universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

Optimization Aspects of Support Vector Machines

verfasst von / submitted by

Katharina Eibensteiner, BSc.

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc.)

Wien, 2017 / Vienna 2017

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 920
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Quantitative Economics, Management and Finance
Betreut von / Supervisor	Univ.-Prof. Dr. Immanuel Bomze

Acknowledgements

I would first like to acknowledge my thesis advisor at University of Vienna, Univ. Prof. Dr. Immanuel Bomze, for the great support during the time of writing this thesis. Whenever I had a question he was ready to give me advice while always letting this thesis being my own work. It was always a pleasure to discuss the contents but also mathematical issues with him as he was always open for new things and respected my ideas for this work.

My major thank goes to my parents Elisabeth and Gerold who encouraged me in all circumstances to be myself and to do what I am interested in. Supporting me financially but also mentally was always a matter of course, making my time as a student to a wonderful period in my life.

Last, I would like to thank my colleagues and friends, especially Felix who always supported me throughout my time as a student.

Curriculum Vitae

Persönliche Daten

Name:	Katharina Eibensteiner
Geboren am:	18.04.1992
Adresse:	Nussbaumallee 18/3/321, 1110 Wien
E-Mail:	kathi.eibensteiner@gmail.com

Universitäre Ausbildung

Seit 03/2015:	Universität Wien; Masterstudium Quantitative Economics, Management and Finance Vertiefung: Finance
Seit 03/2015:	Universität Wien; Masterstudium Mathematik Vertiefung: Biomathematik
03/2011-02/2015:	Universität Wien; Bachelorstudium Mathematik Vertiefung: Berufsvorbereitende Mathematik Bachelorarbeit 1: *Das Chromatische Polynom* Bachelorarbiet 2: *Der Minkowskische Gitterpunktsatz*

Berufliche Laufbahn

Seit 03/2011:	Aushilfe Rechnungswesen; WUK Werkstätten- und Kulturhaus
08/2012:	Praktikum; Österreichische Kontrollbank

Abstract

Support Vector Machines have become a powerful tool in the field of machine learning and pattern recognition within the last few decades. After Vladimir Vapnik introduced this method in the late 1970s, much research effort has gone into the development of this classification mechanism. This thesis aims to give a summary over the basic principles of classification, to develop approaches for binary Support Vector Machines and to introduce various multicategory classification algorithms, including the extension of multiple binary Support Vector Machines, for developing algorithms that divide multiclass problems into smaller binary problems. Also approaches where the whole classification is covered by one Support Vector Machine will be discussed. The implementation of a handwritten digit recognition algorithm using the mathematical software Matlab and the MNIST-database aims to illustrate the advantages and disadvantages of the various multicategory Support Vector Machines.

Zusammenfassung

Im Bereich des maschinellen Lernens und der Mustererkennung sind Support Vector Machines in den letzten Jahrzehnten zu einem mächtigen Werkzeug geworden. Nachdem diese Methode Ende der 1970er Jahre von Vladimir Vapnik entwickelt wurde, ist viel Forschungsarbeit in die Verbesserung dieses Klassifizierungsmechanismus gesteckt worden. Diese Masterarbeit behandelt grundlegende Prinzipien der Klassifizierung, die Entwicklung binärer Support Vector Machines und verschiedener Mehrklassen-Klassifizierungsalgorithmen, unter denen sowohl jene sind, die das Mehrklassenproblem in mehrere kleine binäre Probleme zerteilen, als auch welche die das Klassifizierungsproblem global lösen. Die Implementierung eines Algorithmus zur handschriftlichen Zahlenerkennung mit der mathematischen Software Matlab und der MNIST- Datenbank zielt darauf ab, die Vor- und Nachteile der verschiedenen Mehrklassen-Methoden aufzuzeigen.

Contents

1 Introduction

In the last few decades society underwent a huge technological change, making computers and their manifold of applications more and more important both in private but also in scientific and economic domains. Therefore it is not surprising that a huge effort has gone into the research field of machine learning which aims to build algorithms that discover structure in data [**?**].

Starting in the middle of the 20th century, much effort has gone into the development of machine learning, making it possible to train a machine in such a way that, upon given certain patterns whose nature is known, it is building rules for classification, enabling it to classify unseen patterns. This ability is used in many fields of application as image processing, medical practice, computer vision, pattern recognition, applied statistics and artificial intelligence [**?**]. A method that has particularly done well in this context is the so called Support Vector Machine (SVM) which is the central point of interest in this thesis. Early developments of the SVM have already been made in the 1930s where R.A. Fisher suggested the first algorithm for pattern recognition [**?**]. With the beginning of Statistical Learning Theory, which was developed by Vapnik in the 1970s, SVMs have been investigated and improved to the extent that they have become a very popular tool for machine learning and especially for pattern recognition.

In a nutshell, a Support Vector Machine aims to separate given data, where each pattern is assigned to a certain class, with a maximal separation space in between the different classes. Given a new pattern, it should be able to assign the correct class to it, based on the knowledge it was given due to the so called training data.

Starting with the formal idea of classification, we will develop algorithms that are able to train rules of classification, first for two groups, also called binary classification problems, and later also for multiclass classification. Within this work we will discuss the instance of hard margin SVMs, where no training error is allowed, therefore the classes have to be separable. As this is not often the case for real world data, we will also develop techniques to overcome this restriction, the so called soft margin SVM and also the kernel trick. For extending the binary classification problem to more than two classes, we will discuss various approaches, some that use multiple binary SVMs, decomposing the multiclass problem into smaller binary problems, and some that can cover the whole problem at once. Finally we will test some of these algorithms in order to see how we can implement SVMs using the mathematical software Matlab. To this extent we will perform a pattern recognition on handwritten digits of the MNIST-database in order to see how various settings in the training and testing stage of SVMs influence the error ratio and therefore the generalization ability of SVMs.

2 Basic Principles

Many books and papers, e.g. [**?**],[**?**], are discussing principles of Machine Learning and within this field of research the popular and often investigated Support Vector Machine. Essentially they all want to solve a classification problem that is stated in the following way.

2.1 Idea of Binary Classification

Suppose we are given n objects $x_1,\dots,x_n$ in a set $\mathscr{X}$ which are labelled either by $y_i=-1$ or $y_i=+1$ where $i=1,\dots n$. Our aim is to assign a label $y_\star$ to a new object $x_\star$ whose labelling is unknown.

Example. A classification problem could be stated the following way: Suppose we have 10 green marbles and 10 blue marbles. In this framework the colour is the label y_i and the marbles are the objects x_i, $i=1,\dots,20$, in the set $\mathscr{X}$. Now we get a new marble, i.e. $x_\star$, and we have to decide if it is either green or blue, i.e. if $y_\star=+1$ or $y_\star=-1$.

Formally the **binary classification problem** looks as follows: We are given n pairs of objects

$$(x_1,y_1),\dots,(x_n,y_n)\in\mathscr{X}\times\{\pm1\} \tag{2.1}$$

and we have to decide if $(x_\star,y_\star)$ is either in $\mathscr{X}\times\{+1\}$ or in $\mathscr{X}\times\{-1\}$. In this framework we will call x_i the **training patterns** or **training points**, $\mathscr{X}$ the **training set** and y_i the **lables** or **classes**. The patterns we want to classify are called **test patterns**.

In order to solve such kind of problems we first need a concept that is able to assess similarities between objects.

For our purpose we are looking for a method that compares two objects and assigns a real number according to their similarity as shown in (**??**) [**?**]. This function k is also known as **kernel function** .

$$\begin{aligned} k:\mathscr{X}\times\mathscr{X}&\to\mathbb{R}\\ (x,x')&\mapsto k(x,x') \end{aligned} \tag{2.2}$$

Remark. *Unless stated otherwise we will assume k to be symmetric, i.e.* $k(x,x')=k(x',x)$.

The choice of this rather generally defined similarity measure introduced in (**??**) turns out to be one of the main challenges within the topic of pattern recognition, therefore we start with a

simple kernel function, the dot product.

2.2 A Simple Similarity Measure: The Dot Product

Looking for a suitable similarity measure we will first discuss the **linear kernel** which equates to the dot product as function k in (**??**).

Definition 2.1. *Let $x = (x_1, \ldots, x_n)$ and $x' = (x'_1, \ldots, x'_n)$ be two vectors in $\mathbb{R}^n$. The **canonical dot product** is defined as*

$$\langle x, x' \rangle := x_1 x'_1 + \ldots x_n x'_n = \sum_{i=1}^{n} x_i x'_i. \tag{2.3}$$

*In the following we will also refer to it as inner product or scalar product [**?**],[**?**].*

Using this function as a similarity measure has several advantages because of its geometrical interpretation:

- it computes the cosine of the angle between the vectors x and x' if they are normalized to length 1,
- it allows to compute the length of a vector and
- it allows to compute the distance between two vectors.

Therefore we can easily compare patterns in term of angles, lengths and distances [**?**].

Before we can work with the dot product as similarity measure, we have to make sure that the objects $\{x_1, \ldots, x_n\} \subseteq \mathcal{X}$ actually exist in a dot product space. For this purpose we define a function Φ which maps any pattern from the **input space** $\mathcal{X}$ into a so called **feature space** $\mathcal{H}$,

$$\begin{aligned} &\Phi : \mathcal{X} \to \mathcal{H} \\ &x \mapsto \mathbf{x} := \Phi(x), \end{aligned} \tag{2.4}$$

containing the vectorial representation in the feature space $\mathcal{H}$ of the patterns in $\mathcal{X}$ [**?**]. We do not make any restrictions on $\mathcal{H}$ except that it has to be equipped with a dot product. Therefore it does not necessarily have to coincide with the $\mathbb{R}^n$ [**?**].

After we have mapped the data into the feature space $\mathscr{H}$ which is equipped with a dot product, allowing us to measure similarities between objects, we can start studying learning algorithms using linear algebra and analytic geometry [**?**].

2.3 Toy Example

For a better understanding we will now look at an example[1] where it comes clear why the dot product and the representation of objects in a dot product space is useful in terms of pattern recognition.

