then, one can use the curve

$$D := \{(\text{FMR}(\text{Bayes}^d), (\text{FNMR}(\text{Bayes}^d)) \mid d \in (0, \infty)\} \subset [0, 1]^2 \quad (3.10)$$

as Bayes' DET curve.

Now, the case where d tends to zero is considered. Then,

$$g^{min}(u, v) := \lim_{d \to 0} \operatorname{sgn}(\hat{g}^d(u, v)) = \begin{cases} 1 & \text{if} \quad P(Y = -1 | X \times X = (u, v)) = 0 \\ -1 & \text{otherwise.} \end{cases}$$

For $d \to \infty$ one obtains

$$g^{max}(u, v) = \lim_{d \to \infty} \operatorname{sgn}(\hat{g}^d(u, v)) = \begin{cases} -1 & \text{if} \quad P(Y = 1 | X \times X = (u, v)) = 0 \\ 1 & \text{otherwise.} \end{cases} \quad (3.11)$$

Hence, in the first (second) case any point (u, v) is classified negative (positive) unless the probability that (u, v) is negative (positive) is 0. Therefore, one gets FMR$(g^{min}) = 0$ and FNMR$(g^{max}) = 0$. Note that for $d_1 < d_2$ it holds that FMR(Bayes^{d_1}) $\leq$ FMR(Bayes^{d_2}) and FNMR(Bayes^{d_1}) $\geq$ FNMR(Bayes^{d_2}). Hence, one obtains a maximal FNMR in the first case and a maximal FMR in the second case. In other words, no Bayes' Classifier Bayes^d has a larger FNMR (FMR) than the maximal FNMR (FMR).

Above, g^b was defined in equation (3.8). In the following lemma it is shown that $\operatorname{sgn}(g^b) = \operatorname{sgn}(\hat{g}^d)$ holds for selected parameters b and d. Here, $\hat{g}^b$ is defined as in (3.9). Moreover, it is shown that for every b there is d such that $\operatorname{sgn}(g^b) = \operatorname{sgn}(\hat{g}^d)$ holds and vice versa.

Lemma 3.4. *Let g^b be defined as in* (3.8) *and $\hat{g}^d$ be defined as in* (3.9)*. Then,*

$$1 - b = \frac{2}{d+1}$$

implies

$$\operatorname{sgn}(g^b(u,v)) = \operatorname{sgn}(\hat{g}^d(u,v)) \quad \textit{for all} \quad u, v \in X.$$

Proof. Note that $\operatorname{sgn}(g^b(u,v)) = 1$ if and only if

$$\begin{aligned} & P(Y=1|X\times X=(u,v)) \geq P(Y=-1|X\times X=(u,v)) - b \\ \Leftrightarrow \quad & P(Y=1|X\times X=(u,v)) \geq 1 - P(Y=1|X\times X=(u,v)) - b \\ \Leftrightarrow \quad & P(Y=1|X\times X=(u,v)) \geq \frac{1-b}{2}. \end{aligned}$$

Similarly, $\operatorname{sgn}(\hat{g}^d(u,v)) = 1$ if and only if

$$\begin{aligned} & d \cdot P(Y=1|X\times X=(u,v)) \geq P(Y=-1|X\times X=(u,v)) \\ \Leftrightarrow \quad & d \cdot P(Y=1|X\times X=(u,v)) \geq 1 - P(Y=1|X\times X=(u,v)) \\ \Leftrightarrow \quad & P(Y=1|X\times X=(u,v)) \geq \frac{1}{d+1}. \end{aligned}$$

Thus, $1 - b = \frac{2}{d+1}$ implies $\operatorname{sgn}(g^b(u,v)) = \operatorname{sgn}(\hat{g}^d(u,v))$ for all $u, v \in X$. □

In general, the curve D is not connected. Here, a way to resolve any jump point of D by means of a set of certain classifiers based on the same loss function is presented. Note that it is is assumed that $\operatorname{sgn}(0) = 1$ holds. Now, let d^* be a jump point of D, in other words for any $\varepsilon > 0$ there is some $\delta > 0$ such that

$$\left\| \begin{pmatrix} \text{FMR}(\text{Bayes}^{d^*}) - \text{FMR}(\text{Bayes}^{d^*+\varepsilon}) \\ \text{FNMR}(\text{Bayes}^{d^*}) - \text{FNMR}(\text{Bayes}^{d^*+\varepsilon}) \end{pmatrix} \right\| \geq \delta.$$

Let the subset $\mathcal{A}$ of $X \times X$ be defined as

$$\mathcal{A} := \left\{ (u,v) \in X \times X | g^{d^*}(u,v) = 0 \right\}. \tag{3.12}$$

Hence, $\mathcal{A}$ consists of all such pairs which are classified positive for $d > d^*$ and negative for $d < d^*$. Now, the expected weighted loss l_d (2.26) with $d = d^*$ for an arbitrary $(u,v) \in \mathcal{A}$ is calculated. If (u,v) is classified positive, then

$$l_{d^*}(-1,1) \cdot P(Y=-1|X\times X=(u,v)) = 1 \cdot P(Y=-1|X\times X=(u,v)).$$

If (u, v) is classified negative, then the expected loss is

$$l_{d^*}(1, -1) \cdot P(Y = 1|X \times X = (u, v)) = d^* \cdot P(Y = 1|X \times X = (u, v)).$$

By definition (3.12) it follows that

$$P(Y = -1|X \times X = (u, v)) = d^* \cdot P(Y = 1|X \times X = (u, v)).$$

Hence, the risk does not change regardless of the chosen classification rule on the set A. Note that the corresponding FMR and the FNMR may depend on the chosen classification rule. For each $t \in [0, 1]$ let B_t denote a Bernoulli variable mapping to $\{-1, 1\}$ instead of $\{0, 1\}$ with probability t of drawing 1. Then, one defines $\mathsf{Bayes}_t^{d^*} : (X \times X) \to \mathbb{R}$ by

$$\mathsf{Bayes}_t^{d^*}(u, v) := \begin{cases} \mathsf{Bayes}^{d^*}(u, v) & \text{if } (u, v) \notin \mathcal{A} \\ B_t & \text{if } (u, v) \in \mathcal{A} \end{cases}$$

Now, one can define the curve

$$\hat{D} := D \cup \left\{ (\mathsf{FMR}(\mathsf{Bayes}_t^{d^*}), \mathsf{FNMR}(\mathsf{Bayes}_t^{d^*})) \mid t \in (0, 1] \right\}.$$

Note that for any $t \in [0, 1]$ $\mathsf{Bayes}_t^{d^*}$ is an optimal classifier for the weighted loss function l_{d^*}. Furthermore, the jump point d^* of D is resolved in $\hat{D}$.

In the following, it is shown that Bayes' DET curves are strictly monotone. Let the curve D be connected. Then, one defines $\mathcal{C} \subset [0, 1]^2$ by

$$\mathcal{C} := \left\{ (x, y) \in [0, 1]^2 | \exists d \in (0, \infty) : x = \mathsf{FMR}(\mathsf{Bayes}^d), y < \mathsf{FMR}(\mathsf{Bayes}^d) \right\}.$$

The optimality of the Bayes' Classifiers implies that for any classifier g

$$\begin{pmatrix} \mathsf{FMR}(g) \\ \mathsf{FNMR}(g) \end{pmatrix} \notin \mathcal{C}$$

holds. In other words, there does not exist a pairwise classifier which has the same FMR (FNMR) and a smaller FNMR (FMR) as a pairwise Bayes' Classifier. In particular, one obtains:

Corollary 3.5. *Let $d_1, d_2 \in (0, \infty)$ with $d_1 < d_2$ be given. Then,*

$$\left(\mathit{FMR}(\mathit{Bayes}^{d_1}) < \mathit{FMR}(\mathit{Bayes}^{d_2}) \right) \Leftrightarrow \left(\mathit{FNMR}(\mathit{Bayes}^{d_1}) > \mathit{FNMR}(\mathit{Bayes}^{d_2}) \right).$$

Proof. Above, it was discussed that

$$\text{FMR}(\text{Bayes}^{d_1}) \leq \text{FMR}(\text{Bayes}^{d_2}) \text{ and } \text{FNMR}(\text{Bayes}^{d_1}) \geq \text{FNMR}(\text{Bayes}^{d_2})$$

holds. Now, it is assumed that

$$\text{FMR}(\text{Bayes}^{d_1}) < \text{FMR}(\text{Bayes}^{d_2}) \text{ and } \text{FNMR}(\text{Bayes}^{d_1}) = \text{FNMR}(\text{Bayes}^{d_2}).$$

holds. Now, the risk with weighted loss function l_{d_2} of Bayes^{d_1} is calculated. Equation (3.6) yields

$$\begin{aligned} R(\text{Bayes}^{d_1}) &= P(Y=1)\text{FNMR}(\text{Bayes}^{d_1}) + P(Y=-1)d_2\text{FMR}(\text{Bayes}^{d_1}) \\ &< P(Y=1)\text{FNMR}(\text{Bayes}^{d_2}) + P(Y=-1)d_2\text{FMR}(\text{Bayes}^{d_2}) \\ &= R(\text{Bayes}^{d_2}) \end{aligned}$$

Obviously, this inequality is a contradiction to the optimality of $R(\text{Bayes}^{d_2})$ with respect to the loss function l_{d_2}.

Vice versa one assumes that

$$\text{FMR}(\text{Bayes}^{d_1}) = \text{FMR}(\text{Bayes}^{d_2}) \text{ and } \text{FNMR}(\text{Bayes}^{d_1}) > \text{FNMR}(\text{Bayes}^{d_2})$$

holds. Then, one shows that this is a contradiction by calculating the risk with weighted loss function l_{d_1} of Bayes^{d_2}. □

The following lemma shows that a Bayes' DET curve is convex in some sense.

Lemma 3.6. *Let the curve D be defined as in* (3.10) *and let it be connected. Moreover, using* (3.11) *FMR_{max} is defined by $\text{FMR}_{max} := \text{FMR}(g^{max})$. Additionally, let $\text{FMR}_1, \text{FMR}_2 \in (0, \text{FMR}_{max})$ with $\text{FMR}_1 < \text{FMR}_2$ be selected. Now, for an arbitrary but fixed $\lambda \in (0,1)$ one defines $\text{FMR}_3 := \lambda\text{FMR}_1 + (1-\lambda)\text{FMR}_2$.*

Then, there are $d_1, d_2, d_3 \in (0, \infty)$ such that

$$\text{FMR}(\text{Bayes}^{d_1}) = \text{FMR}_1, \text{FMR}(\text{Bayes}^{d_2}) = \text{FMR}_2, \text{ and } \text{FMR}(\text{Bayes}^{d_3}) = \text{FMR}_3.$$

Additionally, it holds that

$$\text{FNMR}(\text{Bayes}^{d_3}) \leq \lambda\text{FNMR}(\text{Bayes}^{d_1}) + (1-\lambda)\text{FNMR}(\text{Bayes}^{d_2}).$$

Proof. As D is connected d_1, d_2, d_3 exist. Now, for $i = 1, 2, 3$ one sets

$$\mathsf{FNMR}_i := \mathsf{FNMR}(\mathsf{Bayes}^{d_i}).$$

Then, Corollary 3.5 yields $\mathsf{FNMR}_1 > \mathsf{FNMR}_3 > \mathsf{FNMR}_2$. Let the contrary of the hypothesis be assumed, in other words, it should hold that

$$\mathsf{FNMR}_3 > \lambda\mathsf{FNMR}_1 + (1-\lambda)\mathsf{FNMR}_2.$$

Now, one defines

$$\tilde{\mathcal{A}} := \left\{(u,v) \in X \times X | \mathsf{Bayes}^{d_1}(u,v) \neq \mathsf{Bayes}^{d_2}(u,v)\right\}.$$

Additionally, one defines the classifier $f : X \times X \to Y$ by

$$f(u,v) := \begin{cases} \mathsf{Bayes}^{d_1}(u,v) & \text{if } (u,v) \notin \tilde{\mathcal{A}} \\ B_{1-\lambda} & \text{if } (u,v) \in \tilde{\mathcal{A}} \end{cases}.$$

Again, $B_{1-\lambda}$ denotes a Bernoulli variable which maps to $\{-1, 1\}$ instead of $\{0, 1\}$ with probability $(1-\lambda)$ of success. Then, $(1-\lambda)$ of the negative pairs of $\tilde{\mathcal{A}}$ are classified wrong by f, while λ of the positive pairs of $\tilde{\mathcal{A}}$ are classified wrong by f. Therefore,

$$\begin{aligned} \mathsf{FMR}(f) =& \mathsf{FMR}_1 + (1-\lambda)(\mathsf{FMR}_2 - \mathsf{FMR}_1) \\ =& \lambda\mathsf{FMR}_1 + (1-\lambda)\mathsf{FMR}_2 \\ =& \mathsf{FMR}_3 \\ \mathsf{FNMR}(f) =& \mathsf{FNMR}_2 + \lambda(\mathsf{FNMR}_1 - \mathsf{FNMR}_2) \\ =& \lambda\mathsf{FNMR}_1 + (1-\lambda)\mathsf{FNMR}_2 \\ <& \mathsf{FNMR}_3. \end{aligned}$$

Now, the risk (3.6) of f with weighted loss function (2.26) l_{d^3} is calculated.

$$\begin{aligned} R(f) &= P(Y=1)\mathsf{FNMR}(f) + P(Y=-1)d_3\mathsf{FMR}(f) \\ &< P(Y=1)\mathsf{FNMR}(\mathsf{Bayes}^{d_3}) + P(Y=-1)d_3\mathsf{FMR}(\mathsf{Bayes}^{d_3}) \\ &= R(\mathsf{Bayes}^{d_3}). \end{aligned}$$

Hence, f would lead to a smaller risk than the corresponding Bayes' Classifier Bayes^{d_3}. This is a contradiction. □

3.5.2 Properties of Pairwise Bayes' Classifiers

Now, let an non pairwise classification task with $Y_c = \{1, \ldots, c\}$ be given. Additionally, let the non pairwise distribution function F_{X,Y_c} be known. Then, for any $(x, y) \in X \times Y_c$ one can calculate the class probability

$$P(Y_c = y | X = x).$$

If all classes have the same cost of misclassification, then the Bayes' Classifier is

$$g(x) := \operatorname*{argmax}_{y \in Y_c} P(Y_c = y | X = x).$$

Now, a way to obtain a pairwise Bayes' Classifier by the class probabilities is presented. Obviously, under the i.i.d. assumption of the examples, one can calculate the probability $P(Y = 1 | X \times X = (u, v))$ that the two examples of the pair (u, v) belong to the same class by

$$\begin{aligned} \bar{g}(u, v) &:= P(Y = 1 | X \times X = (u, v)) \\ &= \sum_{i=1}^{c} P(Y_c = i | X = u) P(Y_c = i | X = v). \end{aligned} \tag{3.13}$$

Hence, if for each example all class probabilities are known, then one can obtain a pairwise Bayes' Classifier based on the 0-1-loss easily by

$$\operatorname{sgn}(\bar{g}(u, v) - 0.5).$$

Now, let Y_c be equal to $\{1, 2, 3\}$. Moreover, let $P_i : X \to [0, 1]$ be defined by

$$P_i(u) := P(Y_c = i | X = u).$$

In other words, $P_i(u)$ denotes the probability that $u \in X$ belongs to class i. Now, let

$$P_1(u) = P_2(u) = P_3(u) = \frac{1}{3}$$

hold. Then,

$$\bar{g}(u, u) = 3\frac{1}{3^2} = \frac{1}{3} < 0.5$$

holds. Hence, one should not require the reflexivity of a pairwise decision function (see Section 3.1).

Now, for $u, v \in X$ it is assumed that

$$P_1(u) = 0.6, \quad P_2(u) = P_3(u) = 0.2$$
$$\text{and} \quad P_1(v) = 1, \quad P_2(v) = P_3(v) = 0.$$

holds. This implies

$$\bar{g}(u,u) = 0.44, \quad \bar{g}(u,v) = 0.6, \quad \bar{g}(v,v) = 1.$$

Note that u is non reflexive. At the same time the pair (u, v) would be classified positive by $\operatorname{sgn}(\bar{g}(\cdot,\cdot) - 0.5)$. Moreover, $\bar{g}$ is symmetric and

$$\bar{g}(u,u) < 0.5, \quad \text{while} \quad \bar{g}(u,v) = \bar{g}(v,u) > 0.5$$

holds. Therefore, the pairwise transitivity of a decision function is not desirable in general.

3.5.3 Examples of Pairwise Bayes' Classifiers for Interclass Tasks and Interexample Tasks

Now, let the examples a, b, c, and d be given and let Y_c be defined by $Y_c := \{1, 2, 3\}$. Furthermore, let the probabilities P_i that an example belongs to class $i \in Y_c$ be

	$P_i(a)$	$P_i(b)$	$P_i(c)$	$P_i(d)$
$i = 1$	0.8	0.2	0.7	0.7
$i = 2$	0.15	0.6	0.1	0.3
$i = 3$	0.05	0.2	0.2	0.0.

Now, it holds that

$$\bar{g}(a,b) = 0.26 < 0.5 \quad \bar{g}(c,d) = 0.52 > 0.5.$$

Hence, (a, b) would be classified negative by the Bayes' Classifier based on the 0-1-loss and (c, d) would be classified positive.

Above, the interclass task was considered. In the following, the interexample task is considered. In the latter task the set of training classes and the set of test classes are equal. Therefore, a Bayes' Classifier has information about the existing classes. Now, it is assumed that class 1 does not belong to the test set. Hence,

the probabilities P_i can be recalculated by setting the probability of class 1 to zero and normalizing the other probabilities. One obtains

	$P_i(a)$	$P_i(b)$	$P_i(c)$	$P_i(d)$
$i=1$	0.0	0.0	0.0	0.0
$i=2$	0.75	0.75	0.33	1.0
$i=3$	0.25	0.25	0.67	$0.0.$

Now, it holds that

$$\bar{g}(a,b) = 0.625 > 0.5 \quad \bar{g}(c,d) = 0.33 < 0.5.$$

Hence, the classification rule for both pairs is different in the interclass task and interexample task. Later, Section 5.2 presents pairwise classification task which lead to different Bayes' DET curves in the interclass and interexample task.

Remark 3.7. *Let two examples $u, v \in X$ be given. Then, one could use a non pairwise deterministic classifier to obtain a pairwise classification rule. Firstly, one predicts the classes k_u, k_v of the examples u and v. Secondly, the pair (u, v) is classified positive if and only if $k_u = k_v$. Therefore, such a classifier is always reflexive. Hence, the approach of using two input examples in pairwise classification has a significant difference to the naive approach of combining non pairwise classifiers. This does even hold in the interexample setting.*

4 Pairwise Support Vector Machines

Section 2.3 introduced ordinary SVMs. From now on non pairwise SVMs are referred as ordinary SVMs. A SVM is a binary classifier which implements the Structural Risk Minimization principle (see Subsection 2.2.3). Chapter 3 dealt with pairwise classification tasks. In this chapter **pairwise SVMs** are introduced. A pairwise SVM is a SVM which can handle pairwise classification tasks.