Example. We are given 7 objects $x_1, \ldots, x_7$ which are already embedded into the dot product space $\mathbb{R}^2$. Each of these objects is assigned to one of the classes $+1$ or -1. Considering an unseen point we want to assign it to the class with closer mean.

In order to make things easy, we will deal with concrete points in $\mathbb{R}^2$: $x_1 = (2,1)$, $x_2 = (3,2)$, $x_3 = (4,4)$ and $x_4 = (1,5)$ are assigned to class $+1$ and $x_5 = (6,3)$, $x_6 = (7,1)$ and $x_7 = (9,4)$ are members of class -1. We are now faced with the point $x_\star = (6,0.5)$ which has to be marked either by $+1$ or -1 (Figure **??**).

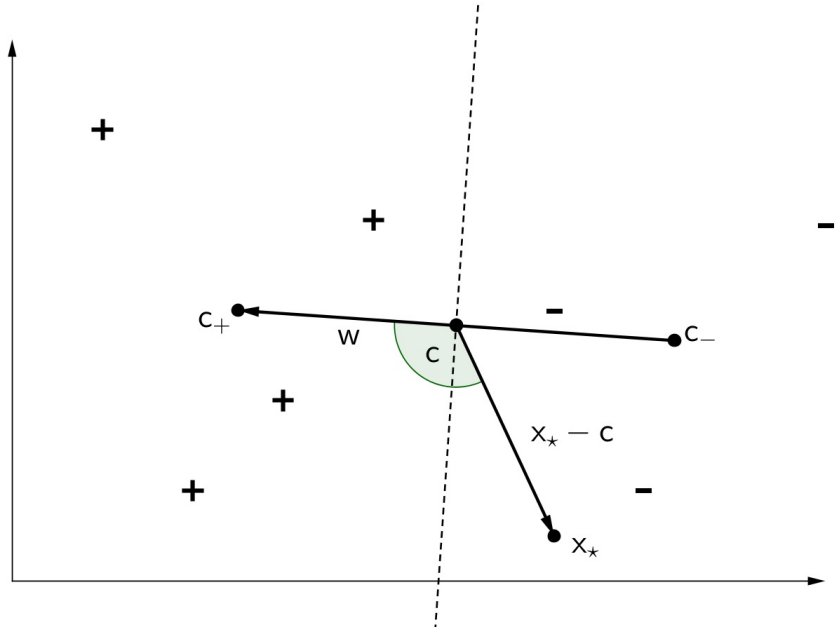


Figure 2.1: Toy example of classification in $\mathbb{R}^2$

[1] The idea for this example is taken from [**?**] extended with numbers and figures.

As stated above, the basic idea of this algorithm is to assign this point to the class with the closer mean. Intuitively this approach would tell us to assign $x_\star$ to the class -1, but in order to see where and how the dot product is used we want to compute the label $y_\star$ explicitly.

Thus, we first have to compute the means of the two classes c_+ and c_- by adding up all points in one class and dividing the sum by the number of points in that class:

$$\begin{aligned} c_+ &= \frac{1}{n_+} \sum_{\{i|y_i=+1\}} x_i = (\frac{5}{2}, 3) \\ c_- &= \frac{1}{n_-} \sum_{\{i|y_i=-1\}} x_i = (\frac{22}{3}, \frac{8}{3}) \end{aligned} \tag{2.5}$$

where n_+ and n_- denote the number of elements with labels $y_i = +1$ and $y_i = -1$ respectively.

In order to classify $x_\star$ correctly, we want to know if the vector between $c := (c_+ + c_-)/2 = (4.92, 2.83)$ and $x_\star$ encloses an angle bigger or smaller than $90° = \pi/2$ with the vector $w := c_+ - c_- = (-4.83, 0.3)$.

The dot product provides an easy way to compute this angle and therefore we just have to evaluate the so called **decision function** (**??**) to determine the labelling $y_\star$ of the new point $x_\star$:

$$\begin{aligned} f(x_\star) := y_\star &= \operatorname{sgn}\langle (x_\star - c), w \rangle \\ &= \operatorname{sgn}(-5.9) = -1 \end{aligned} \tag{2.6}$$

With this solution we can confirm our initial intuition that $y_\star = -1$, because the cosine switches sign from $+1$ to -1 at $\pi/2$, so the sign of our result tells us that $x_\star - c$ encloses an angle bigger than $\pi/2$ with the vector w.

Remark. *In this simple example we have encountered many things that will help us to understand the theoretical part of Support Vector Machines. We will now discuss some details that we will come across many times throughout this thesis.*

- *The decision of which class to assign to a new point is dependent on the dotted line in Figure* **??**. *Therefore this line is called* ***decision boundary****, which in our case is a so called hyperplane, separating objects classified as* $+1$ *from objects classified as* -1. *This term will be explained in more detail in Chapter* **??**, *but the main idea of separating will remain the same.*

- *The labelling decision according to (***??***) could also be formulated more generally in terms*

*of a kernel by using properties of the inner product, as the training pattern do not have to originally exist in a dot product space [**?**].*

$$
\begin{aligned}
f(x_\star) = y_\star &= sgn\langle (x_\star - c), w\rangle = \cdots = \\
&= sgn\left(\frac{1}{n_+}\sum_{\{i|y_i=+1\}} k(x_\star, x_i) - \frac{1}{n_-}\sum_{\{i|y_i=-1\}} k(x_\star, x_i) + b\right), \\
\text{where } b &:= \frac{1}{2}\left(\frac{1}{n_-^2}\sum_{\{(i,j)|y_i=y_j=-1\}} k(x_i, x_j) - \frac{1}{n_+^2}\sum_{\{(i,j)|y_i=y_j=+1\}} k(x_i, x_j)\right).
\end{aligned} \tag{2.7}
$$

*Later on we will see decision functions that have a similar structure to (**??**).*

- *Besides the similarities between this simple toy example and the things we will develop later on, the example also points out the following problem. Although the labelling decision in this kind of problem looks rather easy and one can see that the point belongs to -1 at the first glance, it will get more complicated if the situation is slightly changed as shown in Figure **??**, where only two points are moved in a way such that the mean stays the same but an important thing changes: One of the training patterns is now on the wrong side of the decision boundary. Evaluating the sign of this point with the decision function (**??**) will determine a wrong label. As this is not a desirable situation we want to construct an algorithm that can deal with this kind of problems or even avoid them.*

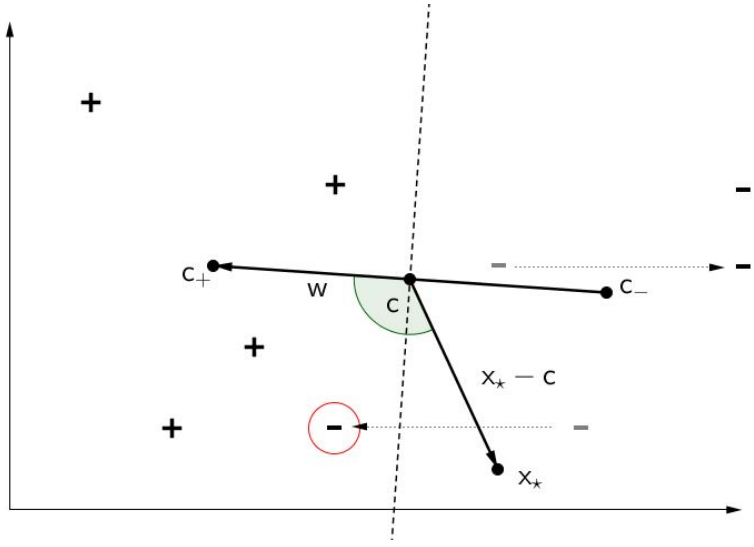


Figure 2.2: Problems of small changes for the thy example in $\mathbb{R}^2$

Therefore we first have to encounter what we expect from a good learning algorithm.

2.4 Characteristics of a Good Learning Algorithm

First of all, a learning algorithm that classifies patterns should do well on the training set [2] $\{x_1,\ldots,x_n\} \subseteq \mathscr{H}$, i.e. the empirical training error $\frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}|f(x_i)-y_i|$ should be minimized [3]. On the other hand we want the algorithm to classify the test patterns correctly too, which is not always implied by minimizing the training error though. Therefore we want our algorithm to be applicable to general situations and not only to particular chosen ones [**?**].

Statistical learning theory, also called VC theory [**?**], shows that it is necessary to restrict the set of functions from which the decision function f is chosen in order to get a function that is suitable for the available training data. Furthermore it provides upper bounds on the test error. [**?**].

The best-known capacity concept of this theory is the **VC dimension** h. It is defined as the largest n that can realise all different labellings of a set of n points. In that case, the function class is said to shatter the n points. If that is not possible it is defined to be ∞ [**?**].

Example. For assigning three points to two classes there are $2^3 = 8$ ways, as we can see in Figure **??**. For the displayed points in $\mathbb{R}^2$, all 8 possibilities can be realized using separating hyperplanes, this means that this function class can shatter 3 points. This would not be the case if we were given 4 points, no matter how we would place them. Therefore, the VC dimension of the class of separating hyperplanes in $\mathbb{R}^2$ is $h = 3$ [**?**].

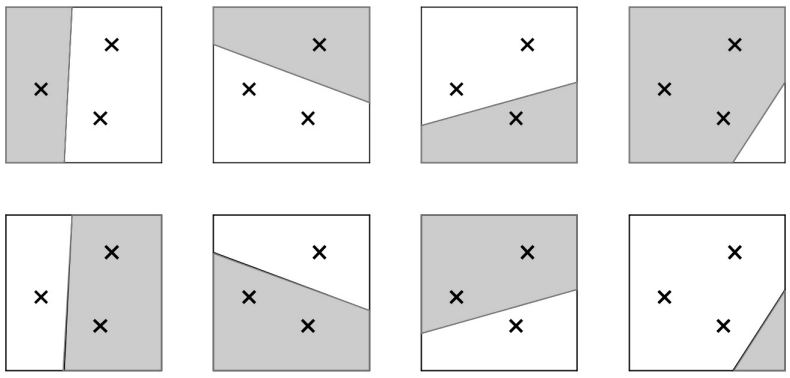


Figure 2.3: Shattering three points in $\mathbb{R}^2$

[2] For convenience we will follow the convention, that x is always interpreted as the vectorial representation in the feature space $\mathscr{H}$.

[3] The training error $\frac{1}{2}|f(x_i)-y_i|$ is zero if the pattern is classified correctly and one otherwise.[**?**].

In general, the VC dimension of an unrestricted n-dimensional linear classifier $f(x) = \langle w,x \rangle + b$, that we will discuss in Chapter **??** in more detail, is given by $h = n+1$ [**?**].

We will not go deep into the theory of VC dimensions, but we will briefly explain how it affects the upper bound of the test error: If the VC dimension h of the class of functions that the learning machine can implement is smaller than the size of the training set n, i.e. $h < n$, then independently of the distribution P that is generating the training data, the test error $R(f)$, i.e. the misclassification of an unseen point that is generated by the same distribution, can be bounded by (**??**) with a probability not lower than $1-\delta$ [**?**], [**?**].

$$R(f) \leq \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}|f(x_i)-y_i| + \sqrt{\frac{1}{n}\left(h\left(\ln\frac{2n}{h}+1\right)+\ln\frac{4}{\delta}\right)} \tag{2.8}$$

If we do not restrict the VC dimension of the class of functions, we can always build an algorithm that achieves zero training error. But due to the fact that the second term in (**??**) increases monotonically with the VC dimension h, a small training error does not guarantee a small test error [**?**].

Another instance one has to think about when it comes to pattern recognition is that we want to implement these classification algorithms. Therefore we additionally want them to be as efficient as possible [**?**] while keeping a good generalization ability.

We will now discuss this "efficiency" briefly by looking at the three classification types in Figure **??**[4], where the task was, just as in the toy example, to separate the crosses from the circles.

The main difference between the two examples is that in Figure **??** the training points were linearly separable, i.e. explained simplified, one could draw a straight line in $\mathbb{R}^2$ that separates the points into the two classes, but in Figure **??** this is not the case anymore [5]. In Figure **??** there are shown three possibilities to deal with the non-separability:

- In the first picture, we almost achieved a linear separation, but at the price that many points which should be classified correctly, are misclassified, even if they are far away from the decision boundary.
- In the second situation, the separation is not linear, but we made a decision boundary which

[4]The idea for this example was taken from [**?**], extended by own figures.

[5]In Chapter **??** we will discuss a trick where linear separation remains possible in a higher dimensional space.

classifies many points correctly without being too complex.

- The third picture, shows a perfect separation of crosses and circles, but this decision boundary may not have a good generalization ability, as potential outliers get their own decision region. Thus, classifying a test point correctly will be hard with this kind of decision boundary.

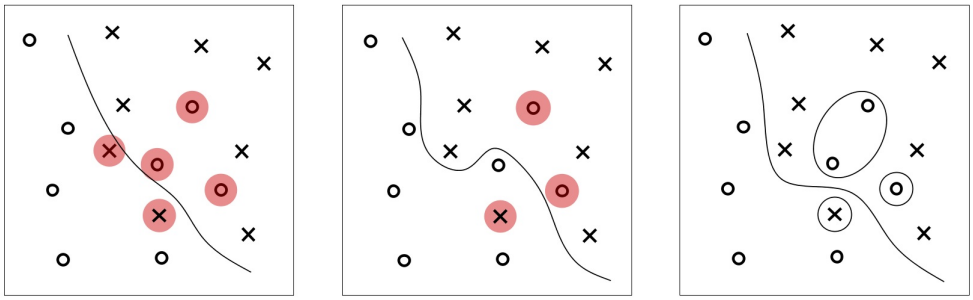


Figure 2.4: Trade off between efficiency, generalization ability and training errors

Having these three situations in mind, we have to decide how to deal with the trade off between efficiency, generalization ability and the number of training errors [**?**].

In general we want to create an algorithm that does well on the training set while it has a good generalization ability at the same time, in order to achieve good results on the training set but on the test set too. Furthermore, the algorithm should be easy to implement.

2.5 Summary

In machine learning for pattern recognition, our aim is to construct an algorithm that separates two different groups of training patterns in a way that it also allows for classifying unseen points. Within this framework one has to deal with several problems, including the question of what similarity is and how to find an appropriate method to state this similarity, how to construct classification functions that, using the chosen similarity measure, having a good generalization ability, while being sufficiently precise and also how to deal with the resulting training and test errors.

3 Support Vector Machines

After we have discussed the basic principles of classification and similarity from the intuitive point of view, we will now extend our knowledge in order to develop the central terms of this thesis: Support Vectors and the resulting Support Vector Machines. In this chapter we will first revise some basic terms from the field of linear algebra that we were already informally using in Chapter **??** and develop an optimization problem dealing with the binary case of pattern recognition.

3.1 The Canonical Hyperplane

Already in the Toy Example in Chapter **??** we informally encountered the term of a separating hyperplane. A formal definition is given as follows.

Definition 3.1. *Let a dot product space $\mathscr{H}$ and a set of pattern vectors $\{x_1,\ldots,x_n\} \subseteq \mathscr{H}$ be given. Any hyperplane in $\mathscr{H}$ can be written as*

$$H = \{x \in \mathscr{H} | \langle w,x\rangle + b = 0\},\ w \in \mathscr{H}, b \in \mathbb{R} \tag{3.1}$$

*with w a vector orthogonal to the hyperplane [**?**], [**?**].*

Given this hyperplane we can now distinguish points in three ways (Figure **??**):

- The point x satisfies $\langle w,x\rangle + b < 0$, then the point lies on the "left" side of the hyperplane, or
- the point x satisfies $\langle w,x\rangle + b > 0$, then it is on the "right" side of the hyperplane, or
- the point x lies on the hyperplane, then $\langle w,x\rangle + b = 0$.

Remark. *The terms "left" and "right" are just depending on how the hyperplane is oriented.*

In the situation we are facing in Figure **??**, where the hyperplane separates the two groups without any error, the set is called **(linearly) separable** and the hyperplane is called **separating hyperplane** [**?**].

Separating the groups via a separating hyperplane is the first step to create an algorithm that labels unseen points correctly. Though, we do not want to find an *arbitrary* separating hyperplane, but one that has a certain distance to the proximal point [6], i.e. the points closest to the

[6] The reason for this will be discussed in Chapter **??**.

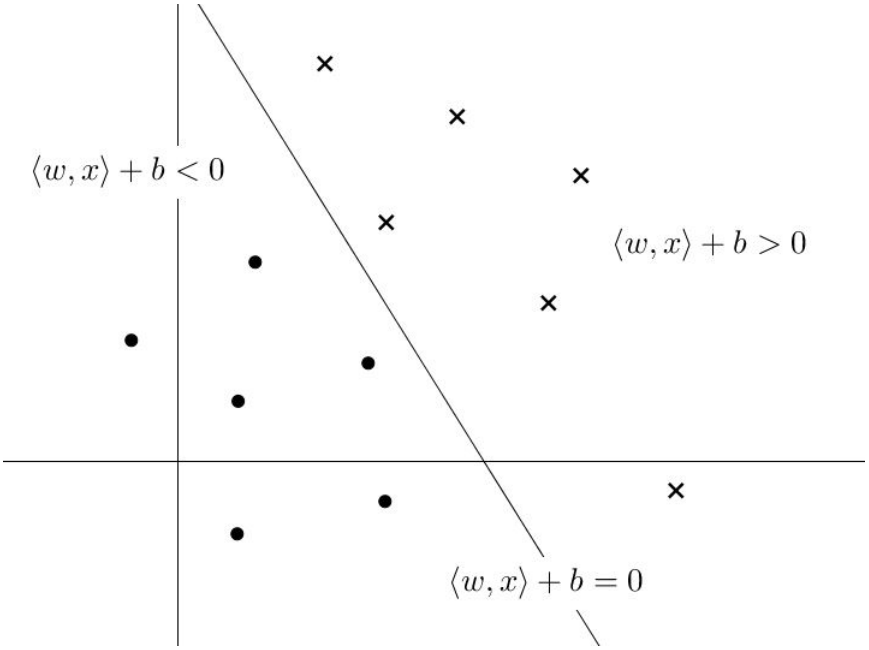


Figure 3.1: Separating hyperplane

hyperplane. For this reason we first have to get rid of the fact that in the formulation (**??**), the representation of the hyperplane is not unique.

Looking at the concept of a hyperplane in (**??**), we notice that if we multiply the orthogonal vector w and the offset parameter b by the same nonzero constant, we get exactly the same hyperplane just represented in terms of different parameters [**?**]. To eliminate this ambiguity we are scaling the two parameters in such a way that we can formulate a unique representation of the hyperplane. This is done by defining the so called **canonical hyperplane**:

Definition 3.2. *[**?**] The pair $(w,b) \in \mathscr{H} \times \mathbb{R}$ is called the canonical form of the hyperplane (**??**) with respect to $x_1, \ldots, x_n \in \mathscr{H}$, if it is scaled such that*

$$\min_{i=1,\ldots,n} |\langle w, x_i\rangle + b| = 1. \tag{3.2}$$

By requiring the scaling of w and b to be such that the point(s) closest to the hyperplane satisfy $|\langle w, x_i\rangle + b| = 1$, we get the canonical form of a hyperplane which can be seen in Figure 3.2.