This chapter is structured as follows. Section 4.1 deals with the extension of ordinary SVMs to pairwise SVMs by means of certain decompositions of the pairwise decision function. Note that there are several other papers dealing with this setting, for instance [1, 2, 3, 5, 7, 19, 32, 43]. However, independently of the other authors I developed a new approach in [11] by decomposing the decision function into several distinct functions. This approach offers some new insight from a theoretical perspective. It is shown that the symmetry of a pairwise decision function can be enforced by means of certain projections and that the kernel trick can be applied in two different ways. Note that I introduced another way to derive pairwise SVMs in [12, Section 2]. This approach may be easier to understand than the approach presented in Section 4.1. However, the latter approach gives more insight into pairwise SVMs. Section 4.2 deals with the evaluation of pairwise kernel function values. Afterwards, in Section 4.3 a discussion about the drawbacks of enforcing the pairwise symmetry by means of projections is presented. Then, another way of enforcing the symmetry by means of training sets with a special structure is introduced in Section 4.4. This result was already claimed in [2, 45]. Here, this result is proven in a more general context. Additionally, it is shown in Section 4.5 that the new approach yields to the same decision function as the approach by means of projections for selected parameters. It is shown that for each parameter set of the approach using selected kernels there exists another parameter set of the approach using training sets with special structure so that both approaches lead to the same decision function and the other way around. The results of Sections 4.2

to 4.5 were already stated in my submitted paper [12]. Finally, Section 4.6 concludes this chapter with some remarks.

Throughout this chapter scalar products on the direct sum of two vector spaces or the tensor product of two vector spaces are needed. To this end, the following remark is given.

Remark 4.1. *Let W denote an arbitrary real Hilbert space and let a, b, c, d be elements of W. Then,*

$$\langle (a,b),(c,d)\rangle := \langle a,c\rangle + \langle b,d\rangle$$

defines a scalar product on the direct sum $W \oplus W$. Moreover, the bilinear continuation of

$$\langle a \otimes b, c \otimes d\rangle := \langle a,c\rangle \cdot \langle b,d\rangle$$

defines a scalar product on the tensor product $W \otimes W$. These scalar products are called ***canonical scalar product on*** *$W \oplus W$ or on $W \otimes W$, respectively.*

4.1 Decomposing Decision Functions

Let $u, v \in X$ denote two examples. Similar to Chapter 3, one is interested in a pairwise decision function $g : X \times X \to \mathbb{R}$ with $g(u,v) > 0$ if and only if u and v belong to the same class. In this section it is shown how a pairwise decision function can be obtained by means of the SVM framework. Section 3.1 discussed that any pairwise decision function should be symmetric with respect to the order of the input examples. Subsection 4.1.1 proposes to decompose g into several functions. Then, the pairwise symmetry can be enforced by using certain projections. Subsection 4.1.2 shows two ways of applying the kernel trick. By using the decomposition of the pairwise decision function presented Subsection 4.1.2, it is shown that those two ways differ from each other.

4.1.1 Linear Pairwise SVMs

In order to construct a symmetric pairwise decision function one could decompose the decision function $g : X \times X \to \mathbb{R}$ into

$$g = h \circ Q,$$

where $Q : X \times X \to \mathcal{D}$, $h : \mathcal{D} \to \mathbb{R}$, and $\circ$ denotes the composition of two functions. In this subsection two possibilities of the mapping Q are analyzed, namely:

- the direct sum mapping $Q := Q_D$ with $Q_D(a, b) := (a, b)$ for all $a, b \in X$ and $\mathcal{D} := X \oplus X$,
- and the tensor product mapping $Q := Q_T$ with $Q_T(a, b) := a \otimes b$ for all $a, b \in X$ and $\mathcal{D} := X \otimes X$.

Note that if the operands of $\otimes$ are vector spaces, then $\otimes$ denotes the tensor product space of these operands. Otherwise, if the operands of $\otimes$ are vectors, then $\otimes$ denotes their tensor product.

Using this decomposition g can be written as $g = h_D \circ Q_D$ with $h_D : X \oplus X \to \mathbb{R}$, or as $g = h_T \circ Q_T$ with $h_T : X \otimes X \to \mathbb{R}$. In Subsection 4.1.2 extensions of this approach are discussed.

As stated above one wants to construct a symmetric pairwise decision function by means of certain projections. If Q_D is used, then it is proposed to use projections which map any element of $X \oplus X$ onto one of two subspaces, namely the symmetric direct sum space $X \oplus_S X$ and the asymmetric direct sum space $X \oplus_A X$. Similarly, if Q_T is used then one should use a projection which maps any element of $X \otimes X$ onto the symmetric tensor space $X \otimes_S X$, or one should use a projection which maps onto the asymmetric tensor space $X \otimes_A X$. Now, one defines

$$
\begin{aligned}
X \oplus_S X &:= \{(a, a) \mid a \in X\}, \\
X \oplus_A X &:= \{(a, -a) \mid a \in X\}, \\
X \otimes_S X &:= \operatorname{span}\{a \otimes b + b \otimes a \mid a, b \in X\}, \\
X \otimes_A X &:= \operatorname{span}\{a \otimes b - b \otimes a \mid a, b \in X\}.
\end{aligned}
$$

Then, $X \oplus_S X \perp X \oplus_A X$ with respect to the canonical scalar product on $X \oplus X$, and $X \otimes_S X \perp X \otimes_A X$ with respect to the canonical scalar product on $X \otimes X$. Moreover, it holds that

$$
\begin{aligned}
X \oplus X &= (X \oplus_S X) \,\hat{\oplus}\, (X \oplus_A X) \\
X \otimes X &= (X \otimes_S X) \,\hat{\oplus}\, (X \otimes_A X).
\end{aligned}
$$

Note that $\oplus$ denotes the outer direct sum of two vector spaces, while $\hat{\oplus}$ denotes the internal direct sum of two subspaces.

Now, the following projections are defined.

$$\begin{aligned} P_{DS} &: X \oplus X \to X \oplus_S X, \quad P_{DS}(a,b) := \frac{1}{2}(a+b, b+a) \\ P_{DA} &: X \oplus X \to X \oplus_A X, \quad P_{DA}(a,b) := \frac{1}{2}(a-b, b-a) \\ P_{TS} &: X \otimes X \to X \otimes_S X, \qquad P_{TS}(z) := \frac{1}{2}(z+\bar{z}) \\ P_{TA} &: X \otimes X \to X \otimes_A X, \qquad P_{TS}(z) := \frac{1}{2}(z-\bar{z}) \end{aligned} \tag{4.1}$$

Where $\bar{z}$ denotes the adjoint of z. For instance, if a, b are elements of $X \subseteq \mathbb{R}^n$, then $ab^\top$ is element of $X \otimes X \subseteq \mathbb{R}^{n \times n}$ with corresponding adjoint $ba^\top$.

Lemma 4.2. *For the projections defined in* (4.1) *it holds that*

$$\begin{aligned} P_{DS}(a,b) &= P_{DS}(b,a), \\ P_{DA}(a,b) &= -P_{DA}(b,a), \\ P_{TS}(z) &= P_{TS}(\bar{z}), \\ P_{TA}(z) &= -P_{TA}(\bar{z}) \end{aligned}$$

for all $a, b \in X$, $z \in X \otimes X$.

Using those projections, one can decompose h_D by

$$h_D = e_D \circ P_{DS} \quad \text{or by} \quad h_D = e_D \circ P_{DA}$$

with $e_D : X \oplus X \to \mathbb{R}$. Analogously, one can decompose h_T by

$$h_T = e_T \circ P_{TS} \quad \text{or by} \quad h_T = e_T \circ P_{TA}$$

with $e_T : X \otimes X \to \mathbb{R}$. For the sake of readability the indices belonging to e, P, Q are skipped from now on, unless a particular decomposition should be specified.

All the presented decompositions lead to

$$g = e \circ P \circ Q. \tag{4.2}$$

As stated above one wants to use the SVM framework to obtain a classifier. To this end, let a pairwise training set (3.2)

$$\{((x_i, x_j), y_{ij})\}_{(i,j) \in I}$$

with $I \subseteq M \times M$ be given.

In the following, this pairwise training set is used for learning. However, instead of learning g directly it is proposed to select appropriate P and Q and to decompose g by (4.2). Then, the SVM framework can be used to learn e. Now, let e be defined by

$$e(z) := \langle w^*, z \rangle + b^*.$$

Then, the parameters w^* and b^* can be obtained from a solution (w^*, b^*, ξ^*) of the **pairwise linear primal SVM** (see (2.16))

$$\begin{aligned} &\min_{w,b,\xi} \frac{1}{2} \langle w, w \rangle + C \sum_{(i,j)\in I} \xi_{ij} \\ &\text{s.t. } y_{ij} \left(\langle w, P\left(Q(x_i, x_j)\right) \rangle + b \right) \geq 1 - \xi_{ij} \quad \text{for all } (i,j) \in I \\ &\qquad \xi_{ij} \geq 0 \quad \text{for all } (i,j) \in I \end{aligned} \tag{4.3}$$

with $I \subseteq M \times M$. Hence, for the pairwise decision function g given by (4.2) one obtains

$$g(u, v) = \langle w^*, P(Q(u, v)) \rangle + b^*. \tag{4.4}$$

Similar as in Subsection 2.3.1 one can determine a solution of (4.3) by solving the following **pairwise linear dual SVM** and by using the KKT Conditions.

$$\begin{aligned} &\min_{\alpha} \frac{1}{2} \sum_{(i,j),(k,l)\in I} \alpha_{ij} \alpha_{kl} y_{ij} y_{kl} \left\langle P\left(Q(x_i, x_j)\right), P\left(Q(x_k, x_l)\right) \right\rangle - \sum_{(i,j)\in I} \alpha_{ij} \\ &\text{s.t. } \sum_{(i,j)\in I} \alpha_{ij} y_{ij} = 0 \\ &\qquad 0 \leq \alpha_{ij} \leq C \quad \text{for all } (i,j) \in I \end{aligned} \tag{4.5}$$

A straightforward modification of Lemma 2.21 shows that there is always a solution (w^*, b^*, ξ^*) of (4.3). Moreover, an extension of Theorem 2.25 yields that each solution leads to the same w^*. Using the KKT Conditions one obtains that

$$w^* = \sum_{(i,j)\in I} \alpha_{ij}^* y_{ij} P(Q(x_i, x_j)) \tag{4.6}$$

for every solution α^* of (4.5). Equation (4.6) provides the following interesting observation which is connected to the Semiparametric Representer Theorem 2.26.

Lemma 4.3. *Assume that (w^*, b^*, ξ^*) is a solution of (4.3). Then, w^* is an element of the subspace corresponding to the chosen mapping Q and P. For*

instance, $w^* \in X \otimes_S X$ *if* $Q = Q_T$ *and* $P = P_{TS}$.

Due to (4.4) and (4.6) any solution α^* of (4.5) can be used and one obtains

$$g(u,v) = \sum_{(i,j)\in I} \alpha_{ij}^* y_{ij} \left\langle P\left(Q(x_i,x_j)\right), P\left(Q(u,v)\right)\right\rangle + b^*. \tag{4.7}$$

Again, b^* can be calculated as in ordinary SVMs (see Remark 2.24).

Lemma 4.4. *Assuming that a projection onto a symmetric subspace is used, that is to say* $P = P_{DS}$ *or* $P = P_{TS}$. *Moreover, let the decision function* g *be given by* (4.7). *Then,* g *is symmetric with respect to the order of the input examples.*

Proof. For any $u, v \in X$ Lemma 4.2 gives

$$P_{DS}(Q_D(u,v)) = P_{DS}(Q_D(v,u)).$$

Hence,

$$\begin{aligned} g(u,v) &= \sum_{(i,j)\in I} \alpha_{ij}^* y_{ij} \left\langle P_{DS}\left(Q_D(x_i,x_j)\right), P_{DS}\left(Q_D(u,v)\right)\right\rangle + b^* \\ &= \sum_{(i,j)\in I} \alpha_{ij}^* y_{ij} \left\langle P_{DS}\left(Q_D(x_i,x_j)\right), P_{DS}\left(Q_D(v,u)\right)\right\rangle + b^* = g(v,u) \end{aligned}$$

follows. The same arguments hold for P_{TS} and Q_T. □

Remark 4.5. *Since Lemma 4.2 implies*

$$P_{DA}(Q_D(u,v)) = -P_{DA}(Q_D(v,u)) \quad \textit{and} \quad P_{TA}(Q_T(u,v)) = -P_{TA}(Q_T(v,v))$$

Lemma 4.4 does not hold if an asymmetric subspace is used. Therefore, the proposed approach does not lead to a symmetric decision function in those cases. However, if the approach is modified in such a way that e *is replaced by an even function, in other words* $e(z) = e(-z)$ *for all* $z \in X \oplus X$ *or* $z \in X \otimes X$, *then one gets a symmetric decision function when any of the projections defined in* (4.1) *is used. One possible way to construct such an even function* e *is presented within the following subsection.*

4.1.2 Nonlinear Pairwise SVMs

This subsection extends the approach of Subsection 4.1.1 to the use of kernels (see Section 2.3).

Above, the decision function g was decomposed by $g = e \circ P \circ Q$ (4.2). A first way to apply the kernel trick is to modify the functions Q and P. Hence, one would decompose g into

$$g = \tilde{e} \circ \hat{P} \circ \hat{Q}$$

with two possibilities of the mapping $\hat{Q} : X \times X \to \mathcal{D}$, namely:

- $\hat{Q} := \hat{Q}_D$ with $\hat{Q}_D(a, b) := (\psi(a), \psi(b))$ for all $a, b \in X$ and $\mathcal{D} := \mathcal{H} \oplus \mathcal{H}$,
- $\hat{Q} := \hat{Q}_T$ with $\hat{Q}_T(a, b) := \psi(a) \otimes \psi(b)$ for all $a, b \in X$ and $\mathcal{D} := \mathcal{H} \otimes \mathcal{H}$.

Here, $\psi : X \to \mathcal{H}$ denotes some (ordinary) Hilbert space mapping.

Remark 4.6. *If $\mathcal{H}$ is a Hilbert space, then $\mathcal{H} \oplus \mathcal{H}$ is a Hilbert space, too. However, $\mathcal{H} \otimes \mathcal{H}$ is not a Hilbert space in general, but it is a pre-Hilbert space, in other words the canonical scalar product exists but the space is incomplete. For the kernel trick it does not matter that $\mathcal{H} \otimes \mathcal{H}$ is incomplete. It only matters that there is some scalar product. Therefore, this dissertation does not distinct between pre-Hilbert spaces and Hilbert spaces.*

Similar to (4.1) one defines

$$\begin{aligned}
\hat{P}_{DS} &: \mathcal{H} \oplus \mathcal{H} \to \mathcal{H} \oplus_S \mathcal{H}, \quad \hat{P}_{DS}(\hat{a}, \hat{b}) := \frac{1}{2}(\hat{a} + \hat{b}, \hat{b} + \hat{a}) \\
\hat{P}_{DA} &: \mathcal{H} \oplus \mathcal{H} \to \mathcal{H} \oplus_A \mathcal{H}, \quad \hat{P}_{DA}(\hat{a}, \hat{b}) := \frac{1}{2}(\hat{a} - \hat{b}, \hat{b} - \hat{a}) \\
\hat{P}_{TS} &: \mathcal{H} \otimes \mathcal{H} \to \mathcal{H} \otimes_S \mathcal{H}, \qquad \hat{P}_{TS}(\hat{z}) := \frac{1}{2}(\hat{z} + \bar{\hat{z}}) \\
\hat{P}_{TA} &: \mathcal{H} \otimes \mathcal{H} \to \mathcal{H} \otimes_A \mathcal{H}, \qquad \hat{P}_{TA}(\hat{z}) := \frac{1}{2}(\hat{z} - \bar{\hat{z}})
\end{aligned} \tag{4.8}$$

with $\hat{a}, \hat{b} \in \mathcal{H}$ and $\hat{z}, \bar{\hat{z}} \in \mathcal{H} \otimes \mathcal{H}$ where $\bar{\hat{z}}$ is the adjoint of $\hat{z}$.

Finally, the function $\tilde{e}_D$ maps $\mathcal{H} \oplus \mathcal{H} \to \mathbb{R}$ and $\tilde{e}_T$ maps $\mathcal{H} \otimes \mathcal{H} \to \mathbb{R}$. Additionally, Lemmas 4.2 and 4.4 can be easily extended to the projections presented above. However, Remark 4.5 would still hold in this case independently of the chosen mapping ψ. In other words, using a asymmetric subspace would still not lead to a symmetric pairwise decision function. In order to overcome this issue, one can decompose the function g another time by

$$g = \hat{e} \circ \Psi \circ \hat{P} \circ \hat{Q}. \tag{4.9}$$

Here, $\Psi_D : \mathcal{H} \oplus \mathcal{H} \to \mathcal{Z}_D$ for some Hilbert space $\mathcal{Z}_D$ and $\Psi_T : \mathcal{H} \otimes \mathcal{H} \to \mathcal{Z}_T$ for some Hilbert space $\mathcal{Z}_T$. Finally, $\hat{e}_D : \mathcal{Z}_D \to \mathbb{R}$ and $\hat{e}_T : \mathcal{Z}_T \to \mathbb{R}$ are defined.

Note that the Hilbert space mapping ψ introduced by $\hat{Q}$ is applied before using some projection. Moreover, $\hat{Q}$ maps to the direct sum or tensor product of two Hilbert spaces. In contrast to this, the Hilbert space mapping Ψ is applied after using some projection $\hat{P}$. Furthermore, Ψ maps to an arbitrary Hilbert space. Hence, there is a difference between both presented ways of applying the kernel trick.