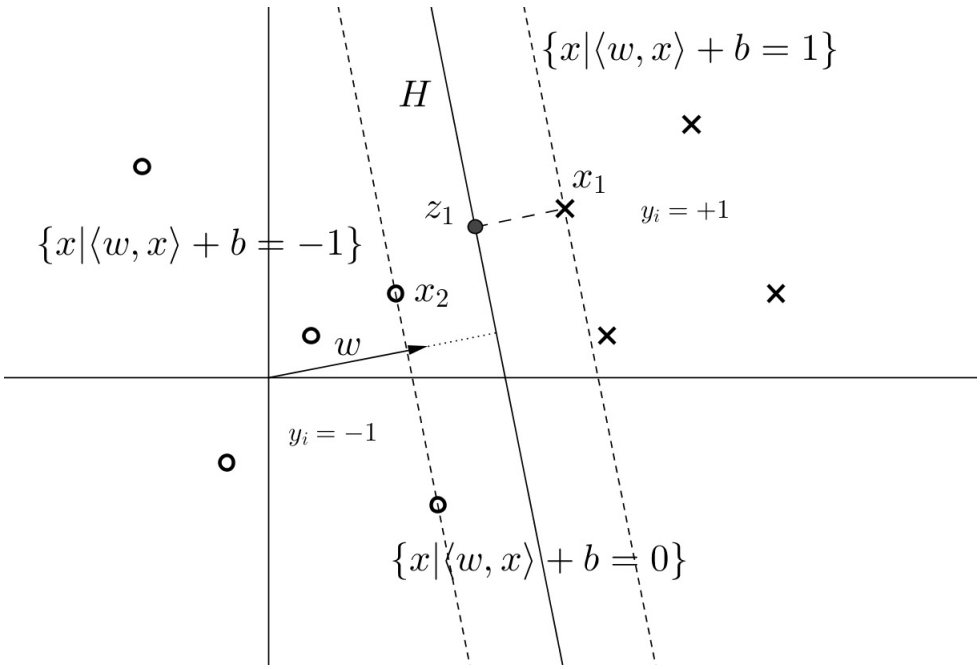


Figure 3.2: Canonical form of a separating hyperplane

Remark. *Definition* **??** *implies that the point(s) closest to the hyperplane have a distance of* $\frac{1}{||w||}$ *[***?***].*

Proof. Let δ_i be the normal distance between x_i and the hyperplane H, i.e. $\delta_i = \text{dist}(x_i, H)$ and $z_i \in H$ with $z_i - x_i = \lambda_i w$ for $\lambda_i \in \mathbb{R}$. Then it holds that $\delta_i = \lambda_i ||w|| = ||z_i - x_i||$. But

$$\begin{aligned} \lambda_i \, ||w||^2 &= w'(z_i - x_i) \\ \lambda_i \, ||w||^2 &= -b - w' x_i, \end{aligned} \tag{3.3}$$

as z_i lies on the hyperplane therefore has to satisfy (**??**). This implies $|\lambda_i| \, ||w||^2 = |\langle w, x_i\rangle + b|$ and consequently, for the canonical hyperplane, it follows that $|\lambda_i| \, ||w||^2 \geq 1$ which in turn implies $\delta_i \geq \frac{1}{||w||}$. Now let x_j satisfy $1 = \min_{i=1,\ldots,n} |\langle w, x_i\rangle + b| = |\langle w, x_j\rangle + b|$, then

$$\begin{aligned} |\lambda_j| \, ||w||^2 &= 1 \\ |\lambda_j| \, ||w|| &= \frac{1}{||w||} \\ \delta_j &= \frac{1}{||w||}, \end{aligned} \tag{3.4}$$

which proves that x_j is a proximal point and that x_j has a distance of $\frac{1}{||w||}$ to the hyperplane. □

Remark. *Let (w,b) be a canonical hyperplane. By the definition above also $(-w,-b)$ is satisfying (**??**).*

*For the purpose of pattern recognition, these two hyperplanes are considered as different because they are oriented differently and correspond to two decision functions [**?**].*

Having found the hyperplane which has distance $\frac{1}{||w||}$ to its closest point(s), the question is what we want to do with this information. Would we like the distance to be large or rather to be small? Do we need all points for defining the hyperplane? These questions will be answered in the next chapters.

3.2 The Role of the Margin

In this chapter we will have a closer look at the distance between the hyperplane and its closest point(s), called the **margin**. In particular we want to find out which properties the margin has to fulfil in order to help us finding an algorithm that separates given points with an error as small as possible and classifies most of the unseen points correctly [**?**].

Before we formally define the margin, we will have a closer look at an example [**?**] which makes clear why the margin plays an important role in pattern recognition.

Example. We are given some points assigned to two separable classes. These points are generated by some unknown but fixed distribution. Now we are given a new point generated by the same distribution. Therefore it seems likely that this point will lie close to at least one of the training points. This fact leads to the intuition that if we are able to separate the training points with a large margin, we will also classify the test point correctly.

In Figure **??** we are facing a separation problem where test patterns are generated by adding some noise $|r| > 0$ to the training patterns. Thus all test points lie within a circle with a training point as center and a radius $|r|$.

If the points are separated with a margin larger than r, all test points that were generated by adding noise to the original points will be classified correctly. As we do not make any assumptions regarding the size of r, but only that it is finite, it seems reasonable that we separate the points with a margin that is maximal to ensure that the test points will be classified correctly as it shown in Figure **??**.

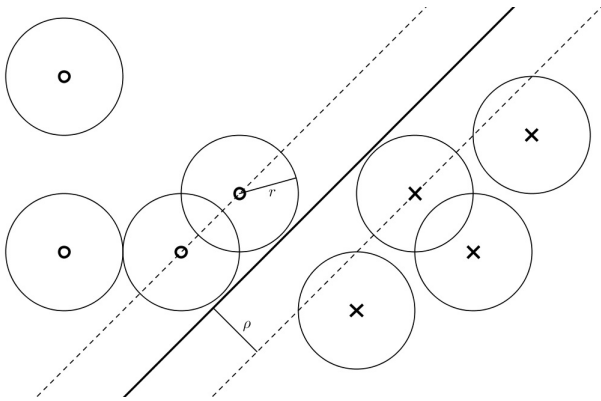


Figure 3.3: Justifying a large margin

Considering the perturbation the other way round, i.e. if the parameters w and b on which the hyperplane is dependent are slightly perturbed, we can formulate another hypothesis why it is reasonable to separate training data with a margin as large as possible (Figure 3.4).

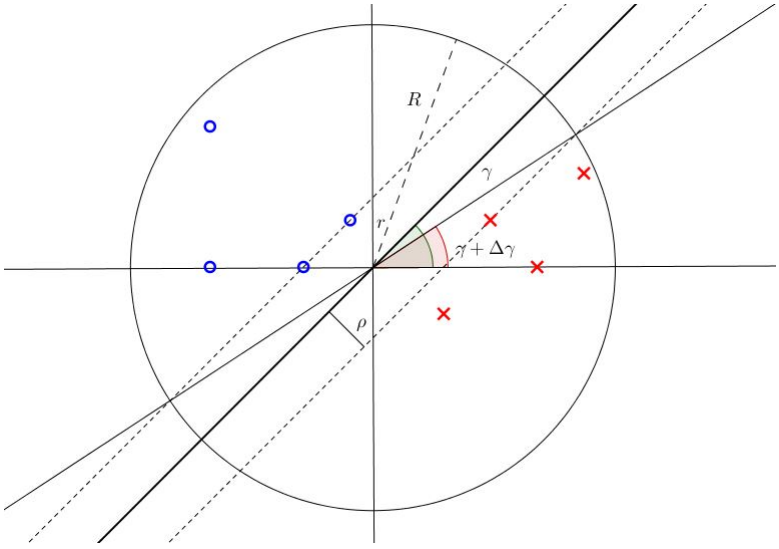


Figure 3.4: Hyperplane perturbation

Suppose the patterns are bounded in length by some $R > 0$ and the closest point has distance ρ from the hyperplane, i.e. the margin is equal to ρ as it is the case in Figure 3.4. In this case, we can rotate the hyperplane by some angle $|\delta\gamma| < \arcsin\frac{\rho}{R}$, without having an error in the classification of the points. [**?**].

Having found some intuitive justifications for large margins, we now want to give a formal

definition of this term.

Definition 3.3. *For a hyperplane* $\{x \in \mathscr{H} | \langle w,x\rangle + b = 0\}$ *we call*

$$\rho_{(w,b)}(x,y) := y\frac{\langle w,x\rangle + b}{||w||} \tag{3.5}$$

the ***geometrical margin of the point*** $(x,y) \in \mathscr{H} \times \{\pm 1\}$. *The minimum value*

$$\rho_{(w,b)} := \min_{i=1,\ldots,n} \rho_{(w,b)}(x_i,y_i) \tag{3.6}$$

is called the ***geometrical margin of the training set*** $(x_1,y_1),\ldots,(x_n,y_n)$. *For writing reasons we will omit the terms geometrical and training set and simply refer to it as margin [**?**].*

Remark. *The margin is negative if and only if misclassification occurs.*

Considering the following Theorem we will have another, more mathematically justification for a large margin:

Theorem 3.1. *For simplicity, consider the set of decision functions* $f_{w,0}(x) = sgn\langle w,x\rangle$, *where the offset parameter b is set to zero. Additionally let the length of w and x be bounded by some constants, i.e.* $||w|| \leq \Lambda$ *and* $||x|| \leq R$, *for R,* $\Lambda > 0$. *Moreover, let the margin* ρ *be positive, and let* ν *denote the fraction of training examples with margin smaller than* $\frac{\rho}{||w||}$, *referred to as the margin error [**?**].*
*Independent of the (fixed) distribution that is generating the data, there exists a constant c such that the probability of a test error, i.e. that a test pattern generated by the same distribution is misclassified, is bounded by (**??**) with a probability of at least* $1-\delta$ *with* $\delta \in (0,1)$ *[**?**], [**?**].*

$$R(f_{w,0}) \leq \nu + \sqrt{\frac{c}{n}\left(\frac{R^2\Lambda^2}{\rho^2}\ln^2 n + \ln\frac{1}{\delta}\right)} \tag{3.7}$$

Remark. *In the second chapter we already discussed an error bound* [7] *that was more general as it contained the abstract expression h for the VC dimension. By restricting the training patterns to lie within a finite sphere we can formulate an upper bound on this VC dimension* [8]*, obtaining the former error bound [**?**].*

Analysing equation (**??**), we can easily make some statements about the upper bound on the test error [**?**]:

[7] For more information about error bounds for Support Vector Machines see [**?**].
[8] We will develop this bound in Chapter **??**

- The test error is bounded by the margin error ν plus a term that tends to zero for the number of training patterns n tending to infinity [9].
- The square root term will be small if the fraction $\frac{R^2\Lambda^2}{\rho^2}$ is small, i.e. if the training data is not widespread, the orthogonal vector w is small and the margin ρ is large.
- Having a large margin ρ on the other hand, will cause the number of training patterns that are closer than $\frac{\rho}{||w||}$ to the hyperplane to increase.

Handling this trade off we can formulate the optimization problem that will be investigated in the next section: Separate the training data with as few errors as possible while keeping a large margin [**?**].

3.3 Optimal Margin Hyperplanes: Binary Support Vector Machines with a Hard Margin

In this chapter we will derive the optimization problem that has to be solved for computing the **optimal hyperplane**, namely the hyperplane that separates the training data without error and maximizes the distance to the closest training pattern(s) [**?**].

Theorem 3.2. *For canonical hyperplanes, maximizing the geometrical margin ρ is the same as minimizing $||w||$ if the data is linearly separable.*

Proof. Let $\rho_{(w,b)}$ be the margin of the training set $(x_1,y_1),\ldots,(x_n,y_n)$. Then it holds that

$$\begin{aligned}\rho_{(w,b)} &= \min_{i=1,\ldots,n} \rho_{(w,b)}(x_i,y_i) \\ &\overset{(??)}{=} \min_{i=1,\ldots,n} \frac{y_i(\langle w,x_i\rangle + b)}{||w||} \\ &\overset{(??)}{=} \frac{1}{||w||}.\end{aligned} \tag{3.8}$$

Therefore minimizing the length of w is equivalent to the maximization of the margin $\rho_{(w,b)}$. □

The objective function of our optimization problem for the optimal hyperplane will therefore be the minimization of $||w||$. But as the term "optimal hyperplane" refers to a hyperplane that

[9] This sounds reasonable, but as we will discuss later, for implementing Support Vector Machines, storing a big matrix is necessary, therefore this should not be the only possibility to reduce the test error [**?**]. Secondly, collecting large training samples is impossible for real world problems. Furthermore, in practice, a good generalization ability was often achieved while working only with subsets of the training patterns [**?**].

separates the training data $(x_1,y_1),\ldots,(x_n,y_n)$ without any error, we have to restrict the optimization problem. Therefore we are including an inequality constraint assuring that no training error is made: $y_i(\langle w,x_i\rangle + b) \geq 1$ [**?**].

Model 1 (Primal optimization problem for binary SVM with hard margin)**.**
Given a set of training points $(x_1,y_1),\ldots,(x_n,y_n)$, $x_i \in \mathscr{H}$, $y_i \in \{\pm 1\}$ and assuming that both classes are not empty, we want to find a decision function that classifies patterns according to $f_{w,b}(x) = sgn(\langle w,x\rangle + b)$, i.e. that determines on which "side" of the hyperplane a point x lies. Additionally, this function must not make any training error therefore it has to satisfy $f_{w,b}(x_i) = y_i$ for $i = 1,\ldots,n$. Assuming that the training set is separable, i.e. there exists such a function, we can formulate the **primal optimization problem for binary Support Vector Machines with hard margin** [**?**]:

$$\begin{aligned} \min_{w\in\mathscr{H},\ b\in\mathbb{R}} \quad & \frac{1}{2}||w||^2 \\ \text{subject to} \quad & y_i(\langle x_i,w\rangle + b) \geq 1 \quad \forall i = 1,\ldots,n. \end{aligned} \tag{3.9}$$

Remark. *Including the scalar $\frac{1}{2}$ and squaring the term that we want to minimize is done only for practical reasons. The solution of the optimization problem is the same as the one for minimizing only $||w||$.*

In order to solve Model **??**, one has to calculate the Lagrangian function (**??**), minimize it with respect to w and b and maximize it with respect to the Lagrangian multipliers $\alpha_i \geq 0$ to obtain the saddle point of (**??**) [**?**].

$$L(w,b,\alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i(y_i(\langle x_i,w\rangle + b) - 1) \tag{3.10}$$

At the saddle point given by the optimal solution $w^\star$, $b^\star$, $\alpha^\star$, the derivatives of L with respect to the primal variables $b^\star$ and $w^\star$ must vanish:

$$\frac{\partial}{\partial b^\star}L(w^\star,b^\star,\alpha^\star) = 0, \quad \frac{\partial}{\partial w^\star}L(w^\star,b,\alpha^\star) = 0 \tag{3.11}$$

which leads to

$$\sum_{i=1}^{n} \alpha_i^\star y_i = 0 \quad \text{and} \quad w^\star = \sum_{i=1}^{n} \alpha_i^\star y_i x_i, \quad \alpha_i^\star \geq 0 \tag{3.12}$$

for $i = 1,\ldots,n$.

According to (**??**), the optimal hyperplane is a linear combination of vectors of training patterns. But not all training patterns $x_1, \ldots, x_n$ have a Lagrangian coefficient $\alpha_i \neq 0$, consequently only those patterns x_i for which $\alpha_i > 0$ will have an impact on the position of the optimal separating hyperplane. Therefore these pattern vectors are called **Support Vectors** (SV). These vectors are the ones that fulfil the inequality constraint in Model **??** with equality. As the other points do not influence the optimal hyperplane we can "ignore" the points x_i with $\alpha_i = 0$ and obtain

$$w^\star = \sum_{SV} \alpha_i^\star y_i x_i, \quad \alpha_i^\star \geq 0. \tag{3.13}$$

According to the KKT Theorem[10] the optimal hyperplane formally has to satisfy the complementary slackness condition

$$\alpha_i^\star [y_i(\langle x_i, w^\star \rangle + b^\star) - 1] = 0, \tag{3.14}$$

for all $i = 1, \ldots, n$ [**?**]. Again we notice that only patterns x_i with a nonzero coefficient α_i make non-trivial contributions to (**??**).

Substituting the conditions (**??**) into the Lagrangian function (**??**), we obtain the so called dual optimization problem.

Model 2 (Dual optimization problem for binary SVM with hard margin)**.**
Let a linearly separable set of training points $(x_1, y_1), \ldots, (x_n, y_n) \in \mathscr{H} \times \{-1, +1\}$ be given and let $\alpha^\star$ solve the following optimization problem:

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^m} \quad & W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle, \\ \text{subject to} \quad & \alpha_i \geq 0, \quad i = 1, \ldots, n, \\ & \sum_{i=1}^n \alpha_i y_i = 0. \end{aligned} \tag{3.15}$$

The optimal hyperplane with margin $\frac{1}{||w^\star||}$ is obtained by the weight vector $w^\star = \sum_{i=1}^n y_i \alpha_i^\star x_i$ [**?**].

[10] The KKT conditions can be found for instance in [**?**].

New patterns are classified [**?**] according to the decision function

$$f(x) = \operatorname{sgn}\left(\sum_{SV} y_i \alpha_i^\star \langle x, x_i \rangle + b^\star\right). \tag{3.16}$$

Remark. *In the dual representation of the binary SVM optimization problem, the value b does not appear, but it can be calculated using the complementary slackness condition (**??**):*

$$b^\star = -\frac{1}{2}\left(\langle w^\star, x_i \rangle + \langle w^\star, x_j \rangle\right) \tag{3.17}$$

*where x_i is any [11] Support Vector in the class $+1$ and x_j any Support Vector belonging to the class -1 [**?**], [**?**].*

Support Vectors and the Test Error Bound

Using the number of Support Vectors we can now formulate an upper bound on the error of the optimal hyperplane, the so called **leave one out bound** [**?**].

It will give us a good indication of the true test error because we are looking how the result obtained by Model **??** resp. Model **??** is changing if we leave one training pattern out.

When leaving out a pattern x_i and constructing the solution with the remaining ones, the following outcomes are possible when classifying x_i:

1. $y_i(\langle x_i, w \rangle + b) > 1$. The left out pattern is classified correctly and does not lie on the margin. Thus it would not have become a Support Vector anyway.

2. $y_i(\langle x_i, w \rangle + b) = 1$. The pattern x_i lies on the margin. Though, the solution w would not change, because, even if it would have become a Support Vector when kept in the training set, the fact that the solution does not change, means that it can be written as $\sum_{SV} \beta_i y_i x_i$ with some $\beta_i \geq 0$.

3. $0 < y_i(\langle x_i, w \rangle + b) < 1$. Although the pattern is classified correctly, it lies within the margin, therefore the solution would have been a different one if x_i would have been kept in the training set because x_i would have become a Support Vector.

4. $y_i(\langle x_i, w \rangle + b) < 0$. In this case the pattern x_i is not classified correctly. If the removed point has been a Support Vector, the solution of the Support Vector Machine that was trained on

[11] Any means we need only **one** pattern of each type.

the remaining training points, would have been different to the one obtained training it with all available points. If the removed point has not been a Support Vector, the solution does not change.

Repeating this "leave-one-out-procedure" and averaging the resulting errors yields an upper bound for the generalisation error:

Theorem 3.3. *Let the training points be separated by the optimal hyperplane. The expected value of an error on the test set is bounded by*

$$\mathbb{E}[\mathbb{P}(\textit{test error})] \leq \frac{\mathbb{E}[\textit{number of support vectors}]}{(\textit{number of training vectors}) - 1}. \tag{3.18}$$

Therefore the generalization ability of an optimal hyperplane which is constructed by a small number of support vectors will be high, independently of the dimensionality of the feature space $\mathscr{H}$ [**?**], [**?**],[**?**].

3.4 Non-linear Support Vector Machines: The Kernel Trick

So far, we have discussed the case where the training patterns were linearly separable in the feature space $\mathscr{H}$. For many problems this will be not the case, but instead of inventing a new classification algorithm we will use so called kernels to non-linearly transform the patterns $x_1, \ldots, x_n \in \mathscr{H}$ into a higher dimensional space $\mathscr{G}$, where linear separation is possible. In this higher dimensional space things we have stated in the last chapter can be applied, therefore almost the same optimization problems will be the result of this so called **Kernel Trick** [**?**].

The meaningfulness of this procedure comes clear by Cover's Theorem [**?**] which basically states that the number of separations increases with the dimensionality of the space where the training data is represented. Furthermore it makes the use of a larger decision function class possible [**?**].