Similar as in Subsection 4.1.1 let $\hat{e}$ be defined by

$$\hat{e}(\hat{z}) := \langle w^*, \hat{z} \rangle + b^*$$

where the parameters $w^* \in \mathcal{Z}$ and $b^* \in \mathbb{R}$ are obtained by a solution (w^*, b^*, ξ^*) of the following quadratic program

$$\begin{aligned} &\min_{w,b,\xi} \frac{1}{2} \langle w, w \rangle + C \sum_{(i,j)\in I} \xi_{ij} \\ &\text{s.t. } y_{ij} \left(\left\langle w, \Psi\left(\hat{P}\left(\hat{Q}(x_i, x_j)\right)\right)\right\rangle + b \right) \geq 1 - \xi_{ij} \quad \text{for all } (i,j) \in I \\ &\qquad \xi_{ij} \geq 0 \quad \text{for all } (i,j) \in I \end{aligned} \tag{4.10}$$

with $I \subseteq M \times M$. Hence, for the pairwise decision function given by (4.9) one obtains

$$g(u,v) = \left\langle w^*, \Psi\left(\hat{P}(\hat{Q}(u,v))\right)\right\rangle + b^*. \tag{4.11}$$

Again, by using the Semiparametric Representer Theorem 2.26 and the KKT theory one can determine a solution of (4.10) by solving the following **pairwise SVM**.

$$\begin{aligned} &\min_{\alpha} \frac{1}{2} \sum_{(i,j),(k,l)\in I} \alpha_{ij}\alpha_{kl}y_{ij}y_{kl} \left\langle \Psi\left(\hat{P}\left(\hat{Q}(x_i, x_j)\right)\right), \Psi\left(\hat{P}\left(\hat{Q}(x_k, x_l)\right)\right)\right\rangle - \sum_{(i,j)\in I} \alpha_{ij} \\ &\text{s.t. } \sum_{(i,j)\in I} \alpha_{ij} y_{ij} = 0 \\ &\qquad 0 \leq \alpha_{ij} \leq C \quad \text{for all } (i,j) \in I. \end{aligned} \tag{4.12}$$

The scalar products in Equations (4.11) and (4.12) motivate to introduce **outer kernels** so that the kernel trick can be applied. To this end, it is assumed that $\mathfrak{a}_1, \mathfrak{b}_1 \in \mathcal{H} \oplus \mathcal{H}$ and $\mathfrak{a}_2, \mathfrak{b}_2 \in \mathcal{H} \otimes \mathcal{H}$. This yields,

$$\begin{aligned} \kappa_D &: (\mathcal{H} \oplus \mathcal{H}) \times (\mathcal{H} \oplus \mathcal{H}) \quad \text{with} \quad \kappa_D(\mathfrak{a}_1, \mathfrak{b}_1) := \langle \Psi_D(\mathfrak{a}_1), \Psi_D(\mathfrak{b}_1) \rangle, \\ \kappa_T &: (\mathcal{H} \otimes \mathcal{H}) \times (\mathcal{H} \otimes \mathcal{H}) \quad \text{with} \quad \kappa_T(\mathfrak{a}_2, \mathfrak{b}_2) := \langle \Psi_T(\mathfrak{a}_2), \Psi_T(\mathfrak{b}_2) \rangle. \end{aligned}$$

By using an outer kernel κ and following similar steps as in Subsection 4.1.1 one

can write g as

$$g(u,v) := \sum_{(i,j)\in I} \alpha_{ij}^* y_{ij} \kappa\left(\hat{P}\left(\hat{Q}(x_i,x_j)\right), \hat{P}\left(\hat{Q}(u,v)\right)\right) + b^*. \tag{4.13}$$

Lemma 4.4 can be easily extended to the use of outer kernels. Additionally, similar as stated in Remark 4.5 a pairwise decision function of the form (4.13) will be not symmetric in general if $\hat{P}_{DA}$ or $\hat{P}_{TA}$ are used. However, for special choices of the outer kernel κ one obtains:

Lemma 4.7. *Let the outer kernel κ have the property*

$$\kappa(\mathfrak{a},\mathfrak{b}) = \kappa(\mathfrak{a},-\mathfrak{b}) \tag{4.14}$$

for all $\mathfrak{a},\mathfrak{b} \in \mathcal{H}\oplus\mathcal{H}$ (if $\kappa=\kappa_D$) or for all $\mathfrak{a},\mathfrak{b}\in\mathcal{H}\otimes\mathcal{H}$ (if $\kappa=\kappa_T$). Then, any function g defined by (4.13) *is symmetric.*

Proof. If $\kappa=\kappa_D$ and $P=P_{DS}$, or if $\kappa=\kappa_T$ and $P=P_{TS}$ the result follows by a small modification of Lemma 4.4. Now, let $\kappa=\kappa_D$ and $P=P_{DA}$ be chosen. Then, one obtains

$$\begin{aligned} g(u,v) &= \sum_{(i,j)\in I} \alpha_{ij}^* y_{ij} \kappa_D\left(\hat{P}_{DA}\left(\hat{Q}_D(x_i,x_j)\right), \hat{P}_{DA}\left(\hat{Q}_D(u,v)\right)\right) + b^* \\ &= \sum_{(i,j)\in I} \alpha_{ij}^* y_{ij} \kappa_D\left(\hat{P}_{DA}\left(\hat{Q}_D(x_i,x_j)\right), -\hat{P}_{DA}\left(\psi(v),\psi(u)\right)\right) + b^* \end{aligned}$$

and by (4.14) one further obtains

$$\begin{aligned} &= \sum_{(i,j)\in I} \alpha_{ij}^* y_{ij} \kappa_D\left(\hat{P}_{DA}\left(\hat{Q}_D(x_i,x_j)\right), \hat{P}_{DA}\left(\hat{Q}_D(v,u)\right)\right) + b^* \\ &= g(v,u). \end{aligned}$$

The case $\kappa=\kappa_T$ and $P=P_{TA}$ follows the same pattern. □

Note that condition (4.14) does not hold for an outer linear kernel or for an outer RBF kernel but it does hold for any outer homogeneous polynomial kernel of even degree.

4.2 Evaluating Pairwise Kernel Function Values

Throughout this section kernel functions of the form

$$k : X \times X \to \mathbb{R}, \quad k(a,b) := \langle \psi(a), \psi(b) \rangle$$

for $a, b \in X$ are needed. In pairwise classification k is called **standard kernel**.

In Subsection 4.1.2 several Hilbert space mappings and outer kernels are introduced. The question arises how an outer kernel function can be evaluated. Here, **pairwise kernel functions** $K : (X \times X) \times (X \times X) \to \mathbb{R}$ are defined and the evaluation of several pairwise kernel functions which can be obtained by selecting an outer kernel, a standard kernel and an appropriate projection (see Subsection 4.1.2) is discussed. In other words, K can be written as

$$K((a,b),(c,d)) := \kappa\left(\hat{P}\left(\hat{Q}(a,b)\right), \hat{P}\left(\hat{Q}(c,d)\right)\right).$$

Now, the calculation of pairwise kernels K for three different outer kernels namely, a linear outer kernel, a polynomial outer kernel, and a RBF outer kernel is discussed. Let a, b, c, d be elements of X. This dicussion starts with a linear kernel κ, especially with $\kappa = \kappa_D$ and $\hat{P} = \hat{P}_{DS}$. Then,

$$\begin{aligned}
&K((a,b),(c,d)) \\
=&\kappa_D\left(\hat{P}_{DS}\left(\hat{Q}_D(a,b)\right), \hat{P}_{DS}\left(\hat{Q}_D(c,d)\right)\right) \\
=&\left\langle \Psi\left(\hat{P}_{DS}\left(\hat{Q}_D(a,b)\right)\right), \Psi\left(\hat{P}_{DS}\left(\hat{Q}_D(c,d)\right)\right)\right\rangle \\
=&\left\langle \hat{P}_{DS}\left(\psi(a),\psi(b)\right), \hat{P}_{DS}\left(\psi(c),\psi(d)\right)\right\rangle \\
=&\frac{1}{4}\left\langle (\psi(a)+\psi(b),\psi(a)+\psi(b)), (\psi(c)+\psi(d),\psi(c)+\psi(d))\right\rangle \\
=&\frac{1}{4}\left\langle \psi(a)+\psi(b),\psi(c)+\psi(d)\right\rangle + \frac{1}{4}\left\langle \psi(a)+\psi(b),\psi(c)+\psi(d)\right\rangle \\
=&\frac{1}{2}\left(\langle \psi(a),\psi(c)\rangle + \langle \psi(b),\psi(d)\rangle + \langle \psi(a),\psi(d)\rangle + \langle \psi(b),\psi(c)\rangle\right) \\
=&\frac{1}{2}\left(k(a,c)+k(b,d)+k(a,d)+k(b,c)\right).
\end{aligned}$$

For example, for a homogeneous polynomial standard kernel of degree p one would get

$$K\left((a,b),(c,d)\right) = \frac{1}{2}\left(\langle a,c\rangle^p + \langle b,d\rangle^p + \langle a,d\rangle^p + \langle b,c\rangle^p\right).$$

For a linear outer kernel κ_T and $\hat{P} = \hat{P}_{TS}$ one obtains

$$\begin{aligned}
&K((a,b),(c,d)) \\
=&\kappa_T\left(\hat{P}_{TS}\left(\hat{Q}_T(a,b)\right),\hat{P}_{TS}\left(\hat{Q}_T(c,d)\right)\right) \\
=&\left\langle\Psi\left(\hat{P}_{TS}\left(\hat{Q}_T(a,b)\right)\right),\Psi\left(\hat{P}_{TS}\left(\hat{Q}_T(c,d)\right)\right)\right\rangle \\
=&\left\langle\hat{P}_{TS}\left(\psi(a)\otimes\psi(b)\right),\hat{P}_{TS}\left(\psi(c)\otimes\psi(d)\right)\right\rangle \\
=&\frac{1}{4}\left\langle\left(\psi(a)\otimes\psi(b)+\psi(b)\otimes\psi(a)\right),\left(\psi(c)\otimes\psi(d)+\psi(d)\otimes\psi(c)\right)\right\rangle \\
=&\frac{1}{4}\left(\langle\psi(a)\otimes\psi(b),\psi(c)\otimes\psi(d)\rangle+\langle\psi(a)\otimes\psi(b),\psi(d)\otimes\psi(c)\rangle\right. \\
&\qquad\left.+\langle\psi(b)\otimes\psi(a),\psi(c)\otimes\psi(d)\rangle+\langle\psi(b)\otimes\psi(a),\psi(d)\otimes\psi(c)\rangle\right) \\
=&\frac{1}{2}\left(k(a,c)k(b,d)+k(a,d)k(b,c)\right).
\end{aligned}$$

As $\hat{P}_{DA}$ and $\hat{P}_{TA}$ do not lead to a symmetric decision function for a linear outer kernel (see Lemma 4.7), these cases are not discussed.

Now, a homogeneous polynomial outer kernel κ of degree s is selected. Here, only $\kappa = \kappa_T$ and $\hat{P} = \hat{P}_{TA}$ are discussed. All other possible choices of κ and $\hat{P}$ follow the same pattern.

$$\begin{aligned}
&K((a,b),(c,d)) \\
=&\kappa_T\left(\hat{P}_{TA}\left(\hat{Q}_T(a,b)\right),\hat{P}_{TA}\left(\hat{Q}_T(c,d)\right)\right) \\
=&\left\langle\Psi\left(\hat{P}_{TA}\left(\hat{Q}_T(a,b)\right)\right),\Psi\left(\hat{P}_{TA}\left(\hat{Q}_T(c,d)\right)\right)\right\rangle \\
=&\left\langle\Psi\left(\hat{P}_{TA}\left(\psi(a)\otimes\psi(b)\right)\right),\Psi\left(\hat{P}_{TA}\left(\psi(c)\otimes\psi(d)\right)\right)\right\rangle \\
=&\left\langle\Psi\left(\frac{1}{2}\left(\psi(a)\otimes\psi(b)-\psi(b)\otimes\psi(a)\right)\right),\Psi\left(\frac{1}{2}\left(\psi(c)\otimes\psi(d)-\psi(d)\otimes\psi(c)\right)\right)\right\rangle \\
=&\frac{1}{4^s}\left\langle\left(\psi(a)\otimes\psi(b)-\psi(b)\otimes\psi(a)\right),\left(\psi(c)\otimes\psi(d)-\psi(d)\otimes\psi(c)\right)\right\rangle^s \\
=&\frac{1}{2^s}\left(k(a,c)k(b,d)-k(a,d)k(b,c)\right)^s.
\end{aligned}$$

Note that this choice of κ ensures a symmetric decision function if s is even (see Lemma 4.7).

Finally, an RBF outer kernel is selected and $\kappa = \kappa_T, \hat{P} = \hat{P}_{TS}$ is discussed. Again, all other possible choices of κ and $\hat{P}$ would follow similar patterns. However, for $\hat{P} = \hat{P}_{DA}$ or $\hat{P} = \hat{P}_{TA}$ one would not obtain a symmetric pairwise decision function.

$$
\begin{aligned}
& K((a,b),(c,d)) \\
=& \kappa_{TS}\left(\hat{P}_{TS}\left(\hat{Q}_T(a,b)\right), \hat{P}_{TS}\left(\hat{Q}_T(c,d)\right)\right) \\
=& \left\langle \Psi\left(\hat{P}_{TS}\left(\hat{Q}_T(a,b)\right)\right), \Psi\left(\hat{P}_{TS}\left(\hat{Q}_T(c,d)\right)\right)\right\rangle \\
=& \left\langle \Psi\left(\hat{P}_{TS}\left(\psi(a)\otimes\psi(b)\right)\right), \Psi\left(\hat{P}_{TS}\left(\psi(c)\otimes\psi(d)\right)\right)\right\rangle \\
=& \left\langle \Psi\left(\frac{1}{2}\left(\psi(a)\otimes\psi(b)+\psi(b)\otimes\psi(a)\right)\right), \Psi\left(\frac{1}{2}\left(\psi(c)\otimes\psi(d)+\psi(d)\otimes\psi(c)\right)\right)\right\rangle \\
=& \exp\left(\frac{-\sigma^2}{4}\left\|\psi(a)\otimes\psi(b)+\psi(b)\otimes\psi(a)-\psi(c)\otimes\psi(d)-\psi(d)\otimes\psi(c)\right\|^2\right) \\
=& \exp\left(\frac{-\sigma^2}{2}\Big(k(a,a)k(b,b)+k(c,c)k(d,d)+k(a,b)^2+k(c,d)^2\right. \\
& \left.\left.-2k(a,c)k(b,d)-2k(a,d)k(b,c)\Big)\right)
\end{aligned}
$$

Note that all the presented calculations of this section are independent of the chosen standard kernel.

Now, several pairwise kernels which are used throughout the dissertation are defined.

$$K_{PD}((a,b),(c,d)) := (k(a,c)+k(b,d)+r)^p \tag{4.15a}$$
$$K_{PT}((a,b),(c,d)) := (k(a,c)\cdot k(b,d)+r)^p \tag{4.15b}$$
$$K_{DS}((a,b),(c,d)) := \frac{1}{2}\left(k(a,c)+k(a,d)+k(b,c)+k(b,d)\right) \tag{4.15c}$$
$$K_{DA}((a,b),(c,d)) := \frac{1}{4}\left(k(a,c)-k(a,d)-k(b,c)+k(b,d)\right)^2 \tag{4.15d}$$
$$K_{TS}((a,b),(c,d)) := \frac{1}{2}\left(k(a,c)k(b,d)+k(a,d)k(b,c)\right) \tag{4.15e}$$
$$K_{TA}((a,b),(c,d)) := \frac{1}{4}\left(k(a,c)k(b,d)-k(a,d)k(b,c)\right)^2 \tag{4.15f}$$
$$K_{D}((a,b,(c,d)) := K_{DS}((a,b),(c,d)) + K_{DA}((a,b),(c,d)) \tag{4.15g}$$
$$K_{TD}((a,b,(c,d)) := K_{TS}((a,b),(c,d)) + K_{DA}((a,b),(c,d)). \tag{4.15h}$$

In [43] K_{DA} is called **metric learning pairwise kernel** due to its close connection to the Euclidean metric, while K_{TS} is called **tensor learning pairwise kernel**. This dissertation calls K_{DS} **symmetric direct sum pairwise kernel** and the pairwise kernel K_{TA} **asymmetric tensor pairwise kernel**. Moreover, this dissertation calls K_D **direct sum pairwise kernel** and K_{TD} **tensor direct asymmetric sum**

pairwise kernel. Finally, this dissertation calls K_{PD} **pairwise polynomial direct sum kernel** and K_{PT} **pairwise polynomial tensor kernel**. Note that K_{PD} and K_{PT} are neither based on projections nor are they **symmetric pairwise kernels**. A pairwise kernel $K : (X \times X) \times (X \times X) \to \mathbb{R}$ is called symmetric pairwise kernel if

$$K((a,b),(c,d)) = K((a,b),(d,c)) \tag{4.16}$$

holds for all $a, b, c, d \in X$. Obviously, the addition and multiplication of two pairwise symmetric kernels lead to a pairwise symmetric kernel (see Proposition 2.27).

In Equations (4.1) and (4.8) several projections are defined. Those projections are closely connected to the presented pairwise kernels. Obviously, if a linear outer kernel is used, then K_{DS} is obtained by using $\hat{P}_{DS}$, while K_{TS} is obtained by using $\hat{P}_{TS}$. Similarly, if a homogeneous polynomial of degree 2 is used as outer kernel, then K_{DA} is obtained by using $\hat{P}_{DA}$, while K_{TA} is obtained by using $\hat{P}_{TA}$.

Note that all pairwise kernels are defined for an arbitrary standard kernel. In the following, K_{DS}^{lin} denotes the kernel K_{DS} with a linear standard kernel and K_{DS}^{poly} denotes the kernel K_{DS} with a homogenous polynomial of degree 2 as standard kernel. Additionally, the kernels $K_{PD}^{lin}, K_{PT}^{lin}, K_{DA}^{lin}, K_{TS}^{lin}, K_{TA}^{lin}, K_{D}^{lin}, K_{TD}^{lin}$ are defined analogously to K_{DS}^{lin} while the kernels $K_{PD}^{poly}, K_{PT}^{poly}, K_{DA}^{poly}, K_{TS}^{poly}, K_{TA}^{poly}, K_{D}^{poly}, K_{TD}^{poly}$ are defined analogously to K_{DS}^{poly}.

4.3 Pairwise Symmetry, Projections, and Information Loss

Section 4.1 showed how the pairwise symmetry of a decision function can be obtained by means of certain projections. It is well known that a projection is not invertible in general. For instance, let $P_{DS}(a,b)$ be presented instead of (a,b) with $a, b \in X \subseteq \mathbb{R}^n$. Then, it is impossible to obtain the Euclidean distance between a and b by $P_{DS}(a,b)$ as

$$P_{DS}(a+t, b-t) = P_{DS}(a,b) \tag{4.17}$$

holds for all $t \in X$. Hence, there may be some information loss if any of the projections presented in (4.1) or (4.8) are used. This section determines which kind of information is lost. Additionally, it is discussed that this information loss may be a drawback.

Remark 4.8. *This section discusses the information loss of several projections and it is claimed that this information loss may be a drawback, which leads to inferior results. However, there is another point of view on this topic. The projections incorporate invariances into the used learning machine. For instance, if one knows for a certain learning task that the absolute position of the examples contains no information at all, then one should use $\hat{P}_{DA}$ to transfer this knowledge to the learning machine.*

At first, let K_{DS}^{lin} be used. Hence, the projection P_{DS} is applied. Obviously, by definition, P_{DS} contains only information about the midpoint of a and b. In other words,

$$P_{DS}(a, b) = P_{DS}(a + t, b - t)$$

holds for all $t \in X$. Below it is shown that this property is a drawback.

Before, K_{DA}^{lin} is discussed. This yields that the projection P_{DA} is used. It is easy to verify that P_{DA} contains only information about the relative position of a and b. Hence,

$$P_{DA}(a, b) = P_{DA}(a + t, b + t)$$

holds for all $t \in X$.

Now, the application of K_{DS}^{lin} and K_{DA}^{lin} to the (synthetic) checker board task is discussed. In this task the input space is $\mathbb{R}^2$. Furthermore, the examples a and b belong to the same class if and only if $\lfloor a \rfloor = \lfloor b \rfloor$ where the floor operator $\lfloor \cdot \rfloor$ is applied elementwise. Now, let K_{DS}^{lin} be used. Then, P_{DS} is implicitly used. Due to (4.17) one would expect a very bad performance since the midpoint of a and b contains almost no relevant information about the classes. Now, let K_{DA}^{lin} be used. Then, P_{DA} is implicitly used. As stated above, one can obtain the Euclidean distance between a and b by $P_{DA}(a, b)$. For large distances one knows that the examples belong to different classes. For smaller distances this becomes more difficult. Nevertheless, one would expect to achieve a better performance by using K_{DA}^{lin} than by using K_{DS}^{lin}. Empirical evidence is given in Subsection 5.2.1. Note that it is possible to reconstruct (a, b) if $P_{DS}(a, b)$ and $P_{DA}(a, b)$ are known. Hence, it might be interesting, to use the direct sum pairwise kernel

$$K_D := K_{DS} + K_{AS}.$$

However, Subsection 5.2.1 will show that K_D does not lead to good results for the checker board task.