Example. In Figure **??** we are faced with a separation problem that is not linear in $\mathbb{R}^2$. Therefore we use a map $\Phi(x_1, x_2) = (x_1, x_2, x_1^2 + x_2^2)$ that transforms our data into $\mathbb{R}^3$, where we can clearly do a linear separation. Going back to the input space, the decision boundary which was a hyperplane in $\mathbb{R}^3$ is now a circle in $\mathbb{R}^2$. By mapping the data in a higher dimensional space we get the non-linear separation with no effort.

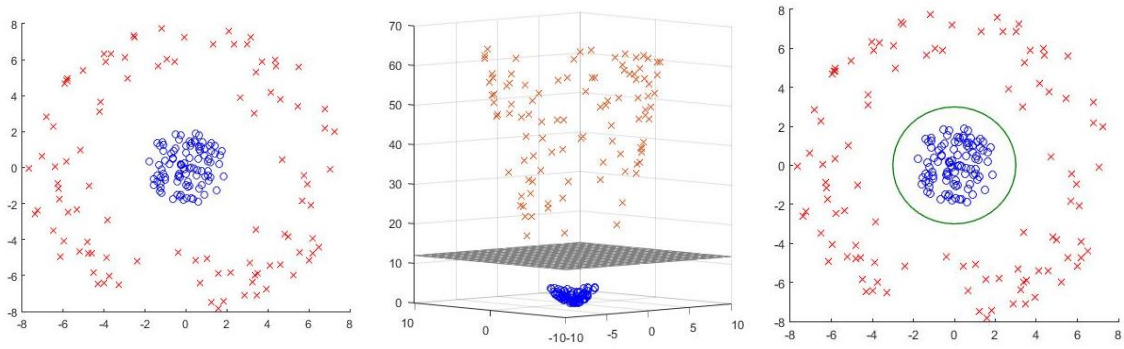


Figure 3.5: Kernel Trick with $\Phi(x_1,x_2) = (x_1,x_2,x_1^2+x_2^2)$

Before we can work with the kernel trick, we have to justify that we can find such kernel functions or find properties of them, that correspond to a dot product in the higher dimensional feature space $\mathscr{G}$.

In the optimization problems in Chapter **??** we did not make any assumptions on the dimensionality of the feature space $\mathscr{H}$, we only had the requirement that it is equipped with a dot product. Therefore, the training data we worked with could also be the result of a mapping as used in the example. We can equally well use $\Phi(x)$ whenever we made a statement about x. Applying this to the optimization problem in Model **??** or Model **??** resp., we get a decision function of the form

$$\begin{aligned} f(x) &= \operatorname{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i \langle \Phi(x), \Phi(x_i)\rangle + b\right) \\ &= \operatorname{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i k(x,x_i) + b\right) \end{aligned} \tag{3.19}$$

where $\langle \Phi(x), \Phi(x_i)\rangle =: k(x,x_i)$ for $i = 1,\ldots,n$.

As we have been calculating dot products in the original feature space, we now have to justify that this is also possible with the kernel in the higher dimensional feature space. Using Mercer's Theorem we can state a necessary and sufficient condition for a function having a dot product expansion.

Theorem 3.4. ***(Mercer)*** *[**?**] A symmetric L^2 function $k(u,v)$ has an expansion*

$$k(u,v) = \sum_{i=1}^{\infty} a_k \phi_k(u) \phi_k(v)$$

with positive coefficients $a_k > 0$, i.e. describes a dot product on some feature space, whenever the necessary and sufficient condition

$$\int\int k(u,v)g(u)g(v)dudv > 0$$

is valid for all $g \neq 0$ for which

$$\int g^2(u)du < \infty.$$

With the help of the so called **convolution of the inner product** we can now formulate decision functions of the form

$$f(x) = \operatorname{sgn}\left(\sum_{\mathrm{SV}} y_i \alpha_i k(x_i, x) + b\right) \tag{3.20}$$

which are linear in the higher-dimensional feature space and nonlinear in the original feature space space.

The optimization problem in Model **??** only changes by one term. Therefore the α_i, $i = 1,\ldots n$, in the non-separable case can be found by solving the following convex [12], quadratic minimization problem [**?**]:

$$\begin{aligned} \max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \\ \text{subject to} \quad & \alpha_i \geq 0, \quad i = 1,\ldots,n \\ & \sum_{i=1}^{n} \alpha_i y_i = 0 \end{aligned} \tag{3.21}$$

Again only nonzero Lagrangian multipliers α_i, $i = 1,\ldots,n$, contribute non-trivially to the hyperplane, consequently we obtain

$$w = \sum_{SV} \alpha_i y_i \Phi(x_i). \tag{3.22}$$

Remark. *Like in the separable case we can calculate the threshold b by using the necessary and sufficient condition for the optimal hyperplane (**??**) including the adjustment of the kernel*

[12] The optimization problem is convex since $\sum_{i,j}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j) = \langle \sum_{i=1}^{n} \alpha_i y_i \Phi(x_i), \sum_{j=1}^{n} \alpha_j y_j \Phi(x_j) \rangle \geq 0$ [**?**].

function. As $\alpha_j > 0$ *implies* $\sum_{i=1}^{n} y_i \alpha_i k(x_j, x_i) + b = y_j$*, the intercept* b *can be calculated by averaging*

$$b = y_j - \sum_{i=1}^{n} y_i \alpha_i k(x_j, x_i) \tag{3.23}$$

over all Support Vectors, i.e. all points with $\alpha_j > 0$*.*

For problems with a few hundred entities this optimization problem can be solved by any standard convex quadratic optimization algorithm such as the Newton method, conjugate gradient, or primal-dual interior point methods [**?**]. For larger problems this methods may have problems with storing the $n \times n$ matrix containing the values of the kernel function for all pairs of training points. Using the Sequential Minimization Optimization Method (SMO), which was developed particularly for Support Vector Machies, can ease the problem of handling this big matrix [**?**].

3.4.1 Kernels and the VC Dimension

When we discussed the characteristics of a good learning algorithm in Chapter **??**, we mentioned that the generalization ability should be high. This property was closely related to the VC dimension of the class of decision functions a Support Vector Machine can implement. This in turn is closely related to the type of kernel functions one introduces for a learning machine. First let us have a look at a formal bound on the VC dimension for the optimal hyperplane [**?**].

Theorem 3.5. *[**?**] Let the training data* $(x_1, y_1), \ldots, (x_n, y_n) \in \mathscr{H} \times \{-1, +1\}$ *be bounded by a sphere of radius R. The VC dimension h of a subset of (separating) hyperplanes* $f(x) = \operatorname{sgn}(\langle w, x \rangle + b)$ *satisfying* $||w|| \leq \Lambda$ *is bounded by the inequality*

$$h \leq \min(R^2 \Lambda^2, m) + 1, \tag{3.24}$$

where m is the dimension of the space $\mathscr{H}$*.*

With the smallest norm of weight w, as we have calculated for Model **??** resp. Model **??** [13], we also get a smallest upper bound on the VC dimension. This is the case because by minimizing $||w||$ we also minimize the right handside of (**??**). By having a minimal VC dimension in turn, we can construct learning algorithms with high generalization ability, i.e. the probability of test

[13] Actually this is the mathematical motivation why we are maximizing the margin in a statistical learning theory point of view.

errors is minimal.

For Support Vector Machines that use kernel functions that do not correspond to the dot product as similarity measure we are considering functions that satisfy the conditions $R^2||w||^2 \leq k$ in the set of canonical hyperplanes in the higher dimensional feature space, where R is the smallest sphere that contains all $\Phi(x_i)$, $i = 1,\ldots,n$ and $||w||$ is again the norm of the weights, but now for the hyperplane in the higher dimensional feature space. Applying Theorem 3.5 in the higher dimensional feature space, k gives an estimate of the VC dimension of the set of functions that was used.

Consequently, by estimating R^2 and $||w||^2$ which can be done by finding

$$R^2 = R^2(k) = \min_a \max_{x_i} [k(x_i,x_i) + k(a,a) - 2k(x_i,a)], \tag{3.25}$$

which will give a minimum value for the squared radius of a sphere around $\Phi(a)$ containing all $\Phi(x_i)$ for $i = 1,\ldots,n$, and

$$||w||^2 = \sum_{i=1}^{n} \alpha_i \alpha_j k(x_i,x_j) y_i y_j, \tag{3.26}$$

which can be derived from Model **??**, resulting in an estimation for the weights w, we can estimate the VC dimension. As the kernel function appears in both expressions, we can conclude that it has an impact on the generalization ability of a Support Vector Machine, consequently, they have to be chosen wisely. [**?**].

3.4.2 Popular Kernels

By now we have only used the dot product as a measure of similarity. In this context it is also referred to as **linear kernel**.

But there are other popular kernel functions that are widely used in the context of Support Vector Machines. In the following we will discuss the quadratic kernel, polynomial kernels of higher degree, the radial basis function (RBF) and also mention Two-Layer Neural Networks (NN) and their implications for the generalization ability of the learning algorithm.

Polynomial Kernel

The **polynomial kernel of degree** d uses the function

$$k(x,x_i) = [\langle x,x_i\rangle + 1]^d \tag{3.27}$$

for the convolution of the inner product and constructs a decision function of the form

$$f(x) = \operatorname{sgn}\left(\sum_{\mathrm{SV}} y_i\alpha_i[\langle x,x_i\rangle + 1]^d + b\right). \tag{3.28}$$

Although the dimensionality in the feature space could be high because polynomials of degree d in m-dimensional feature space have $\mathcal{O}(m^d)$ free parameters, one can use the choice of the best degree d to minimize the VC bound because both the radius R and the norm of the weights $||w||$ depend on the degree of the polynomial. Also a local polynomial approximation can be used to minimize $R^2||w||^2$ by a priori choosing the radius R_β of the sphere and doing an optimization according to the n_β training patterns that lie within this sphere, minimizing $\frac{R_\beta^2||w||^2}{n_\beta}$ [**?**].

The **quadratic kernel** is the special case where $d = 2$.

Radial Basis Function

Radial Basis Function (RBF) Machines with Gaussian kernel of width σ are using the function

$$k(x,x_i) = \exp\left[\frac{-(x-x_i)^2}{\sigma^2}\right], \tag{3.29}$$

for the convolution of the inner product [**?**], which leads to a decision function [**?**] of the form

$$f(x) = \operatorname{sgn}\left(\sum_{\mathrm{SV}} y_i\alpha_i \exp\left[\frac{-(x-x_i)^2}{\sigma^2}\right] + b\right). \tag{3.30}$$

Using this kernel, the VC dimension of the family of classifiers consisting of the resulting Support Vector Machines can be infinite. However, choosing the RBF widths can be a good way to control the VC dimension if the data is restricted to lie in a finite sphere R [**?**].

Two Layer Neural Networks

Also widely used in the field of Support Vector Machines is the so called **Two-Layer Neural Network**. It uses kernel functions

$$k(x, x_i) = S[\nu\langle x, x_i\rangle + c], \tag{3.31}$$

where $S(u)$ is a sigmoid function [14] which leads to a decision function

$$f(x) = \operatorname{sgn}\left(\sum_{\mathrm{SV}} \alpha_i S[\nu\langle x, x_i\rangle + c] + b\right). \tag{3.32}$$

Although good experience has been made with this kind of kernel [**?**], it also has some drawbacks. First of all the kernel does not always satisfy the Mercer Theorem. To overcome this, one has to choose particular values ν and c for the sigmoid kernel $S(u) = \tanh(\nu u + c)$ on $u \in (-1, 1)$ [**?**]. Furthermore, it can be the case that the solution to the optimization problem is only a local and not a global optimum [**?**].

As it is beyond the scope of this thesis we will not discuss them in more detail.

3.5 Soft Margin Hyperplanes

Until now, our attention was limited to the problem that we want to linearly separate training examples in two classes with no training error and with good generalization ability, no matter if this happens in the original feature space or the higher dimensional feature space using a kernel function. But as we discussed in Chapter **??**, when we were talking about the characteristics of a good learning algorithm, we already had the idea that we may not want to put the same weight of interest in each single training point.

Remark. *In this chapter we will only discuss soft margin hyperplanes without additionally using the kernel trick. In practice, often both are used simultaneously to achieve even better results.*

In order to do this we are constructing an algorithm that somehow "ignores" outliers or points that we do not want to affect the solution of the optimization problem, but for which the generalization ability remains high and the number of training errors remains rather small.

The first idea would be to find a hyperplane that separates the two classes with a minimal number of training errors. As this turns out to be a combinatorial problem it is hard to solve

[14] A sigmoid function S is an approximation of the indicator or in this context also called activation function, which responds whenever the input exceeds a certain threshold [**?**].

efficiently [**?**].

Therefore, we rather want to modify Model **??**, extending it by introducing so called **slack variables** $\xi_i \geq 0$, where $i = 1, \ldots, n$, This will allow training points to violate the hyperplane inequality constraint of Model **??** (**??**) [**?**], [**?**], [**?**].

This leads to a so called **soft margin hyperplane** where we allow the distance of the closest point to the hyperplane to be not strictly $1/||w||$ but smaller by the factor ξ_i.

$$y_i(\langle x_i, w\rangle + b) \geq 1 - \xi_i \; \forall \; i = 1, \ldots n. \tag{3.33}$$

Since this constraint can always be fulfilled by making ξ_i large enough, leading to the trivial solution, we have to include this term in the objective function in Model **??** as well, penalizing for each error made in separating the training patterns. As there are many ways to do this, we will now look at two of them, the **C-Support Vector Machine** (C-SVM) where the penalty parameter is given a priori and the **ν-Support Vector Machine** (ν-SVM) for which the penalty weights are additional decision variables which are data driven [**?**], [**?**], [**?**].

3.5.1 C-Support Vector Machine

In the simplest case, we add $\sum_i \xi_i$ multiplied with a weight of $\frac{C}{n}$ to the decision function (**??**), where C is some positive constant [15].

The value of C acts as a parameter for the importance of the number of training errors. If we want few training errors we have to choose C large in order to penalize each error, i.e. a point x_i where $\xi_i > 0$ comes with a huge cost in the objective function. If this property is not that important one can choose a smaller C which does not penalize the errors that strong. As we can see, one has always to deal with the trade off between maximizing the margin and minimizing the number of training errors [**?**].

The optimization problem for the so called **C-Support Vector Machine** is then stated as follows [**?**]:

Model 3 (Primal optimization problem for binary C-SVM)**.**

[15] Cortes and Vapnik [**?**] omitted the term n but this makes no difference as C can be chosen as an arbitrary positive constant.

$$
\begin{aligned}
\min_{w\in\mathscr{H},\xi\in\mathbb{R}^n} \quad & \tau(w,\xi) = \frac{1}{2}||w||^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \\
\text{subject to} \quad & y_i(\langle x_i, w\rangle + b) \geq 1-\xi_i \quad \forall i = 1,\ldots,n \\
& \xi_i \geq 0 \quad \forall i = 1,\ldots,n.
\end{aligned} \tag{3.34}
$$

Remark. *This kind of Support Vector Machine will generalize well if there is not much overlapping between the two classes, but if there is a strong overlap, for instance due to noise, there is no guarantee that the hyperplane will generalize well [**?**].*

In order to find the solution to this optimization problem, we proceed as in the separable case. First we formulate the Lagrangian, then take derivatives with respect to the decision variables and finally substitute the solution into the objective function to obtain the dual optimization problem [**?**] to Model **??**.

Model 4 (Dual optimization problem for binary C-SVM)**.**

$$
\begin{aligned}
\max_{\alpha\in\mathbb{R}^n} \quad & W(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle \\
\text{subject to} \quad & 0 \leq \alpha_i \leq \frac{C}{n} \quad \forall i = 1,\ldots,n \\
& \sum_{i=1}^{n}\alpha_i y_i = 0
\end{aligned} \tag{3.35}
$$

Solving this quadratic optimization problem yields the parameters α_i which are necessary to find the coefficients for the optimal hyperplane

$$
w = \sum_{i=1}^{n}\alpha_i y_i x_i, \tag{3.36}
$$

where only points (x_i, y_i) with $\alpha_i > 0$ make a real contribution as they are meeting the constraint (**??**) with equality. Again these vectors are called Support Vectors [**?**], [**?**], [**?**].

To compute the threshold b, we can proceed as in the separable case because for Support Vectors x_j with $\xi_j = 0$ we have

$$
\sum_{i=1}^{n} y_i\alpha_i\langle x_j, x_i\rangle + b = y_j. \tag{3.37}
$$

Therefore b can be calculated by averaging

$$b = y_j - \sum_{i=1}^{n} y_i \alpha_i \langle x_j, x_i \rangle \tag{3.38}$$

over all Support Vectors x_j with $0 < \alpha_j \leq \frac{C}{n}$ [**?**].

Similar to the separable case, the decision function takes the form (**??**) and also the inner products taken as similarity measure can be replaced by any kernel function that satisfies the Mercer Theorem.

Remark. *The value of C has to be chosen a priori and to obtain the optimal value for this penalizing parameter one has to go through a wide range of values in the training part in order to get the optimal C, i.e. the one that gives the smallest test error [**?**], [**?**].*

3.5.2 ν-Support Vector Machine

Another way of including the margin errors into the objective function was proposed by Schölkopf, Smola, Williamson and Bartlett [**?**] in order to get rid of the unintuitive a priori choice of the parameter C [**?**]. It is replaced by a parameter ν which is able to take the number of margin errors and Support Vectors into account.

For this type of Support Vector Machine we will consider the following primal optimization problem [**?**]:

Model 5 (Primal Optimization problem for the binary ν-SVM)**.**

$$\begin{aligned} \min_{w \in \mathscr{H},\ \xi \in \mathbb{R}^n,\ \eta,\ b \in \mathbb{R}} \quad & \tau(w, \xi, \eta) = \frac{1}{2}||w||^2 - \nu\eta + \frac{1}{n}\sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & y_i(\langle x_i, w \rangle + b) \geq \eta - \xi_i \quad \forall i = 1, \ldots, n \\ & \xi_i \geq 0 \quad \forall i = 1, \ldots, n \\ & \eta \geq 0 \end{aligned} \tag{3.39}$$

In [**?**] it was proven that the parameter ν is an upper bound for the fraction of margin errors and also a lower bound on the fraction of Support Vectors.

Since these values are not chosen a priori, this choice of the penalty parameter does not afford to heuristically search for the optimal value.

The dual problem to Model **??** can be derived as in the C-SVM case by computing the Lagrangian, minimizing with respect to the primal variables w, ξ, b and η and maximizing with respect to the Lagrangian variables [**?**]. This procedure leads to the following optimization problem:

Model 6 (Dual optimization problem for the binary ν-SVM)**.**

$$\begin{aligned} \max_{\alpha\in\mathbb{R}^n} \quad & W(\alpha) = -\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle \\ \text{subject to} \quad & 0 \le \alpha_i \le \frac{1}{n} \quad \forall i = 1,\dots,n \\ & \sum_{i=1}^{n}\alpha_i y_i = 0 \quad \forall i = 1,\dots,n \\ & \sum_{i=1}^{n}\alpha_i \ge \nu \quad \forall i = 1,\dots,n. \end{aligned} \tag{3.40}$$

As in the C-SVM the a new pattern is classified according to the decision function [**?**]

$$f(x) = \operatorname{sgn}\left(\sum_{\mathrm{SV}}\alpha_i y_i \langle x, x_i\rangle + b\right). \tag{3.41}$$

Remark. *Compared to the dual problem of the C-SVC, the objective function of the ν-SVC does not include the term $\sum_{i=1}^{n}\alpha_i$. Furthermore there is an additional constraint. However, it can be shown that for a C-SVC classification with $C = \frac{1}{\eta}$ both classification methods lead to the same decision function [**?**],[**?**].*

Both the threshold b and the parameter η can be computed by the following procedure: Let us consider sets S_+ and S_- of the same size $s > 0$. S_+ contains the Support vectors x_i with $0 < \alpha_i < 1$ and $y_i = +1$ and S_i the Support Vectors x_i with $0 < \alpha_i < 1$ and $y_i = -1$. According to the KKT conditions the inequality constraint for the hyperplane in the primal optimization problem becomes an equality with $\xi_i = 0$ [**?**]. Therefore we have the following expressions for b and η:

$$\begin{aligned} b &= -\frac{1}{2s}\sum_{x\in S_+\cup S_-}\sum_{j=1}^{n}\alpha_j y_j\langle x, x_j\rangle \\ \eta &= \frac{1}{2s}\left(\sum_{x\in S_+}\sum_{j=1}^{n}\alpha_j y_j\langle x, x_i\rangle - \sum_{x\in S_-}\sum_{j=1}^{n}\alpha_j y_j\langle x, x_i\rangle\right). \end{aligned} \tag{3.42}$$

Inserting the value for b into the decision function yields that only b is necessary to classify new points [**?**].