Remark 4.9. *Here, an approach of [23] is mentioned. Within this approach it is proposed to use a representation of* (a, b) *as*

$$\begin{pmatrix} a+b \\ \operatorname{sgn}(a_1 - b_1)(a - b) \end{pmatrix}.$$

Here, a_1, b_1 *denote the first component of the vector* a, b*, respectively. Obviously,* $a + b$ *is connected to* P_{DS} *while* $\operatorname{sgn}(a_1 - b_1)(a - b)$ *is connected to* P_{DA}*. By multiplying* $\operatorname{sgn}(a_1 - b_1)$ *the symmetry is enforced and more general pairwise kernels can be used. For instance,*

$$\bar{K}((a, b), (c, d)) := \left\langle \begin{pmatrix} a+b \\ \operatorname{sgn}(a_1 - b_1)(a - b) \end{pmatrix}, \begin{pmatrix} c+d \\ \operatorname{sgn}(c_1 - d_1)(c - d) \end{pmatrix} \right\rangle.$$

This approach was tested on several datasets. However, first results showed that this approach of enforcing the symmetry of pairwise decision functions is inferior to the approach of using projections for pairwise SVMs.

Above, the information loss of projections based on the direct sum of two vector spaces, namely P_{DS} and P_{DA}, was discussed. Those results can be easily transferred to $\hat{P}_{DS}$ and $\hat{P}_{DA}$. Now, the information loss of projections based on the tensor product of two vector spaces will be discussed. This discussion starts with the following lemma which shows that the use of tensors leads to a loss of information even if no projection is applied. Note that only $X \subseteq \mathbb{R}^n$ is proven here. The extension to arbitrary finite dimensional Hilbert spaces is straightforward. The extension to infinite dimensional Hilbert spaces seems possible. However, such an extension seems not to provide additional insight, but would be much harder to understand. Therefore, those results are not presented.

Lemma 4.10. *For some* $x, y \in \mathbb{R}^n \setminus \{0\}$ *and* $u, v \in \mathbb{R}^n$ *let* $B \in \mathbb{R}^{n \times n}$ *be defined by* $B := x \otimes y = xy^\top$*. Then,* $uv^\top = B$ *holds, if and only if there is some* $\lambda \in \mathbb{R} \setminus \{0\}$ *so that*

$$(u, v) = (\lambda x, \lambda^{-1} y).$$

Proof. Firstly, let $(u, v) = (\lambda x, \lambda^{-1} y)$ be valid for some $\lambda \in \mathbb{R} \setminus \{0\}$. Obviously, it holds that $uv^\top = xy^\top = B$.

Secondly, it is proofen that no other choice of (u, v) exists. As $y \neq 0$ there is $k \in \{1, \ldots, n\}$ with $y_k \neq 0$. Hence, the $k-$th column of B is $y^k x$. Hence, u and x must be linearly dependent. Similar arguments show the linear dependence of v and y. Thus, there are $\lambda_1, \lambda_2 \in \mathbb{R} \setminus \{0\}$ so that $uv^\top = \lambda_1 x (\lambda_2 y)^\top = B$. This is true if and only if $\lambda_1 = \lambda_2^{-1}$. □

In order to discuss the information loss caused by the use of tensors let the examples $u, v \in \mathbb{R}^n$ be given. Lemma 4.10 shows that some information about the norm of the examples is lost if $uv^\top$ is given instead of (u, v). For instance, if one wants to decide whether

$$\lfloor \|u\|_2 \rfloor = \lfloor \|v\|_2 \rfloor$$

holds or not, then one cannot answer this question by means of $uv^\top$.

Remark 4.11. *The information loss on the norm can be reduced if it is known that each example has the same (Euclidean) norm. In this case Lemma 4.10 implies that* $(u, v) = (x, y)$ *or that* $(u, v) = (-x, -y)$*. Moreover, one could additionally enforce the first (or any other) index of each vector to be larger (or smaller) than zero. Then, the lemma implies that* $(u, v) = (x, y)$*.*

Theorem 4.13 will analyze the information loss caused by P_{TS}. Before, a result which is needed to prove Theorem 4.13 is presented.

Lemma 4.12. *Let* $x, y \in \mathbb{R}^n$ *be linear independent. Then, the matrices*

$$xx^\top, yy^\top, xy^\top, \text{ and } yx^\top$$

are linear independent, too.

Proof. The linear independence of x and y yields that there are $i, j \in \{1, \ldots, n\}$ such that

$$\gamma := x_i y_j - x_j y_i \neq 0.$$

W.o.l.g. let $i = 1$ and $j = 2$ be chosen. Now, the linear equation is considered

$$t_1 xx^\top + t_2 xy^\top + t_3 yx^\top + t_4 yy^\top = 0$$

with $t_1, t_2, t_3, t_4 \in \mathbb{R}$. Thus, the four matrices mentioned above are linear independent if the following system of linear equations has only the trivial solution

$$\begin{pmatrix} x_1^2 & x_1 y_1 & x_1 y_1 & y_1^2 \\ x_1 x_2 & x_1 y_2 & x_2 y_1 & y_1 y_2 \\ x_1 x_2 & x_2 y_1 & x_1 y_2 & y_1 y_2 \\ x_2^2 & x_2 y_2 & x_2 y_2 & y_2^2 \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{pmatrix} = 0.$$

The following determinant gives

$$
\begin{aligned}
&\begin{vmatrix} x_1^2 & x_1y_1 & x_1y_1 & y_1^2 \\ x_1x_2 & x_1y_2 & x_2y_1 & y_1y_2 \\ x_1x_2 & x_2y_1 & x_1y_2 & y_1y_2 \\ x_2^2 & x_2y_2 & x_2y_2 & y_2^2 \end{vmatrix} \\
=&\begin{vmatrix} x_1^2 & x_1y_1 & x_1y_1 & y_1^2 \\ x_1x_2 & x_1y_2 & x_2y_1 & y_1y_2 \\ 0 & x_2y_1 - x_1y_2 & x_1y_2 - x_2y_1 & 0 \\ x_2^2 & x_2y_2 & x_2y_2 & y_2^2 \end{vmatrix} \\
=&\begin{vmatrix} x_1^2 & x_1y_1 & 2x_1y_1 & y_1^2 \\ x_1x_2 & x_1y_2 & x_1y_2 + x_2y_1 & y_1y_2 \\ 0 & \gamma & 0 & 0 \\ x_2^2 & x_2y_2 & 2x_2y_2 & y_2^2 \end{vmatrix} \\
=&-\gamma \begin{vmatrix} x_1^2 & 2x_1y_1 & y_1^2 \\ x_1x_2 & x_1y_2 + x_2y_1 & y_1y_2 \\ x_2^2 & 2x_2y_2 & y_2^2 \end{vmatrix} \\
=&-\gamma \left(x_1^3y_2^3 - 3x_1^2y_2^2x_2y_1 + 3x_1y_2x_2^2y_1^2 - x_2^3y_1^3\right) \\
=&-\gamma^4 \neq 0.
\end{aligned}
$$

Hence, the four matrices are linear independent. □

Theorem 4.13. *For some $x, y \in \mathbb{R}^n \setminus \{0\}$ and $u, v \in \mathbb{R}^n$ let $A \in \mathbb{R}^{n\times n}$ be defined as $A := x \otimes y + y \otimes x = xy^\top + yx^\top$. Then, $uv^\top + vu^\top = A$ holds, if and only if there is some $\lambda \in \mathbb{R} \setminus \{0\}$ so that*

$$(u, v) = (\lambda x, \lambda^{-1} y) \qquad \textit{or} \qquad (u, v) = (\lambda y, \lambda^{-1} x).$$

Proof. Obviously, for $(u, v) = (\lambda x, \lambda^{-1} y)$ it holds that $uv^\top + vu^\top = A$ and for $(u, v) = (\lambda y, \lambda^{-1} x)$ it holds that $uv^\top + vu^\top = A$, too. Thus, one needs to prove that no other choice of (u, v) exists.

Firstly, it is assumed that x does not linearly depend on y. Then, for any $w \in \mathbb{R}^n$ with $w \perp \text{span}\{x, y\}$ it holds that $Aw = 0$. Therefore, A must have rank 1 or rank 2. In other words, the image $\text{Im}(A)$ of A must have dimension 1 or 2. Now, this image should be obtained by analyzing the linear independence of

$$z_1 := Ax = \langle x, y\rangle x + \langle x, x\rangle y \neq 0 \quad \text{and} \quad z_2 := Ay = \langle y, y\rangle x + \langle x, y\rangle y \neq 0.$$

Therefore, $rz_1 + sz_2 = 0$ is considered for unknown $r, s \in \mathbb{R}$. This yields

$$\begin{aligned} & r(\langle x, y\rangle x + \langle x, x\rangle y) + s(\langle y, y\rangle x + \langle x, y\rangle y) = 0 \\ \Leftrightarrow \quad & (r\langle x, y\rangle + s\langle y, y\rangle)x + (r\langle x, x\rangle + s\langle x, y\rangle)y = 0. \end{aligned}$$

By the linearly independence of x and y this is equivalent to

$$\begin{pmatrix} \langle x, y\rangle & \langle y, y\rangle \\ \langle x, x\rangle & \langle x, y\rangle \end{pmatrix} \begin{pmatrix} r \\ s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{4.18}$$

For the determinant of the matrix in (4.18) one obtains by the Cauchy Schwarz inequality and the linear independence of x and y

$$\langle x, y\rangle^2 - \langle x, x\rangle\langle y, y\rangle < 0.$$

Hence, $r = s = 0$ is the only solution of (4.18) and z_1 is linearly independent of z_2. This yields that the dimension of $\mathrm{Im}(A)$ is 2 if x and y are linear independent. Moreover, one can conclude that $\mathrm{Im}(A) = \mathrm{span}\{z_1, z_2\} = \mathrm{span}\{x, y\}$. Therefore, u, v must belong to $\mathrm{span}\{x, y\}$ since otherwise $\mathrm{Im}\left(uv^\top + vu^\top\right) \neq \mathrm{Im}(A)$. Thus, with $u = r_1x + s_1y$ and $v = r_2x + s_2y$ one obtains

$$\begin{aligned} A = xy^\top + yx^\top &= (r_1x + s_1y)\,(r_2x + s_2y)^\top + (r_2x + s_2y)\,(r_1x + s_1y)^\top \\ &= (2r_1r_2)xx^\top + (2s_1s_2)yy^\top + (r_1s_2 + r_2s_1)xy^\top + (r_2s_1 + r_1s_2)yx^\top. \end{aligned}$$

Lemma 4.12 shows that the four occurring matrices $xx^\top, yy^\top, xy^\top$, and $yx^\top$ are linearly independent. Hence, by equating the coefficients one obtains the system

$$2r_1r_2 = 0 \quad r_1s_2 + r_2s_1 = 1 \quad 2s_1s_2 = 0$$

with the solution set

$$\{(r_1, s_1, r_2, s_2) = (\lambda, 0, 0, \lambda^{-1})\} \cup \{(r_1, s_1, r_2, s_2) = (0, \lambda, \lambda^{-1}, 0)\}$$

with $\lambda \in \mathbb{R} \setminus \{0\}$. Hence, $(u, v) = (\lambda x, \lambda^{-1}y)$ or $(u, v) = (\lambda y, \lambda^{-1}x)$ if x and y are linear independent.

Secondly, it is assumed that $y = \delta x$ for some $\delta \in \mathbb{R} \setminus \{0\}$. Then, for any $w \in \mathbb{R}^n$ with $w \perp x$ it holds that $Aw = 0$. Hence, A has rank 1 and $\mathrm{Im}(A)$ has dimension 1. Due to $Ax = 2\delta\,\langle x, x\rangle\,x$ it follows that $\mathrm{Im}(A) = \mathrm{span}\{x\}$. Hence, u and v depend linearly on x as otherwise $\mathrm{Im}\left(uv^\top + vu^\top\right) \neq \mathrm{Im}(A)$. Thus, $u = \lambda_1 x$ and

$v = \lambda_2 x$ for some $\lambda_1, \lambda_2 \in \mathbb{R} \setminus \{0\}$. From

$$uv^\top + vu^\top = 2\lambda_1\lambda_2 xx^\top = A = 2\delta xx^\top$$

it follows that $\lambda_1 = \delta/\lambda_2$. Then, one obtains $u = \lambda_1 x = (\delta/\lambda_2)x = \lambda_2^{-1} y$ and $v = \lambda_2 x$, or $u = \lambda_1 x$ and $v = \lambda_2 x = (\delta/\lambda_1)x = \lambda_1^{-1} y$. □

The last theorem shows that if $P_{TS}(ab^\top)$ is known instead of $ab^\top$, then no information is lost except the ordering of a and b, in other words $ab^\top$ is regarded as the same as $ba^\top$ (see also Remark 4.11).

Now, it is shown that some additional information might be lost when $P_{TA}(ab^\top)$ is used instead of $ab^\top$.

Firstly, let $a, b \in \mathbb{R}^2$ be given and let the rotation matrix be defined by

$$R(\theta) := \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \quad \text{for some} \quad \theta \in [0, 2\pi).$$

Then, one gets $P_{TA}\left((R(\theta)a)\,(R(\theta)b)^\top\right) = P_{TA}(ab^\top)$ for all θ. Hence, by $P_{TA}(ab^\top)$ one cannot answer whether a, b belong to the same orthant, in other words whether $a_1 b_1 > 0$ and $a_2 b_2 > 0$.

Secondly, let $a, b \in \mathbb{R}^n$ be given and be linearly dependent. Then, $P_{TA}(ab^\top) = 0$ holds.

Hence, kernels which enforce the symmetry by using an asymmetric tensor subspace seem to be inferior to pairwise kernels which enforce the symmetry by using a symmetric tensor subspace. Therefore, the loss of information by P_{TA} is not further analyzed.

4.4 Symmetric Training Sets

Section 4.1 presented an approach how a symmetric decision function by means of certain projections can be obtained. For pairwise SVMs another approach for ensuring a symmetric decision function is known. It is not based on symmetric pairwise kernels but on specially structured training sets. Obviously, if a symmetric pairwise kernel (4.16) is used, then one can exclude the pair (b, a) from the training set if the pair (a, b) is contained. Now, let a **symmetric training set** for the training of pairwise SVMs be given, that is if (a, b) is a training pair then (b, a) is

a training pair, too. In this setting, one obtains a symmetric decision function for more general pairwise kernels (see Lemma 4.14 below and [23, 45]).

Let $x_i \in \mathbb{R}^n$ with $i \in M := \{1, \dots, m\}$ be given and let $I \subseteq M \times M$ be given with $(i,j) \in I \Rightarrow (j,i) \in I$. Obviously, I leads to a symmetric training set. Additionally, let $I_R \subseteq I$ be defined by $I_R := \{(i,i) | (i,i) \in I\}$ and $I_N := I \setminus I_R$. Furthermore, let $K : (X \times X) \times (X \times X) \to \mathbb{R}$ be a pairwise kernel with

$$K((a,b),(c,d)) = K((b,a),(d,c)). \tag{4.19}$$

Note that any kernel can be seen as a real scalar product. For a standard kernel this yields $k(a,b) = k(b,a)$ for all $a, b \in X$. For a pairwise kernel this yields $K((a,b),(c,d)) = K((c,d),(a,b))$ for all $a, b, c, d \in X$. Therefore, (4.19) holds for any symmetric pairwise kernel. Additionally, (4.19) holds for other pairwise kernels, for instance for $K = K_{PD}$ (4.15a) or $K = K_{PT}$ (4.15b). Now, let the dual pairwise SVM be considered

$$\begin{aligned} \min_{\alpha} h(\alpha) := \frac{1}{2} \sum_{(i,j),(k,l) \in I} \alpha_{ij} \alpha_{kl} y_{ij} y_{kl} K((x_i, x_j), (x_k, x_l)) - \sum_{(i,j) \in I} \alpha_{ij} \\ \text{s.t. } 0 \leq \alpha_{ij} \leq C \quad \text{for all } (i,j) \in I_N \\ 0 \leq \alpha_{ii} \leq 2C \quad \text{for all } (i,i) \in I_R \\ \sum_{(i,j) \in I} y_{ij} \alpha_{ij} = 0. \end{aligned} \tag{4.20}$$

Lemma 4.14. *Let* (4.19) *be valid. Then, there is a solution* $\hat{\alpha}$ *of* (4.20) *with* $\hat{\alpha}_{ij} = \hat{\alpha}_{ji}$ *for all* $(i,j) \in I$. *Such a solution symmetric is called symmetric.*

Proof. The Weierstrass-Theorem implies that there is a solution α^* of (4.20). Let another feasible point $\tilde{\alpha}$ of (4.20) be defined by

$$\tilde{\alpha}_{ij} := \alpha^*_{ji} \quad \text{for all } (i,j) \in I.$$

For easier notation let $K_{ij,kl} := K((x_i, x_j), (x_k, x_l))$ be defined. Then,

$$2h(\tilde{\alpha}) = \sum_{(i,j),(k,l) \in I} \alpha^*_{ji} \alpha^*_{lk} y_{ij} y_{kl} K_{ij,kl} - 2 \sum_{(i,j) \in I} \alpha^*_{ji}.$$

Note that $y_{ij} = y_{ji}$ holds for all $(i,j) \in I$. By (4.19) one further obtains

$$2h(\tilde{\alpha}) = \sum_{(i,j),(k,l) \in I} \alpha^*_{ji} \alpha^*_{lk} y_{ji} y_{lk} K_{ji,lk} - 2 \sum_{(i,j) \in I} \alpha^*_{ji} = 2h(\alpha^*).$$

The last equality follows by the symmetry of the set I. Hence, $\tilde{\alpha}$ is also a solution of (4.20). From SVM theory it is known that problem (4.20) is convex. By Remark 2.2 it follows that

$$\alpha^{\lambda} := \lambda\alpha^* + (1-\lambda)\tilde{\alpha}$$

solves (4.20) for any $\lambda \in (0,1)$. Thus, $\alpha^{1/2}$ is a symmetric solution. □

In [45] a similar result for regression is presented. However, they conclude by means of $h(\tilde{\alpha}) = h(\alpha^*)$ that any solution has the described symmetry. This does not hold in general and there are counterexamples for it.

Theorem 4.15. *It is assumed that* (4.19) *holds. Then, each solution α of the optimization problem* (4.20) *leads to a symmetric decision function $g : X \times X \to \mathbb{R}$. In other words $g(a,b) = g(b,a)$ holds for any $(a,b) \in X \times X$.*

Proof. For any solution α let $f_\alpha : X \times X \to \mathbb{R}$ be defined by

$$f_\alpha(u,v) := \sum_{(i,j)\in I} \alpha_{ij} y_{ij} K((x_i,x_j),(u,v)).$$

Then, the obtained decision function can be written as

$$g_\alpha(u,v) = f_\alpha(u,v) + c$$

for some appropriate $c \in \mathbb{R}$. Theorem 2.25 shows that if α^1 and α^2 are solutions of (4.20) then $f_{\alpha^1} = f_{\alpha^2}$. According to Lemma 4.14 there is always a solution α^* of (4.20) with $\alpha^*_{ij} = \alpha^*_{ji}$ for all $(i,j) \in I$. Obviously, such a solution leads to symmetric functions f_{α^*} and g_{α^*}. As f_α is the same function for all solutions α of (4.20) one obtains that f_α and therefore g_α are symmetric for all solutions. □

4.5 Connecting Projections and Symmetric Training Sets

In order to obtain a symmetric decision function it was discussed in Sections 4.1 and 4.3 that if projected pairs are presented to a learning machine, then a loss of information may occur. Thereafter, in Section 4.4 symmetric training sets are used in pairwise SVMs to obtain a symmetric decision function. In this approach all the available information is presented to the learning machine. Now, Theorem 4.16 shows that the same decision function is obtained, regardless whether a symmetric

training set is used or a certain projection is made to enforce symmetry. Hence, even if a symmetric training set is presented to a pairwise SVM and no projection is made the same information loss occurs as in the case of projections.

Again, let I be a symmetric training set. Additionally, let J denote a subset of $I \subseteq M \times M$ with maximal cardinality and with the property $(i,j) \in J$ and $j \neq i$ imply $(j,i) \notin J$. Furthermore, $J_R := I_R$ and $J_N := J \setminus J_R$. Moreover, let $\hat{K}$ be defined by $\hat{K}_{ij,kl} := \frac{1}{2}(K_{ij,kl} + K_{ji,kl})$, where K is a pairwise kernel which fulfills (4.19). For instance, if $K = K_{PD}$ with $r = 0, p = 1$ (4.15a) then $\hat{K} = K_{DS}$ (4.15c) or if $K = K_{PT}$ with $r = 0, p = 1$ (4.15b) then $\hat{K} = K_{TS}$ (4.15e). In this section let the optimization problem be considered

$$\begin{aligned} \min_{\beta} \quad & \frac{1}{2} \sum_{(i,j),(k,l) \in J} \beta_{ij} \beta_{kl} y_{ij} y_{kl} \hat{K}_{ij,kl} - \sum_{(i,j) \in J} \beta_{ij} \\ \text{s.t. } & 0 \leq \beta_{ij} \leq 2C \quad \text{for all } (i,j) \in J \\ & \sum_{(i,j) \in J} y_{ij} \beta_{ij} = 0. \end{aligned} \tag{4.21}$$

The following theorem shows that both approaches indeed lead to the same decision function.

Theorem 4.16. *Let I and J be defined as above and let the functions $f_\alpha : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ and $h_\beta : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be defined by*

$$\begin{aligned} f_\alpha(a,b) &:= \sum_{(i,j) \in I} \alpha_{ij} y_{ij} K((x_i, x_j), (a,b)), \\ h_\beta(a,b) &:= \sum_{(i,j) \in J} \beta_{ij} y_{ij} \hat{K}((x_i, x_j), (a,b)), \end{aligned}$$

where α is a feasible point of (4.20) *and β is a feasible point of* (4.21)*. Then, for any solution α^* of* (4.20) *and for any solution β^* of* (4.21) *it holds that $f_{\alpha^*} = h_{\beta^*}$.*

Proof. Due to Lemma 4.14 and Theorem 4.15 one can assume that α^* is a symmetric solution of (4.20). Let

$$\bar{\alpha}_{ij} := \begin{cases} \beta^*_{ij}/2 & \text{if } (i,j) \in J_N \\ \beta^*_{ij} & \text{if } (i,j) \in J_R \\ \beta^*_{ji}/2 & \text{else} \end{cases} \quad \text{and} \quad \bar{\beta}_{ij} := \begin{cases} \alpha^*_{ij} + \alpha^*_{ji} & \text{if } (i,j) \in J_N \\ \alpha^*_{ij} & \text{if } (i,j) \in J_R \end{cases}$$

be defined. Obviously, $\bar{\alpha}$ is a feasible point of (4.20) and $\bar{\beta}$ is a feasible point of

(4.21). Then, by $\alpha^*_{ij} = \alpha^*_{ji}$ one obtains for

$$\begin{aligned}(i,j) \in J_N: \qquad \bar{\beta}_{ij}\hat{K}_{ij,kl} &= \frac{\bar{\beta}_{ij}}{2}(K_{ij,kl} + K_{ji,kl}) = \frac{\alpha^*_{ij} + \alpha^*_{ji}}{2}(K_{ij,kl} + K_{ji,kl}) \\ &= \alpha^*_{ij}K_{ij,kl} + \alpha^*_{ji}K_{ji,kl} \\ (i,i) \in J_R: \qquad \bar{\beta}_{ii}\hat{K}_{ii,kl} &= \frac{\bar{\beta}_{ii}}{2}(K_{ii,kl} + K_{ii,kl}) = \alpha^*_{ii}K_{ii,kl}\end{aligned}$$

This implies $f_{\alpha^*} = h_{\bar{\beta}}$. Additionally, one obtains for

$$\begin{aligned}(i,j) \in J_N: \quad \bar{\alpha}_{ij}K_{ij,kl} + \bar{\alpha}_{ji}K_{ji,kl} &= \frac{\beta^*_{ij}}{2}(K_{ij,kl} + K_{ji,kl}) = \beta^*_{ij}\hat{K}_{ij,kl} \\ (i,i) \in J_R: \quad \bar{\alpha}_{ii}K_{ii,kl} &= \frac{\beta^*_{ii}}{2}(K_{ii,kl} + K_{ii,kl}) = \beta^*_{ii}\hat{K}_{ii,kl}.\end{aligned}$$

Hence, $f_{\bar{\alpha}} = h_{\beta^*}$ follows.

In a second step it is proven that $\bar{\alpha}$ and $\bar{\beta}$ are solutions of problem (4.20) and (4.21), respectively. To this end, note that for any solution of (4.20) or (4.21) a corresponding Karush-Kuhn-Tucker (KKT) point exists and, vice versa, that each KKT point corresponds to a solution (see Corollary 2.7). Therefore, the KKT systems of both problems are compared. The KKT system of (4.20) is

$$\sum_{(k,l)\in I} y_{ij}y_{kl}\alpha_{kl}K_{ij,kl} - 1 - u_{ij} + v_{ij} + wy_{ij} = 0 \quad \text{for all } (i,j) \in I$$

$$\sum_{(i,j)\in I} y_{ij}\alpha_{ij} = 0$$

$$\begin{aligned} u_{ij} \geq 0 \quad &\text{for all } (i,j) \in I & \qquad v_{ij} \geq 0 \quad &\text{for all } (i,j) \in I \\ u_{ij}\alpha_{ij} = 0 \quad &\text{for all } (i,j) \in I & v_{ij}(C - \alpha_{ij}) = 0 \quad &\text{for all } (i,j) \in I_N \\ v_{ii}(2C - \alpha_{ii}) = 0 \quad &\text{for all } (i,i) \in I_R & C \geq \alpha_{ij} \geq 0 \quad &\text{for all } (i,j) \in I_N \\ 2C \geq \alpha_{ii} \geq 0 \quad &\text{for all } (i,i) \in I_R. \end{aligned}$$

Accordingly, the KKT system of problem (4.21) is

$$\sum_{(k,l)\in I} y_{ij}y_{kl}\beta_{kl}\hat{K}_{ij,kl} - 1 - \lambda_{ij} + \mu_{ij} + \kappa y_{ij} = 0 \quad \text{for all } (i,j) \in J$$

$$\sum_{(i,j)\in J} y_{ij}\beta_{ij} = 0$$

$$\begin{aligned} \lambda_{ij} \geq 0 \quad &\text{for all } (i,j) \in J & \qquad \mu_{ij} \geq 0 \quad &\text{for all } (i,j) \in J \\ \lambda_{ij}\beta_{ij} = 0 \quad &\text{for all } (i,j) \in J & \mu_{ij}(2C - \beta_{ij}) = 0 \quad &\text{for all } (i,j) \in J \\ 2C \geq \beta_{ij} \geq 0 \quad &\text{for all } (i,j) \in J. \end{aligned}$$

Note that by exchanging the summation index in the definition of f_α from (i,j) to (k,l) one can rewrite the first line of the first KKT system by

$$y_{ij} f_\alpha(x_i, x_j) - 1 - u_{ij} + v_{ij} + w y_{ij} = 0 \quad \text{for all } (i,j) \in I.$$

Analogously, one obtains for the first line of the second KKT system

$$y_{ij} h_\beta(x_i, x_j) - 1 - \lambda_{ij} + \mu_{ij} + \kappa y_{ij} = 0 \quad \text{for all } (i,j) \in J.$$

Let $(\alpha^*, u^*, v^*, w^*)$ be a KKT point of problem (4.20) and $(\beta^*, \lambda^*, \mu^*, \kappa^*)$ be a KKT point of problem (4.21). Moreover, one defines

$$\begin{aligned} \bar{\lambda}_{ij} &:= u^*_{ij} \quad \text{for all } (i,j) \in J \\ \bar{\mu}_{ij} &:= v^*_{ij} \quad \text{for all } (i,j) \in J \\ \bar{\kappa} &:= w^* \end{aligned}$$

and

$$\begin{aligned} \bar{u}_{ij} &:= \begin{cases} \lambda^*_{ij} & \text{for all } (i,j) \in J, \\ \lambda^*_{ji} & \text{for all } (i,j) \in I \setminus J, \end{cases} \\ \tilde{v}_{ij} &:= \begin{cases} \mu^*_{ij} & \text{for all } (i,j) \in J, \\ \mu^*_{ji} & \text{for all } (i,j) \in I \setminus J \end{cases} \\ \bar{w} &:= \kappa^*. \end{aligned}$$

Then, using $h_{\bar{\beta}} = f_{\alpha^*}$ it can be verified that $(\bar{\alpha}, \bar{u}, \bar{v}, \bar{w})$ is a KKT point of (4.20). Similarly, it can be shown that $(\bar{\beta}, \bar{\lambda}, \bar{\mu}, \bar{\kappa})$ is a KKT point of (4.21), too. Hence, $\bar{\alpha}$ is a solution of (4.20) and $\bar{\beta}$ is a solution of (4.21).

Theorem 2.25 implies that independently of the chosen solution α^* of (4.20) and β^* of (4.21) one obtains the same f_{α^*} and h_{β^*}, respectively. This implies $f_{\alpha^*} = f_{\bar{\alpha}} = h_{\bar{\beta}} = h_{\beta^*}$. □

Note that the proof shows how to construct a solution of (4.20) by means of a solution of (4.21) and vice versa.

Remark 4.17. *In Theorem 4.16 any bias is disregarded. Let $(\alpha^*, u^*, v^*, w^*)$ denote a KKT point of problem* (4.20) *and let $(\beta^*, \lambda^*, \mu^*, \kappa^*)$ denote a KKT point of problem* (4.21) *then w^* and κ^* can be used as bias (see Lemma 2.22). Moreover,*

the proof showed that any optimal bias of (4.20) *is an optimal bias of* (4.21) *and vice versa.*

As both approaches lead to the same decision function one should analyze whether one approach is computationally cheaper than the other. The needed training time of SMO (empirically) scales quadratically with the number of training points [33]. For symmetric training sets the number of training pairs is nearly doubled compared to the number in the case of symmetric kernels. Simultaneously, the evaluation of symmetric kernels is computationally four times more expensive compared to the corresponding non symmetric kernel (see Section 5.1 and Equation (4.15)). Hence, by this argumentation one expects that both approaches need the same training time. However, in Table 4.1 it is demonstrated that the approach of using projections or symmetric kernels is significantly faster. Therefore, for pairwise SVMs the approach of using certain projections supersedes the approach of using symmetric training sets. Note that to generate the results in Table 4.1 the technique of caching the standard kernel values as described in Chapter 5 is used for both approaches.

Number of examples	Symmetric Training Set (t in hh:mm)	Symmetric Kernel
500	0:03	0:01
1000	0:46	0:17
1500	3:26	0:56
2000	9:44	2:58
2500	23:15	6:20

Table 4.1: Training time of symmetric training sets vs. training time of symmetric kernels. The technique described in Section 5 is also used for those measurements.

There is another argumentation for the needed training time of a SVM. There, one assumes that the training time scales cubically with the number of support vectors. If symmetric training sets are used and α^* is a symmetric solution, then the number of support vectors is nearly doubled compared to the use of projections. Using this argumentation yields that the training time is doubled if a symmetric training set is used instead of projections.

4.6 Remarks

In this section two remarks concerning pairwise SVMs and extensions of the presented results are given.

At first, it is discussed whether different penalty parameters for the positive and negative pairs should be used. In general, pairwise classification tasks are imbalanced classification problems. See [14, 26] as an anchor to this topic. There are several advices of how a class imbalance problem can be tackled. A special kind of class imbalance problem occurs in pairwise classification. This kind is called relative class imbalance. In other words, there are many positive pairs but the relation of positive pairs and negative pairs is small. In this case it is proposed to use different penalty parameters for positive and negative training pairs. In particular, it is proposed to use a larger penalty parameter for the positive pairs than for the negative pairs. However, many different combinations of penalty parameters for positive and negative pairs were tested in Section 5.2. Surprisingly, the obtained results differed only slightly.

At second, pairwise One Class Support Vector Machines (OCSVM) should be mentioned. If a training set consists of only one class and the classification task is to determine whether a new point belongs to this class or not, then a OCSVM can be used. There are two different approaches of OCSVMs. One approach tries to find the minimal sphere around the training points [41]. The other approach tries to separate the training points from the origin using a hyperplane [38]. An extension of the latter approach can be found in [37].

The results of Sections 4.1 and 4.3 can be transferred to OCSVMs easily. Additionally, for OCSVMs which separate the data from the origin one can show that similar results, which where described in Sections 4.4 and 4.5, hold. In other words, using a symmetric training set leads to a symmetric decision function. Moreover, one can connect the use of symmetric training sets and the use of projections. In order to prove those results one follows exactly the same steps as presented in the corresponding parts of this work. This does not offer new insights. Therefore, those results are not presented into detail.

5 Efficient Implementation and Numerical Results

In Section 5.1 several implementation details are presented. Note that the discussed techniques provide a way to train pairwise SVMs with all pairs of significantly more than $1{,}000$ examples within an acceptable time. Afterwards, results of pairwise SVMs for several synthetic and two real world datasets are presented in Section 5.2.

5.1 Implementing Pairwise SVMs Efficiently

In Chapter 4 two different approaches how a symmetric pairwise decision function can be obtained by a SVM are presented. Furthermore, it is shown in Section 4.5 that both approaches lead to the same decision function. Moreover, it is empirically shown in Section 4.5 that the approach using projections is significantly faster. Therefore, only the faster approach is considered in this section.

5.1.1 Caching the Standard Kernel Values

Much effort has been put into solving (non pairwise) SVMs efficiently. One of the most widely used techniques is the sequential minimal optimization (SMO) [34]. A well known implementation of this technique is LIBSVM [13]. For the moment, it is assumed that one wants to solve a pairwise SVM with kernel K_{DA}^{lin} (4.15d). In order to create a training set which can be used by the LIBSVM one could calculate $P_{DA}(a, b)$ for all used training pairs (a, b) explicitly and save those projected pairs in a file for training. However, this approach leads to superfluously large files as all examples are part of many pairs and therefore are saved repeatedly. For example, to store all pairs of $10{,}000$ examples of dimension $1{,}000$ in an ASCII file, one needed

at least 5GB. This situation becomes even worse for other kernels. Therefore, the LIBSVM code was modified.

In a first attempt, the examples were stored in RAM and each standard kernel was calculated on demand. This modification suffered from a bad computational performance. One reason for this seems the empirically known fact that the SMO scales quadratically with the number of training points [34]. Note that for pairwise classification tasks the training points are the training pairs. If all existing pairs are used for training then the number of training pairs grows quadratically with the number of training examples. For instance, if one uses all the existing pairs of m examples, then there are $m\left(m+1\right)/2$ pairs. Hence, the runtime of the LIBSVM scales at least quartically with the number m of used training examples. Using 500 training examples already results in $125{,}250$ training pairs and corresponding pairwise SVMs would need several hours to be solved. Therefore, a technique to reduce the needed training time is presented.

Kernel evaluations are crucial for the performance of LIBSVM. If the whole kernel matrix could be cached one would get a huge increase of speed. Today, this seems impossible for significantly more than $125{,}250$ training pairs as storing the (symmetric) kernel matrix for this number of pairs in double precision needs approximately 59GB. A way of drastically reducing the costs of kernel evaluations is now described. In Section 4.2 several kernels are introduced. For example, let K_{TS} (4.15e) and an arbitrary standard kernel be selected. For a single evaluation of K_{TS} the standard kernel has to be evaluated four times with vectors of $\mathbb{R}^n$. Afterwards, four arithmetic operations are needed. It is easy to see that each standard kernel value is needed for many different evaluations of K_{TS}. In general, it is possible to cache the standard kernel values for all pairs of examples in the training set. For instance, to cache the standard kernel values for $10{,}000$ examples one needs 400MB. Thus, if all standard kernel values are cached, then each kernel evaluation of K_{TS} costs four arithmetic operations. This does not depend on the chosen standard kernel. Using any other pairwise kernel which is presented in (4.15) is at most slightly more expensive. Furthermore, one could free memory by deleting the examples after computing the standard kernel values as the examples are not needed anymore. Additionally, the dimension of the examples does not influence the costs of a single pairwise kernel evaluation in any case. Only the time needed to calculate all standard kernel values depends on the dimension n of the examples. However, if a linear standard kernel is used, then in case of $10{,}000$ examples of dimension $1{,}000$, one needs about 10^{11} operations to calculate all such values. This can be done in less than a minute.

Tables 5.1a and 5.1b compare the training times with and without the described technique. For this measurement examples from the Double Interval Task (cf. Subsection 5.2.2) are used with a constant EPCR of 5, K_{TS}^{lin} as pairwise kernel, a cache size of 100MB, and all pairs are used for training. In each run of Table 5.1a 250 examples are used for different dimensions n. Table 5.1b shows results for different numbers of examples of dimension $n = 500$. The speedup factor by the described technique is up to 130.

Dimension n of examples	Standard kernel (t in mm:ss)	
	not cached	cached
200	2:08	0:07
400	4:31	0:07
600	6:24	0:07
800	9:41	0:08
1000	11:27	0:09

(a) Different dimensions n of examples

Number of examples	Standard kernel (t in mm:ss)	
	not cached	cached
200	0:04	0:00
400	1:05	0:01
600	4:17	0:02
800	12:40	0:06
1000	28:43	0:13

(b) Different numbers of examples

Table 5.1: Training time with and without caching the standard kernel values

5.1.2 Further Implementation Details

Additional to the technique described in Section 5.1 several other modifications are applied to LIBSVM. Here, selected modifications are described.

Firstly, LIBSVM uses a shrinking technique to shorten the needed training time. Due to this shrinking technique LIBSVM needs to evaluate the decision function values for all training pairs to determine whether an optimal point was found or not. The subroutine `reconstruct_gradient` calculates all such decision function values. However, the shrinking technique shuffles the ordering of the training pairs. This leads to a very low cache hit ratio in the subroutine. Hence, the training pairs are reordered before calling this subroutine. This significantly speeds up the subroutine mentioned above and decreases the total training time by $10\% - 15\%$.

Secondly, OpenMP is used to parallelize several loops of LIBSVM. Table 5.2 presents the needed training time and scale factor with respect to the number of used processors. There, all pairs of $1{,}500$ examples of the double interval type data of dimension $n = 500$ are used for training. The used computer has 2 CPUs, where

each CPU consists of 6 cores. The training time decreases with the number of used cores. However, the scaling is smaller than 0.65 for 5 or more cores.

# Cores	t in sec.	scaling	# Cores	t in sec.	scaling
1	1569	1	7	453	0.494
2	844	0.930	8	417	0.470
3	646	0.810	9	412	0.423
4	536	0.732	10	413	0.380
5	488	0.643	11	406	0.351
6	472	0.554	12	395	0.331

Table 5.2: Scaling of LIBSVM using OpenMP

Thirdly, Section 5.1 shows that the standard kernel matrix should be cached. It is well known that the standard kernel matrix is symmetric. Using this symmetry around 50% of the needed memory for caching this matrix could be saved. However, accessing any element of the matrix becomes more expensive in this setting. Implementing this technique decreases the performance of LIBSVM by $15\% - 20\%$, while offering to use training sets which consist of around 40% more examples.

Finally, a technique similar as the caching technique described in Section 5.1 is implemented for evaluating pairwise test sets. Note that for testing, it is often sufficient to cache the standard kernel values of those pairs whose first example belongs to the training set and whose second example belongs to the test set.

5.2 Empirical Results

In this section several datasets are introduced and selected results are presented. Several synthetic datasets are used, namely the checker board task in Subsection 5.2.1, the double interval task in Subsection 5.2.2, the (disturbed) orthant task in Subsection 5.2.3, and the disturbed single interval task in Subsection 5.2.4. Note that the checker board task, the double interval task, and the orthant task do not contain noise. In other words, any optimal classifier never makes an error on those datasets. In contrast to this, the disturbed single interval task and the disturbed orthant task do contain noise. In the disturbed single interval task there exist only very few classes and any interclass training does not provide meaningful results. Therefore, only Bayes' DET curves for this dataset are presented and differences between the interclass task and interexample task by means of Bayes'

DET curves are shown. For the disturbed orthant type task it is also possible to calculate Bayes' DET curves. There, the DET curve of pairwise SVMs to Bayes' classifiers are compared. Afterwards, two real world datasets are presented. At first, results for the LFW database are presented in Subsection 5.2.5. This dataset is freely available. At second, results for confidential face datasets by Cognitec Systems GmbH are presented in Subsection 5.2.6. Note that many of the results of Subsections 5.2.1, 5.2.2 and 5.2.5 were already presented in my paper [12].

If not stated otherwise any drawing which is used in this section is uniformly distributed.

5.2.1 Checker Board Task

As mentioned in Section 4.3 an example x of this task has the following form:

$$x \in \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad \text{with} \quad x_1, x_2 \in [0, s)$$

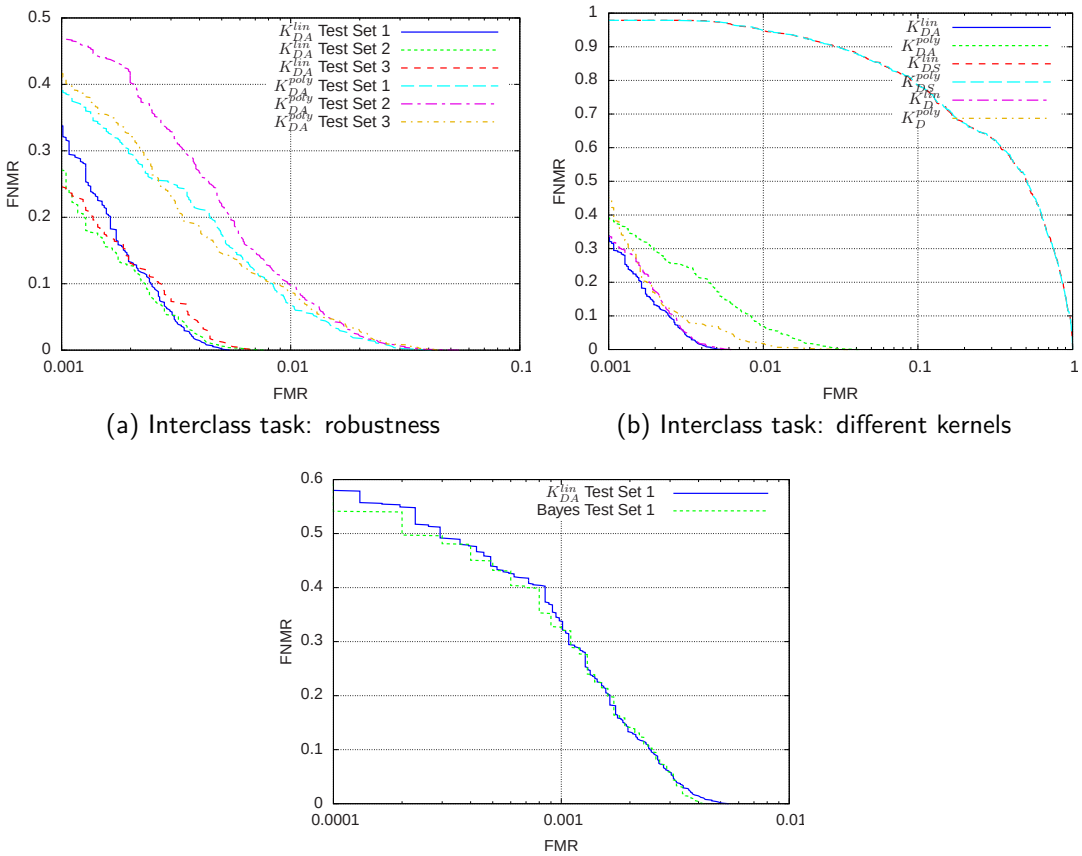
and $s \in \mathbb{N}$ arbitrary but fixed. The class c of an example x is determined by

$$c(x) := \text{floor}(x_1)s + \text{floor}(x_2).$$

In other words each square of the checker board is a single class of the s^2 classes.

For the measurements $s = 25$ is selected and the penalty parameter $C = 1$ is set. For model selection the technique described in Section 3.4 is used. Many different training parameters are tested on this task. However, to keep this thesis short, only selected results are presented. In the pair task a set consisting of 50 classes with a constant EPCR of 5 is created. Using the pair task many kernels are excluded as they led to bad results in this task. As test set for the interexample and interclass task the whole set used in the pair task is used and is called Test Set 1. In the interexample task one trained on newly generated training sets with different EPCRs $(5, 10, 15, 20, 25)$. It was observed that the EPCR does not significantly influence the performance for the checker board task. Hence, an EPCR of 5 was chosen. Furthermore, it was obtained by the interexample task that K_{DA}^{lin} (4.15d) and K_{DA}^{poly} have the best performance. Then, different numbers $(50, 75, \ldots, 200)$ of training classes within the interclass task are tested. Again, the results differed only slightly. Hence, the two models with 50 classes were slected. Figure 5.1a presents the performance of the models with K_{DA}^{lin} and K_{DA}^{poly} on Test Set 1. Additionally, the performance on two newly generated test sets (Test Set 2 and Test Set 3) with

the same properties as Test Set 1 (interclass task, 50 classes, EPCR=5) is tested. This shows the robustness of the model selection technique. In order to complete the discussion of Section 4.3 the performance for other kernels in the interclass task is presented in Figure 5.1b.



(a) Interclass task: robustness

(b) Interclass task: different kernels

(c) Interclass task: Bayes' Classifier

Figure 5.1: DET curves for checker board task. In (b) K_D^{lin} and K_D^{poly} (4.15g) provide almost the same curve.

In the beginning of this section it was stated that one can correctly determine for any pair of this task whether it is positive or not. However, it was shown that the pairwise kernels K_{DA}^{lin} works best. Hence, the projection P_{DA} (4.1) is applied. Section 4.3 discusses that some information is lost if P_{DA} is used. Due to this information loss it is harder to correctly determine whether a given pair is positive or not. Therefore, the pairs of Test Set 1 were projected using P_{DA} and then the Bayes' DET curve was calculated. Figure 5.1c presents this Bayes' DET curve and the DET curve obtained by the pairwise SVM with K_{DA}^{lin} for Test Set 1. Both curves are similar. Note that the same comparison for Test Set 2 and Test Set 3

were done, too. Those comparisons lead to similar results.

5.2.2 Double Interval Task

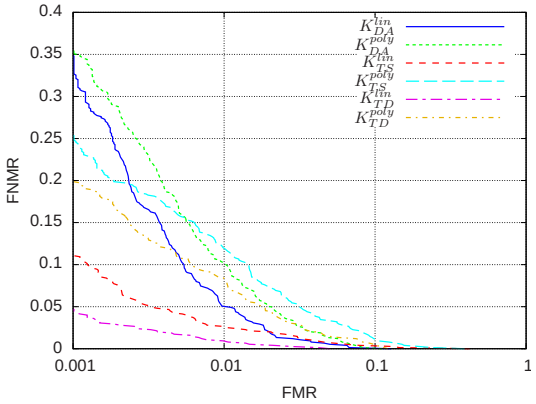
Now, the **double interval task** of dimension n are defined. In order to draw an example $x \in \{-1, 1\}^n$ of the double interval task one draws $i, j, k, l \in \mathbb{N}$ so that $2 \leq i \leq j, j+2 \leq k \leq l \leq n$ and sets

$$x_t := \begin{cases} 1 & t \in \{i, \ldots, j\} \cup \{k, \ldots, l\}, \\ -1 & \text{otherwise.} \end{cases}$$

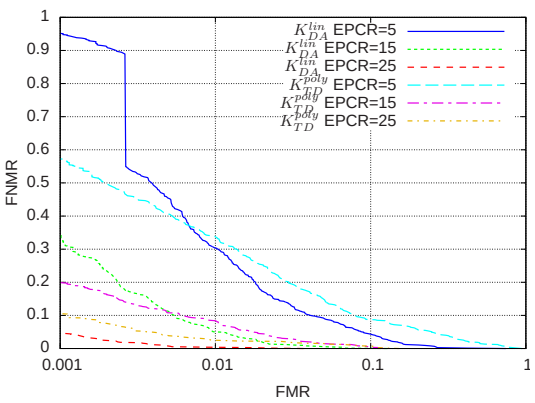
The class c of such an example is defined by $c(x) := (i, k)$. Note that (j and l) do not influence the class. Hence, there are $(n-3)(n-2)/2$ classes. The first component of any double interval type example is enforced to be negative. This was done according to Remark 4.11. Measurements showed that if the first component is not enforced to be negative then the results are not robust.

Obviously, all examples have the same Euclidean norm. For the measurements $n = 500$ is selected. In the pair task an initial set consisting of 750 examples out of 50 classes with a constant EPCR of 15 is created. Then, 75% of all pairs are used for training and the remaining ones are used for testing. By the pair task several parameters were selected. Firstly, it turned out that the penalty parameter C should be set to $1{,}000$ independently of the other parameters. Secondly, the kernels K_{DA}^{lin}, K_{DA}^{poly}, K_{TS}^{lin}, K_{TS}^{poly}, K_{TD}^{lin}, and K_{TD}^{poly} (4.15) were selected due to their superior performance. Figure 5.2a presents the performance of those kernels in the pair task. Afterwards, the whole set of the pair task is used as test set for the interexample and interclass task and is called Test Set 1. In the interexample task, different EPCRs $(5, 10, 15, 20, 25)$ are tested. Figure 5.2b shows that increasing the EPCR leads to better results in the interexample task. This holds for all kernels selected. Due to space limitations only results for K_{DA}^{lin} and K_{TD}^{poly} are presented. Note that as trade-off between performance and needed training time an EPCR of 15 is chosen. Figure 5.2c shows that an increasing number of used classes increases the performance in the interclass task. Again, this holds for all kernels mentioned above but only results for K_{DA}^{lin} and K_{TD}^{poly} are presented. Furthermore, using six different test sets Figure 5.2d shows that the heuristic model selection technique leads to robust results for the interclass task.

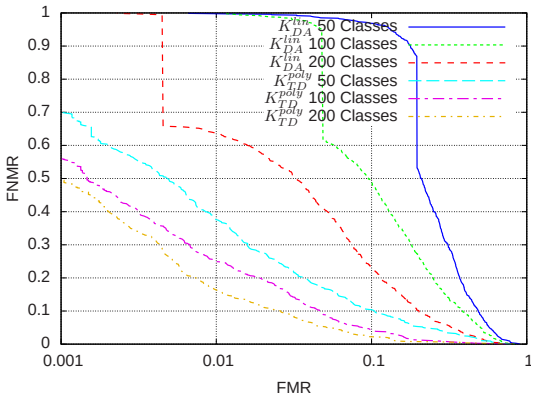
For a fixed kernel it can be seen by means of Figures 5.2a, 5.2e and 5.2f that the DET curve of the pair task is below the DET curve of the interexample task,



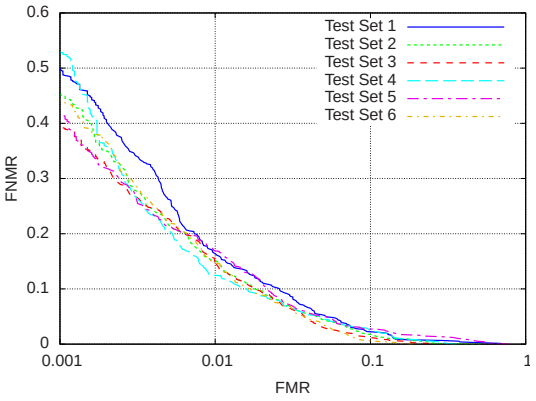
(a) Pair task: different kernels



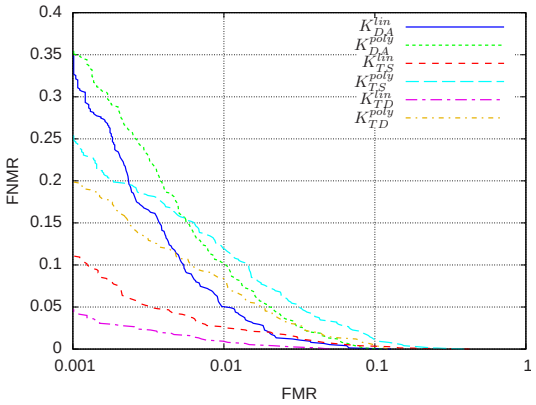
(b) Interexample task: different EPCRs



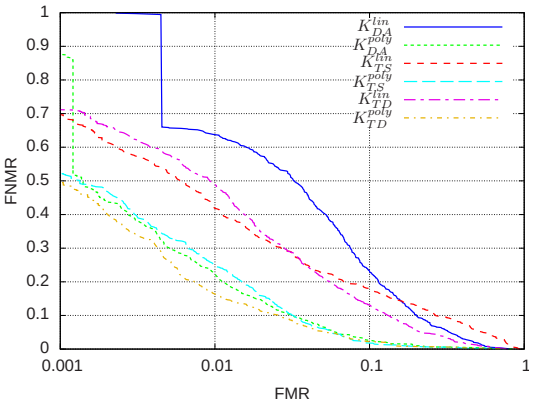
(c) Interclass task: different class numbers



(d) Interclass task: robustness



(e) Interexample task: different kernels



(f) Interclass task: different kernels

Figure 5.2: DET curves for double interval tasks

which again is below the DET curve of the interclass task. In the interexample task (Figure 5.2e) a training set consisting of 50 classes with a constant EPCR of 15 is used, while a training set consisting of 200 classes with a constant EPCR of 15 is used in the interclass task (Figure 5.2f). One obtains that K_{DA}^{lin} is the best kernel in the pair task and the interexample task. However, compared to the other selected kernels it leads to bad results in the interclass task. At the same time the performance of K_{TD}^{poly} in the pair task and interexample task is worse than most of the other used kernels, but K_{TD}^{poly} leads to the best performance in the interclass task.

Above it was shown that an increasing number of classes increases the performance. To increase the performance further training sets with up to $16{,}000$ classes were learned. However, for $n = 500$ there are $124{,}250$ classes. As $16{,}000$ classes represent more than 12.5% of the existing classes for $n = 500$ it was decided to increase n to $2{,}000$. Then, $1{,}997{,}000$ classes exist. Figure 5.3a shows that an increasing number of classes increases the performance of the obtained classifier. Note that only results for K_{TD}^{poly} are presented as this kernel works best for smaller dimensions. In Figure 5.3a a test set consisting of 800 examples with a constant EPCR of 8 is used. In other words there are 100 classes in the test set. The training sets consist of different numbers of classes with a constant EPCR of 8. The size of those training sets prohibits the use of all pairs of the training examples. Hence, all positive pairs are used for training and the number of negative training pairs is set so that the set of training pairs consists of $2.5/5/10/20$ million pairs for $2{,}000/4{,}000/8{,}000/16{,}000$ classes.

Above, pairwise SVMs which did not use all possible training pairs are trained. Figure 5.4a presents DET curves of pairwise SVMs which use $6{,}000$ examples of 750 classes with an constant EPCR of 8 for training. However, different numbers $(2/4/8/18$ mio$)$ of training pairs are used. Note that any positive pair always belongs to the training set. Obviously, the performance increases if more training pairs are used. In Table 5.3a the needed training time and the obtained EER are presented. It can be seen that the training time increases significantly but the EER drops only slightly when more pairs are used for training. Additionally, test sets with different numbers $(250, 500, 750, 1250)$ of classes with an constant EPCR of 8 are created. Then, $2{,}001{,}000$ pairs are used for training, independently of the number of classes. Again, all positive pairs are included into the sets of training pairs. Figure 5.4b shows that increasing the number of classes while fixing the number of used pairs increases the performance. However, Table 5.3b shows that larger sets of training examples increase the needed training time if the number of training pairs is fixed. Tables 5.3a and 5.3b show that a training set with 8 or 18

million pairs of $6{,}000$ examples leads to a comparable performance as a training set with 2 million pairs of $8{,}000$ or $10{,}000$ examples while the needed training time is increase by factor three or six, respectivly.

This subsection is concluded by a discussion whether reflexive test pairs should be used in a test set. In Section 3.1 it was discussed that a pairwise decision function might not be reflexive. Often, a reflexive pair is positive and is very easy to classify. In Figure 5.3b DET curves in the interclass and interexample setting are presented. For training a set consisting of $2{,}000$ classes and K_{TD}^{poly} is used. The interclass test set consists of $2{,}000$ classes with a constant EPCR of 2, while the interexample test set consists of $2{,}000$ classes with a constant EPCR of 8. In Figure 5.3b, DET curves with an without recursive pairs are presented. There, the DET curve of the interclass task with reflexive pairs is below both DET curves of the interexample task. At the same time, the DET curve of the interclass task without the reflexive pairs is above both DET curves of the interexample task. Hence, one might come to wrong conclusions about the quality of a classifier if reflexive test pairs are used.

5.2.3 (Disturbed) Orthant Task

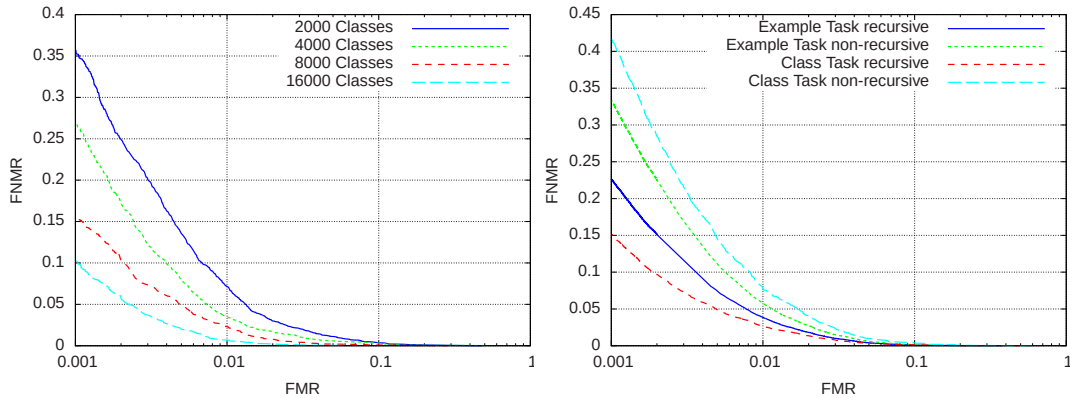
In this subsection two different tasks which are closely connected are presented. Firstly, the **orthant task** is introduced. Again, there exists a pairwise classifier which never makes an error in this task. Secondly, the **disturbed orthant task** is introuduced by modifying the orthant task. If this modification is applied then there is no classifier which never makes an error, but it is possible to obtain pairwise Bayes' Classifiers as all class probabilities for any example x of the disturbed orthant task can be calculated.

This subsection starts with the **orthant task** of dimension n. In order to obtain an example $x \in [-1,1]^n$ of the orthant task one firstly draws $k \in \{1\} \times \{-1,1\}^{n-1}$ and $\bar{x} \in [0,1]^n$. In other words, k is of dimension n and its first component is set to 1. Secondly, one sets

$$x_t := k_t \bar{x}_t, \text{ for all } t \in \{1,\ldots,n\}.$$

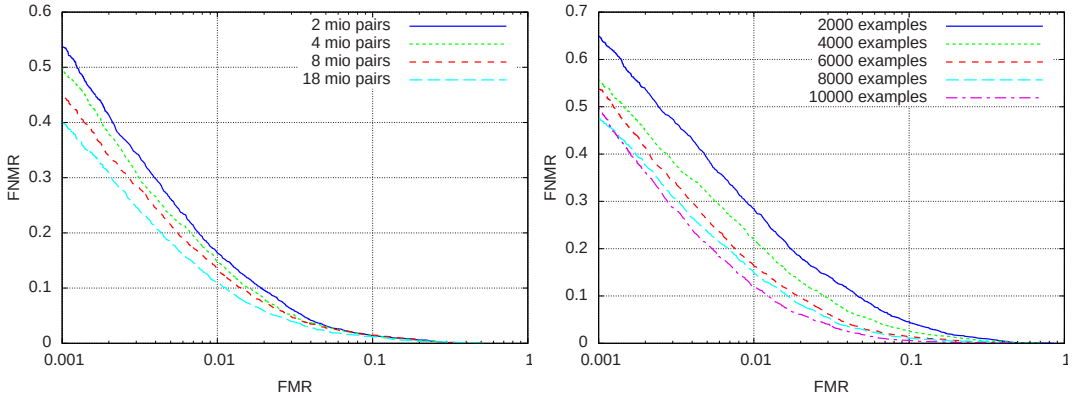
The class c of x is given by $c(x) := k$. Hence, there are 2^{n-1} classes.

Again, the model selection heuristic is used to find sufficiently good parameters. The test sets always consist of 100 classes. The EPCR was set to 8 for the training sets and test sets. The measurements showed that the penalty parameter C should be set to $C = 1{,}000$. In Figure 5.5 selected results for this task are presented.



(a) Class task with many classes (b) Recursive pairs in the test set

Figure 5.3: DET curves for double interval task with examples of dimension 2000



(a) $6{,}000$ examples in the test set (b) 2 million pairs in the test set

Figure 5.4: DET curves for double interval tasks in the interclass setting. A subset of all existing training pairs is used.

# pairs	t in mm:ss	EER in %
2 mio	16:26	4.096
4 mio	31:57	3.810
8 mio	65:03	3.771
18 mio	187:03	3.464

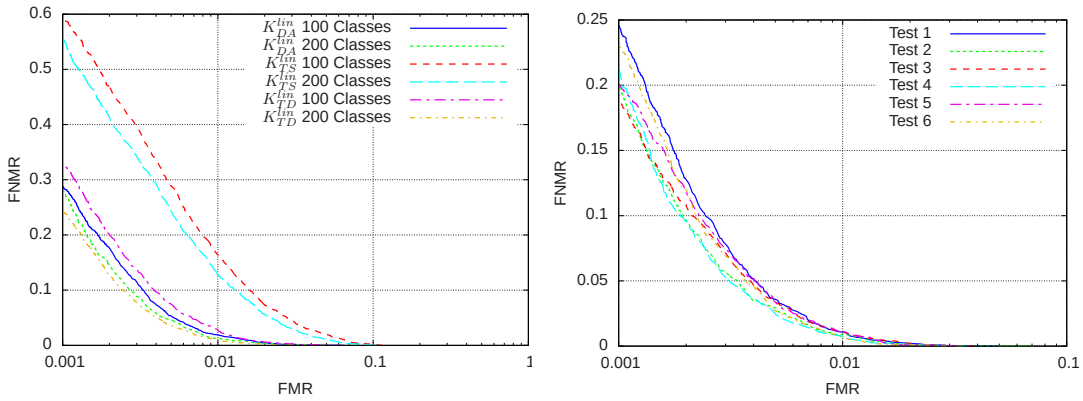
(a) Different numbers of pairs

# examples	t in mm:ss	EER in %
2000	06:30	6.763
4000	12:50	5.355
6000	15:38	4.096
8000	20:13	3.767
10000	24:51	3.303

(b) Different numbers of examples

Table 5.3: Training time for double interval tasks using a subset of all training pairs

Figure 5.5a shows DET curves which lead to the best results in the interclass task. There, the test set is the test set of the interexample task. This test set is called Test Set 1. Additionally, the robustness of the obtained classifiers is analyzed by testing them on newly drawn test sets. All classifiers used in Figure 5.5a lead to robust results. To keep this dissertation short only the performance of the classifier using K_{TD}^{lin} which was trained on 200 classes is presented. In Figure 5.5b results of this classifier on 5 newly drawn test sets are presented.



(a) Interclass task: different kernels

(b) Interclass task: robustness

Figure 5.5: DET curves for orthant tasks

Note that all the presented models lead to very good results with EER of $1\% - 3\%$. It is especially interesting that the K_{TS}^{lin} works well in this database although the data is not normalized. The reason might be that the (Euclidean) norm of the examples does not influence the class in this dataset.

Now, the **disturbed orthant task** of dimension n are introduced. Firstly, let $p_1, p_2 \in [0,1]$ with $p_1 < p_2$ be given. To obtain an example $x \in [-1,1]^n$ of the orthant task one draws $k \in \{1\} \times \{-1,1\}^{n-1}$ and $p \in [0,1]$. Then, if $p < p_1$ one draws $\bar{x} \in [0,1]^n \setminus [\sqrt[n]{0.5}, 1]^n$. Otherwise, if $p \geq p_1$ one draws $\bar{x} \in [0, \sqrt[n]{0.5}]^n$. Now, if $p < p_2$ one sets

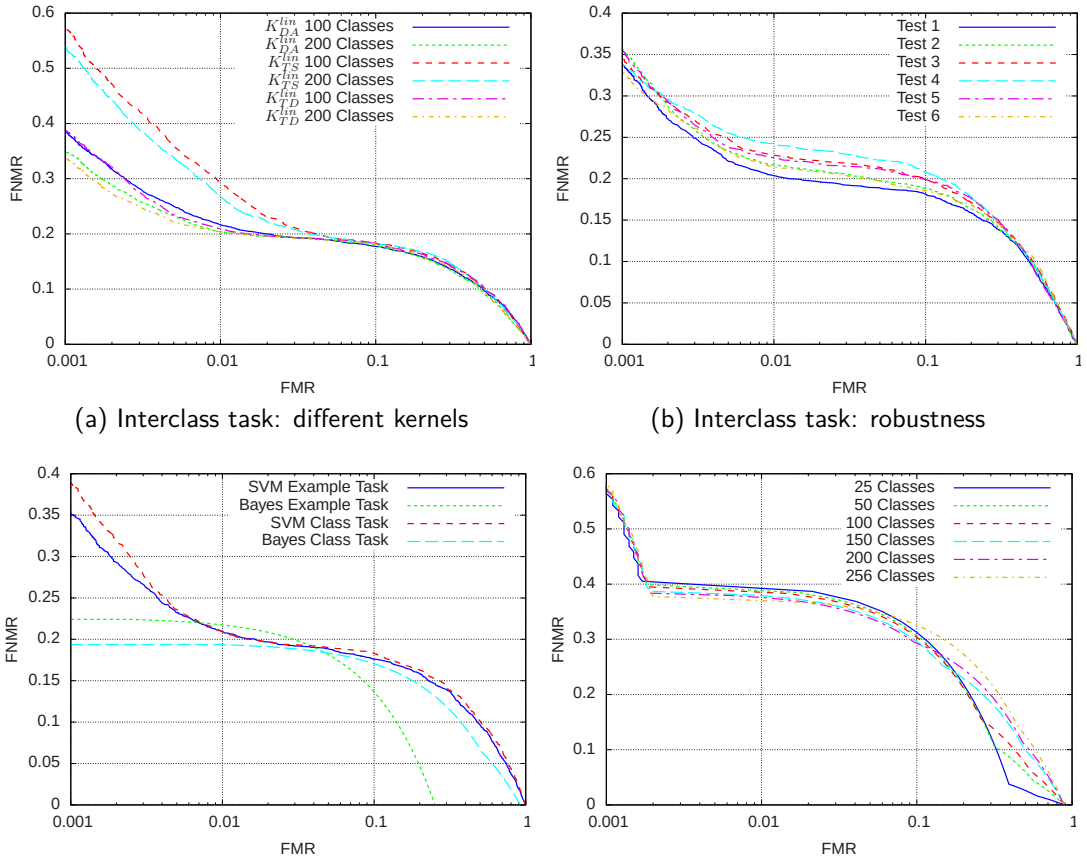
$$x_t := k_t \bar{x}_t, \text{ for all } t \in \{1, \ldots, n\}.$$

If $p \geq p_2$ one draws $\bar{k} \in \{1\} \times \{-1,1\}^{n-1}$ with $k \neq \bar{k}$ and sets

$$x_t := \bar{k}_t \bar{x}_t, \text{ for all } t \in \{1, \ldots, n\}.$$

Again, the class c of x is given by $c(x) := k$ and there are 2^{n-1} classes.

Let an example x of this task be given. If $\max\{|x_t|\} > \sqrt[n]{0.5}$ on can correctly



(a) Interclass task: different kernels

(b) Interclass task: robustness

(c) Interclass task: Bayes Classifiers and SVMs

(d) Interclass task: convergence Bayes Classifiers

Figure 5.6: DET curves for disturbed orthant tasks

determine the class c. Otherwise, this is not possible. However, one can use the signs of the indices of x to determine some $\tilde{k} \in \{-1, 1\}^n$ and can calculate the class probabilities. For any x with $\max\{|x_t|\} \leq \sqrt[n]{0.5}$ and corresponding $\tilde{k}$ the class probabilities are

$$p(C = c|X = x) = \begin{cases} \frac{p_2 - p_1}{1 - p_1} & \text{if } c = \tilde{k} \\ \frac{1 - p_1 + p_2}{(1 - p_1)(2^{n-1} - 1)} & \text{otherwise.} \end{cases}$$

Now, results for the disturbed orthant task are presented. For the measurements $p_1 = 0.5$ and $p_2 = 0.85$ are selected. Again, the model selection heuristic is used to find good parameters. Similar to the orthant type task, an EPCR of 8 is selected for the training sets and test sets and the penalty parameter C is set to $1{,}000$. In the test sets 100 classes are used. The kernels which lead to the best results in the orthant task lead to the best results in the disturbed orthant task, too (see

Figure 5.6a). Figure 5.6b shows that those results are robust. There, the model using K_{TD}^{lin} obtained by training 200 classes is used. Test Set 1 denotes the test set which was already used in the interexample task. In Figure 5.6c Bayes' DET curves of Test Set 1 in the interclass and interexample task are presented. The pairwise Bayes' Classifier of the interexample task only uses the existing 100 classes. Note that the Bayes' DET curve of the interclass task and the Bayes' DET curve of the interexample task intersect. Additionally, in Figure 5.6c the DET curves of Test Set 1 of two SVMs are provided. For both SVMs $C = 1{,}000$ and K_{TD}^{lin} is chosen. Both training sets consist of 100 classes with an EPCR of 8. One training set is created according to the interexample task, while the other one is created according to the interclass task. Interestingly, both DET curves approach the Bayes' DET curve of the interclass task for sufficiently large FMRs.

Additionally, the convergence of Bayes' DET curves in the interexample task for an increasing number of used classes is analyzed. To this end, another test set consisting of 25 classes with an EPCR of 8 of dimension $n = 9$ (256 classes) is created. This test set is always used in Figure 5.6d. There, Bayes' DET curves for different numbers of classes are presented. In other words, any Bayes classifier assumes that the given number of classes exist. However, there are only 25 classes in the test set. Note that the subset relation discussed in Section 3.4 is valid in those measurements. Figure 5.6d demonstrates that the Bayes' DET curves of the interexample task empirically converge to the Bayes' DET curve of the interclass task for an increasing number of classes.

5.2.4 Disturbed Single Interval Task

Now, the **disturbed single interval task** of dimension n is defined. There, one sets $p_0 = 0$ and selects $p_i \in [0,1], i = \{1,\dots,10\}$ with $p_i \leq p_j$ for $i < j$. In order to draw an example $x \in \{-1,1\}^n$ of the double interval task one firstly draws $k \in \{1,\dots,n\}$ and $p \in [0,1]$. Then, one calculates

$$h := \max\{z \in \{0,1\dots,10\} \mid p_z < p\} - 6$$

and sets

$$\tilde{k} := \mathsf{mod}((k+h), n) + 1$$

where mod is the modulo operator. Now, one draws $l \in \{\tilde{k},\dots,n\}$ and sets

$$x_t := \begin{cases} 1 & \text{if } \tilde{k} \leq t \leq l, \\ -1 & \text{otherwise .} \end{cases}$$

Then, the class c of the example x is given by $c(x) := k$. Obviously, there is a close connection between the double interval type task and the disturbed single interval type task. However, there are two differences. Firstly, one positive interval exists in the disturbed single interval task while there are two positive interval in the double interval task. Secondly, any starting index may be modified in the disturbed single interval task while the starting indices are fixed in the double interval task.

In general, it is not possible to uniquely determine the class c of a given example x. However, one can recover $\tilde{k}$ and l by any given example. This enables to calculate all class probabilities. Therefore, it is possible to obtain pairwise Bayes' Classifiers and to calculate Bayes' DET curves.

Figure 5.7 presents different Bayes' DET curve of a test set. In this test set, $n = 500$ and p is selected according to the following table.

i	0	1	2	3	4	5	6	7	8	9	10
p_i	0	.01	.03	.05	.1	.15	.85	.9	.95	.97	.99

The test set consists of 25 classes with a constant EPCR of 8. Interestingly, many Bayes' DET curves intersect. Again, the Bayes classifiers of the interexample task converge empirically to the Bayes classifier of the interclass task. The interclass Bayes classifier is denoted by Bayes500 in Figure 5.7.

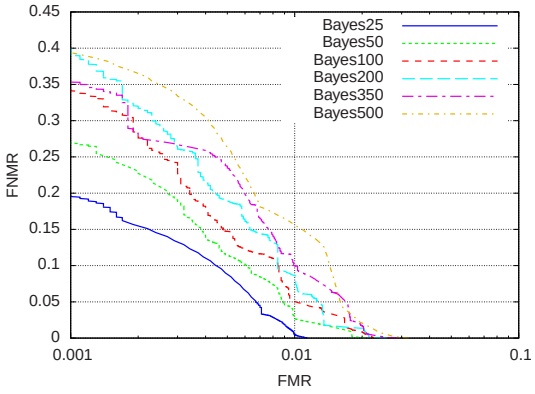


Figure 5.7: Bayes' DET curves for disturbed single interval tasks

5.2.5 LFW Database

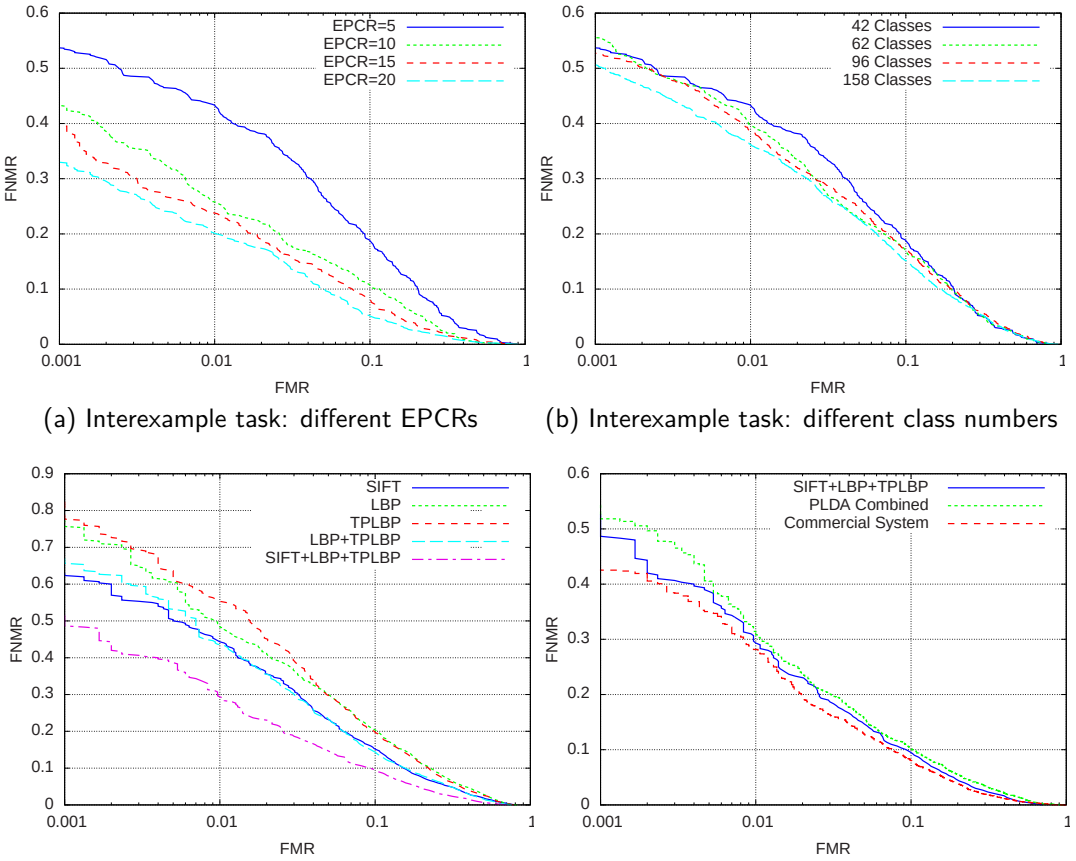
The Labeled Faces in the Wild (LFW) dataset [25] consists of $13{,}233$ images of $5{,}749$ persons. Several remarks on this dataset have to be described. Firstly, the

dataset is very inhomogeneous. There are only $1{,}680$ persons with two or more images. Moreover, there are persons with up to 530 images. Secondly, in [25] there are two standard test procedures suggested for this dataset. Here, the unrestricted test procedure is used. This test procedure is a fixed tenfold cross validation in the interclass setting, where each test set consists of 300 positive pairs and 300 negative pairs. Thirdly, there are several feature vectors available for the LFW dataset. The presented measurements mainly follow article [35] and use the scale-invariant feature transform (SIFT)-based feature vectors for the funneled version of LFW which are described in [21]. In addition, the aligned images presented in [47] are used as well. Again, following [35], the aligned images are cropped to 80×150 pixels and are then normalized by passing them through a log function ($\log(x+1)$). Afterwards, the local binary patterns (LBP) (see [30]) and three-patch LBP (TPLBP) (see [46]) are extracted. In contrast to [35] the pose is neither estimated nor swapped. Furthermore, no PCA is applied to the data. As the norm of the LBP feature vectors is not the same for all images those features are scaled to unit norm.

For models selection, the View 1 partition of the LFW database is recommended [25]. In this partition K_{TD}^{poly} works best independently of the chosen type of feature vector. Additionally, the model selection technique was applied to the LFW database. Due to the inhomogeneity of the dataset the model selection technique is only used to select the pairwise kernel. By the pair task and the interexample task the same results as by the View 1 partition are obtained. It seems that K_{TD}^{poly} will work best in this dataset independently of the chosen feature vector.

In addition, using the idea of the model selection technique in Section 3.4 an interesting analysis about the EPCR by means of the SIFT-based feature vectors is presented. In Figure 5.8a 42 classes are used. There, it is shown that the performance in the interexample task increases with an increasing EPCR. In particular, a constant EPCR of 5 seems to be too small. Hence, this dataset seems to suffers from a small EPCR (2.3 in average). Fortunately, using an EPCR of 5 and increasing the numbers of classes in the training set increases the performance in the interexample setting, too (see Figure 5.8b).

Now, the interclass task is analyzed by the tenfold cross validation mentioned above. The speed up technique presented in Section 5.1 enabled to train with large numbers of training pairs. However, if all pairs are used for training, then any training set would consist of approximately $50{,}000{,}000$ pairs and the training would still need too much time. Hence, whereas in any training set all positive training pairs are used, the negative training pairs are randomly selected in such a way that any training set consist of $2{,}000{,}000$ pairs. The training of all ten models took less than 24



(a) Interexample task: different EPCRs

(b) Interexample task: different class numbers

(c) Interclass task: different feature vectors

(d) Interclass task: comparison to other classifiers

Figure 5.8: DET curves for the LFW dataset

hours in this case. Figure 5.8c presents the average DET curves for feature vectors based on SIFT, LBP, and TPLBP. Inspired by [35] the decision function values of these pairwise SVMs are added and two further DET curves are obtained. This led to very good results (see Figure 5.8c). Furthermore, the SIFT, LBP, and TPLBP feature vectors were concatenated. Surprisingly, the training of some of those models needed longer than a week. Therefore, these results are not presented.

In Table 5.4 the mean equal error rate (EER) and standard error of the mean (SEM) for several types of feature vectors are provided. Note that many of the presented results are state of the art or even better. The current state of the art can be found on the homepage of [25] and in the publication of [35]. If only SIFT-based feature vectors are used, then the best result is 0.125 ± 0.0040 (EER$\pm$SEM). Pairwise SVMs achieve the same EER but a higher SEM 0.1252 ± 0.0062. If the decision function values corresponding to the LBP and TPLBP feature vectors are added, then the

result 0.1210 ± 0.0046 is slightly worse compared to 0.1050 ± 0.0051. One possible reason for this fact might be that the pose was not swapped. Finally, for the added decision function values, the performance 0.0947 ± 0.0057 is significantly better than 0.0993 ± 0.0051. Furthermore, it is worth noting that the SEM of pairwise SVMs are comparable to the other presented learning algorithms although most of them use a PCA to reduce noise and dimension of the feature vectors. Only for the SIFT based feature vectors the SEM of pairwise SVMs is larger. Note that the commercial system uses outside training data. In other words, the commercial system uses another test protocol which includes data not part of the LFW database.

		SIFT	LBP	TPLBP	L+T	S+L+T	CS
Pairwise	Mean	0.1252	0.1497	0.1452	0.1210	0.0947	-
SVM	SEM	0.0062	0.0052	0.0060	0.0046	0.0057	-
State of	Mean	0.1250	0.1267	0.1630	0.1050	0.0993	0.0870
the Art	SEM	0.0040	0.0055	0.0070	0.0051	0.0051	0.0030

Table 5.4: EER and SEM for LFW database. Abbreviations: S=SIFT, L=LBP, T=TPLBP, +=adding the decision function values, CS=Commercial system `face.com r2011b`

Additionally, in Figure 5.8d the DET curve of the added decision function values of SIFT, LBP and TPLBP is presented. Moreover, the DET curve of the probabilistic linear discriminant analysis (PLDA Combined), the current state of the art, and the DET curve of a commercial system (CS) are presented. The DET of the commercial system is below the DET curve of the added decision function values which again is below the DET curve of PLDA Combined.

5.2.6 Cognitec Databases

In this section results on other face datasets are presented. Those datasets and the used feature vectors were provided by Cognitec Systems GmbH. However, I am only allowed to present the performance of pairwise SVMs using those feature vectors but not to provide those feature vectors or databases. Each feature vectors has dimension 1952 and is normalized. In total, there are three different datasets: the **Cognitec-Train** dataset, the **Cognitec-Test-B** dataset and the **Cognitec-Test-A** dataset. The first dataset is used for training, the other two datasets are used for testing. Please note that the two test sets are created according to the interclass setting with respect to the training set.

Now, those datasets are described. The Cognitec-Train dataset consists of $122{,}052$ images of $20{,}981$ persons. This dataset is very inhomogeneous. There are $6{,}304$ persons which are represented by 1 image, while there are persons represented by up to 191 images. See Table 5.5 for an overview of the EPCRs. In this dataset, there are $3{,}077{,}129$ positive pairs. However, $1{,}000{,}000$ of those pairs belong to 177 persons and $2{,}000{,}000$ of those pairs belong to 389 persons.

EPCR	Classes	EPCR	Classes
1	6304	7	952
2	7138	8	180
3	1375	9	146
4	629	10	71
5	378	11-100	1306
6	2321	101-191	181

Table 5.5: EPCR and number of classes for CognitecTrain

The Cognitec-Test-A dataset consists of two parts, namely a probe and a gallery. The probe consists of $3{,}174$ images of $1{,}347$ persons while the gallery consists of $9{,}302$ images of $3{,}103$ persons. Note that there are 13 persons in the probe which are not represented in the gallery. For this dataset another test setting is used. In this setting one firstly creates all pairs whose first image belongs to the probe and whose second image belongs to the gallery. Then, one has to determine whether those pairs are positive or not. In this setting, there are $19{,}984$ positive pairs and around 29.5 million negative pairs. The DET curve is represented in Figure 5.9.

The Cognitec-Test-B dataset consists of $2{,}000$ passport-style images of $1{,}005$ persons. Here, 995 persons are presented by 2 images and 10 persons are presented by 1 image. Note that reflexive test pairs are not used in Figure 5.9.

The Cognitec-Train and the Cognitec-Test-A dataset come from many different sources. For instance, some pictures have been taken in a labor or are passport-like, while other pictures have been taken in real life with strong expressions, occlusions, or different poses.

For training all existing classes of Cognitec-Train are used. At the same time, each person is represented by at most 8 images. Hence, $64{,}773$ examples were used for training. This yields to a training set consisting of $191{,}149$ positive training pairs where $126{,}376$ pairs are not reflexive. Additionally, around 23 million negative pairs are used.

It was considered to apply and present other pairwise classification techniques to those training sets but several problems occurred. Firstly, source codes are sometimes not available. Secondly, many methods are only able to handle up to $10{,}000$ pairs within an acceptable training time. In other words, only $20{,}000$ examples could be used. Thirdly, many different models for pairwise SVMs and the presented datasets were calculated. Due to time restrictions it would be impossible to test as many models for any technique as there were tested for pairwise SVMs. Thus, any comparison would not be fair. Therefore, only the performance of the angle classifier is presented. This classifier uses the angle between each pair of examples. For Cognitec-Test-B the angle classifier leads to bad but acceptable results with an EER of 18.1%. Surprisingly, the angle classifier leads to very bad results for Cognitec-Test-A with an EER of 46.5%. In other words this classifier is not much better than randomly assigning any pair to be positive or negative.

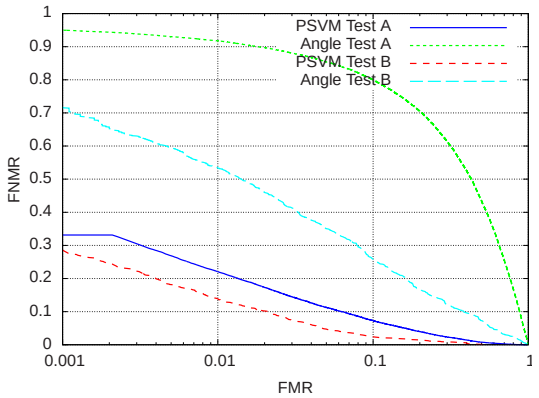


Figure 5.9: DET curves for Cognitec datasets

List of Symbols

C	> 0 Penalty Parameter
g	decision function
$\mathcal{H}$	a real Hilbert space
$k(\cdot, \cdot)$	standard kernel
$K((\cdot, \cdot), (\cdot, \cdot))$	pairwise Kernel
M	$\{1, 2, \ldots, m\}$
m	number of training points
$\mathbb{N}$	$\{1, 2, \ldots\}$
l	loss function
n	dimension of the training points
P	probability measure
$\mathbb{R}_+$	$\{x \in \mathbb{R} \mid x \geq 0\}$
X	Input Space
Y	Output Space
Z	A Stochastic Process
$\kappa((\cdot, \cdot), (\cdot, \cdot))$	Outer Kernel
$\psi(\cdot)$	$\psi : X \to \mathcal{H}$
$(\cdot)_+$	$\max\{0, \cdot\}$
∇	gradient
∇_x	gradient with respect to x
$\|\cdot\|$	norm induced by the scalar product
$\oplus$	Direct Sum
$\otimes$	Tensor Product
$\times$	Cartesian product
$\circ$	composition of two functions

List of Abbreviations

DET	Detection Error Trade-off
EER	Equal Error Rate
EPCR	Example Per Class Ratio
ERM	Empirical Risk Minimization
FMR	False Match Rate
FNMR	False Non Match Rate
i.i.d.	Independent and Identical Distributed
KKT	Karush-Kuhn-Tucker
LFW	Labeled Faces in the Wild
OCSVM	One Class Support Vector Machine
PLDA	Probabilistic Linear Discriminant Analysis
RBF	Radial Basis Function
SEM	Standard Error of the Mean
SRM	Structural Risk Minimization
s.t.	subject to
SV	Support Vector
SVM	Support Vector Machine
VC	Vapnik-Chervonenkis
w.l.o.g.	without loss of generalization

Index

Bibliography

[1] J. Abernethy, F. Bach, T. Evgeniou, and J.-P. Vert. A New Approach to Collaborative Filtering: Operator Estimation with Spectral Regularization. *Journal of Machine Learning Research*, 10:803–826, 3 2009.

[2] A. Bar-Hillel, T. Hertz, and D. Weinshall. Boosting margin based distance functions for clustering. In *In Proceedings of the Twenty-First International Conference on Machine Learning*, pages 393–400, 2004.

[3] A. Bar-Hillel, T. Hertz, and D. Weinshall. Learning distance functions for image retrieval. volume 2, pages II–570–II–577 Vol.2, June 2004.

[4] A. Bar-Hillel and D. Weinshall. Learning a kernel function for classification with small training samples. In *International Conference on Machine Learning (ICML*, pages 401–408, 2006.

[5] A. Bar-Hillel and D. Weinshall. Learning distance function by coding similarity. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 65–72, New York, NY, USA, 2007. ACM.

[6] A. Bar-Hillel and D. Weinshall. Learning distance function by coding similarity. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 65–72, New York, NY, USA, 2007. ACM.

[7] A. Ben-Hur and W. S. Noble. Kernel methods for predicting protein–protein interactions. *Bioinformatics*, 21(suppl 1):38–46, 1 2005.

[8] O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

[9] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004.

[10] R. C. Bradley. Basic properties of strong mixing conditions. a survery and some open questions. *Probability Surveys*, 2:107–144, 2005.

[11] C. Brunner, A. Fischer, K. Luig, and T. Thies. Association problems and association support vector machines. Technical Report MATH-NM-05-2010, Institute for Numerical Mathematics, TU Dresden, 08 2010.

[12] C. Brunner, A. Fischer, K. Luig, and T. Thies. Pairwise kernels, support vector machines, and the application to large scale problem. Technical Report MATH-NM-04-2011, Institute for Numerical Mathematics, TU Dresden, 08 2011.

[13] C. C. Chang and C. J. Lin. *LIBSVM: a library for support vector machines; Revised 2007*, 2001.

[14] N. V. Chawla, N. Japkowicz, and A. Kotcz. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explor. Newsl.*, 6:1–6, June 2004.

[15] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, Cambridge, UK, March 2000.

[16] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Mach. Learn.*, 29:103–130, November 1997.

[17] R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918, December 2005.

[18] M. Gamassi, M. Lazzaroni, M. Misino, V. Piuri, D. Sana, and F. Scotti. *Accuracy and performance of biometric systems*, pages 510 – 515. Institute of electrical and electronics engineers, Piscataway, 2004.

[19] T. Gao and D. Koller. Multiclass boosting with hinge loss based on output coding. In *Proceedings of International Conference on Machine Learning (ICML)*, 2011.

[20] C. Geiger and C. Kanzow. *Theorie und Numerik restringierter Optimierungsaufgaben*. Springer, 2002.

[21] M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? metric learning approaches for face identification. In *International Conference on Computer Vision*, pages 498–505, Sep 2009.

[22] R. Herbrich. *Learning Kernel Classifiers: Theory and Algorithms*. MIT Press, Cambridge, MA, USA, 2001.

[23] T. Hertz, A. Bar-Hillel, and D. Weinshall. Boosting margin based distance functions for clustering. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pages 50–, New York, NY, USA, 2004. ACM.

[24] T. Hertz, A. Bar-Hillel, and D. Weinshall. Learning distance functions for image retrieval. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 570–577, 2004.

[25] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[26] N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6:429–449, October 2002.

[27] T. Joachims. Making large-scale svm learning practical. LS8-Report 24, Universität Dortmund, LS VIII-Report, 1998.

[28] Y. Lin. Support vector machines and the bayes rule in classification. *Data Mining and Knowledge Discovery*, 6:259–275, 2002.

[29] M. Mohri and A. Rostamizadeh. Stability bounds for stationary ϕ-mixing and β-mixing processes. *J. Mach. Learn. Res.*, 11:789–814, March 2010.

[30] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:971–987, July 2002.

[31] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. pages 130–136, 1997.

[32] P. J. Phillips. Support vector machines applied to face recognition. In *Advances in Neural Information Processing Systems 11*, pages 803–809. MIT Press, 1999.

[33] J. C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, April 1998.

[34] J. C. Platt. *Fast training of support vector machines using sequential minimal optimization*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

[35] S. Prince, P. Li, Y. Fu, U. Mohammed, and J. Elder. Probabilistic models for inference about identity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(PrePrints), 2011.

[36] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *J. Mach. Learn. Res.*, 5:101–141, December 2004.

[37] B. Schölkopf, J. Platt, and A. Smola. Kernel method for percentile feature extraction. TR MSR 2000-22, Microsoft Research, Redmond, WA, 2000.

[38] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[39] I. Steinwart, D. Hush, and C. Scovel. Learning from dependent observations. *Journal of Multivariate Analysis*, 100(1):175 – 194, 2009.

[40] D. Tasche. *Oszillationsmaße und stark mischende zufällige Folgen*. PhD thesis, TU Berlin, 1996.

[41] D. M. J. Tax and R. P. W. Duin. Support vector data description. *Machine Learning*, 54:45–66, January 2004.

[42] V. N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, September 1998.

[43] J.-P. Vert, J. Qiu, and W. Noble. A new pairwise kernel for biological network inference with support vector machines. *BMC Bioinformatics*, 8:1–10, 2007.

[44] M. Vidyasagar. *A Theory of Learning and Generalization*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2nd edition, 2002.

[45] L. Wei, Y. Yang, R. M. Nishikawa, and M. N. Wernick. Learning of perceptual similarity from expert readers for mammogram retrieval. In *ISBI*, pages 1356–1359, 2006.

[46] L. Wolf, T. Hassner, and Y. Taigman. Descriptor based methods in the wild. In *Real-Life Images workshop at the European Conference on Computer Vision (ECCV)*, October 2008.

[47] L. Wolf, T. Hassner, and Y. Taigman. Similarity scores based on background samples. In *ACCV (2)*, pages 88–97, 2009.

[48] B. Yu. Rates of convergence for empirical processes of stationary mixing sequences. Ann. Prob. 22, 94-116 (1994)., 1994.

[49] B. Zou, L. Li, and Z. Xu. The generalization performance of erm algorithm with strongly mixing observations. *Machine Learning*, 75:275–295, June 2009.

Affirmation

Hereby I affirm that I wrote the present thesis without any inadmissible help by a third party and without using any other means than indicated. Thoughts that were taken directly or indirectly from other sources are indicated as such. This thesis has not been presented to any other examination board in this or a similar form, neither in Germany nor in any other country.

I have written this dissertation at Dresden University of Technology under the scientific supervision of Prof. Dr. rer. nat. habil. Andreas Fischer.

There have been no prior attempts to obtain a PhD at any university.

I accept the requirements for obtaining a PhD (Promotionsordnung) of the Faculty of Science of the TU Dresden, issued February 23th, 2011.

Versicherung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt.

Die vorliegende Dissertation habe ich an der Technischen Universität Dresden unter der wissenschaftlichen Betreuung von Prof. Dr. rer. nat. habil. Andreas Fischer angefertigt.

Es wurden zuvor keine Promotionsvorhaben unternommen.

Ich erkenne die Promotionsordnung der Fakultät Mathematik und Naturwissenschaften der TU Dresden vom 23. Februar 2011 an.

Place, Date

Signature