3.6 Summary

In this Chapter we first introduced a formal term for the decision boundary, the canonical hyperplane. This concept plays a central role in the field of Support Vector Machines as it states how classification of points is done. To this extent we discussed how the value of the margin, the distance from the hyperplane to its closest point(s), has to be included in order to form an optimization problem for binary Support Vector Machines.

Furthermore we stated the primal and the dual optimization problem for the hard margin binary Support Vector Machines for separable training sets which was then extended by introducing a kernel function in order to deal with the nonseparable case. This was a useful tool to map input vectors that are not linearly separable in the input space into a higher dimensional feature space where linear separation is possible.

Not only the inclusion of the kernel, but also the case of Soft Margin Hyperplanes was discussed, where we included a penalty term for wrongly classified training patterns. This helped us to relax the hyperplane constraint in order to put not the same weight on each single training point. Within this chapter we introduced two ways of including the slack variables into the objective function and the constraint for the hyperplane, the C-SVC and the ν-SVC. Both types aimed to do a linear separation for training points which are not linearly separable but the methods differed to some extent as the first required to a priori choose a penalty parameter which has to be found heuristically. The second method solved this problem by including a parameter which plays as an upper bound for the margin errors and also the number of Support Vectors and is an additional decision variable in the optimization problem.

4 Multi-Class Classification

So far we have looked at cases where there were two classes of objects to separate. Now we want to expand our knowledge of binary classification to multi-class classification.

First we will develop some procedures that allow us to work with the things we already observed in the binary case, using some number of binary Support Vector Machines in order to be able to separate more than two classes. This is done by decomposing the multiclass optimization problem into various smaller problems which can be done by various approaches.

But also procedures that treat only one single optimization problem will be developed in order to solve the classification problem globally.

4.1 Extending Binary Support Vector Machines to Multicategory Support Vector Machines

4.1.1 One Versus the Rest

As in the two-class classification, we start with n objects $x_1,\dots,x_n \in \mathscr{H}$, but now the objects are not assigned to two different classes but to k different classes $1,\dots,k$. Therefore we have the pairs $(x_1,y_1),\dots,(x_n,y_n)$ as before, but with more different values for y_i. For instance, for the digit recognition problem which will be discussed in Chapter **??** there are ten classes $y_1 = 0$, $y_2 = 1,\dots,y_{10} = 9$.

In order to separate these k classes with the One-Versus-the-Rest (OVR) method, we want to find k different classification functions $f_1,\dots,f_k$ with

$$f_g(x_i) = \langle w_g, x_i\rangle + b_g \quad g = 1,\dots,k \tag{4.1}$$

where each of them is trained to separate one class from the union of the rest of the classes.

To obtain these k different classification functions one has to solve k optimization problems which were developed for the binary Support Vector Machines in order to obtain the Lagrangian variables α_i and the bias b that are necessary to be able to evaluate the decision function.

Having found the k different classification function $f_1,\dots,f_k$, we will classify new points by assigning them to the group that has the highest so called confidence score for

$$\arg\max_{g=1,\dots k} f_g(x) = \arg\max_{g=1,\dots k} \sum_{i=1}^{n} y_i \alpha_i^g \langle x, x_i\rangle + b_g \tag{4.2}$$

where α_i^g are the Lagrange multipliers and b_g is the offset parameter for the gth optimization problem of the binary Support Vector Machine.

Unclassified points are then allocated to the class which has the highest confidence score on (**??**). In other words a point x is allocated to class g, if $f_g(z) > f_j(z)\ \forall j \neq g$ [**?**].

Remark. *Because we are talking about several binary classifiers, we can also use the kernel trick to map data that is not linearly separable in the original feature space into a higher dimensional feature space in order to be able to do a linear separation. Again we only need to replace the inner product in (**??**) by a kernel function that satisfies the Mercer Theorem.*

Also the Soft Margin Support Vector Machines can be used for the OVR approach for multi-category classification.

According to [**?**] the main drawback of this method is that it is a heuristic approach. Schoelkopf et al. describe it as a *winner-takes-all* approach because the binary classifiers that are used in (**??**) are obtained by training Support Vector Machines on different binary classification problems as there is always the decision one versus all other classes. Thus it could be the case that it is unclear whether their confidence scores, which are real valued numbers, are on comparable scales. This could be a problem if many classifiers assign the unclassified pattern to their class because only one can be chosen. Also the other way round, when no classifier would assign its class to the new pattern, one class always has to be chosen.

4.1.2 Pairwise Classification

Another way to set up a multi-class classification framework that works with multiple binary classification problems is by training a classifier for each possible pair of classes, classifying a point according to the number of votes for one class. This approach, first developed by Friedman [**?**], has become very popular because it is intuitive and easy to implement [**?**].

We are starting with the same setup as before: Given are training points that are members of group $1,\ldots,k$ but now we are comparing each training class to the others individually and not one versus the rest, ending up with $\frac{(k-1)k}{2}$ binary classifiers. For instance, in the digit recognition problem in Chapter **??** we have to find 45 different classifiers because in this case $k = 10$ [**?**], [**?**].

As before, the optimization problem for finding the optimal separating hyperplane is the same as in the binary case, because individually these problems are binary. Therefore we can use all

findings we had before to train the classifiers. Also using the kernel trick or soft margin hyperplanes works in the same way as we have seen before [**?**].

For a new test pattern, each classifier has to decide in which of the two classes the pattern fits better. Afterwards one counts the number of votes for each class, assigning the point to the class that got the most votes. Therefore this approach is often called "Max-Wins"[**?**], [**?**].

Although there are many more classifiers to train than in the OVR approach, each single optimization problem is smaller. Thus, for small k, i.e. for a small number of different classes, this approach has the advantage that the training of the SVM is easier than for the OVR method. On the other hand, for large k, the classifier that is obtained with the pairwise approach grows superlinearly in k and consequently will be slow to evaluate for this problems [**?**]. Besides this, the tendency of overfitting, happening due to the number of precisely trained classifiers, is another disadvantage of the proposed method [**?**], [**?**].

4.1.3 Decision Directed Acyclic Graph SVM

To overcome some shortcomings of the Pairwise Classification, Platt, Cristianini and Shawe-Taylor [**?**] introduced classification according to a Decision Directed acyclic Graph (DDAG). The training procedure of the DDAG is the same as for the pairwise approach, but the classification happens by going through a directed acyclic graph with $\frac{k(k-1)}{2}$ internal nodes and k leaves, where each node is equipped with one binary decision function [**?**].

Remark. *A directed acyclic graph is a graph whose edges have an orientation and no cycles. A rooted directed acyclic graph has a unique node such that it is the only node which has no arcs pointing into it. A rooted binary directed acyclic graph consists of nodes which have either* 0 *or* 2 *arcs leaving them [**?**].*

A new test pattern is classified via the following procedure: Beginning in the root node the binary decision function decides either to go to the left or to the right, depending on the output value which is either -1 or $+1$. During this procedure a so called evaluation path, beginning in the root node and ending in a leaf node, is developed. The end point of this path now indicates the predicted class [**?**], [**?**]. An example of a DDAG with $k = 4$ can be seen in Figure **??**.

In each step this procedure excludes one of the k classes, thus $k-1$ decisions have to be made to come to a classification decision.

Similar to the Pairwise approach, the training set of each classifier is smaller than in the OVR approach. Therefore its main advantage is that with a small number of classes the algorithm

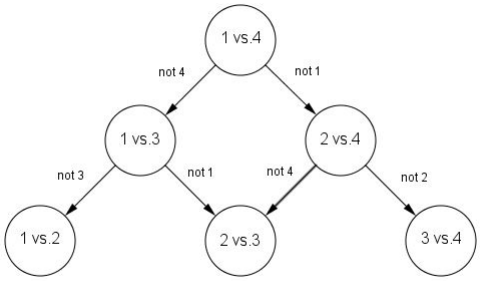


Figure 4.1: DDAG for a digit recognition problem with 4 digits.

needs less training time. The advantages of the DDAG compared to the Pairwise approach lies in the number of evaluations that have to be made in order to come to a solution. In the Pairwise approach all $\frac{k(k-1)}{2}$ classifiers have to be evaluated, whereas in the DDAG only $k-1$ classifiers have to be considered.

Remark. *According to Platt et al., the order in which the classifiers are evaluated throughout this evaluation path can be chosen arbitrarily, as re-orderings have not changed the outcomes in their experiments significantly. Therefore they followed an order starting with the first and the last classifier, working through the graph in numerical order [**?**].*

4.1.4 Error-Correcting Output Coding

Dietterich and Bakiri [**?**] suggested another method that is similar to the OVR approach. As we can train k binary Support Vector Machines for k classes, we can distinguish the training set in many other ways. For instance, we could split it into only two subsets, e.g. separating the even from the odd digits [**?**], or defining the classes according to the similarities the digits have, for instance vertical lines, closed curves [**?**] etc. in the Optical Character Recognition (OCR) case. The idea behind this method can be explained in a short example.

Example. For the OCR problem with digits 0 to 9, six binary classifiers f_{vl}, f_{hl}, f_{dl}, f_{cc}, f_{ol}, f_{or} are trained according to the similarities of the digits shown in Table **??**.

From this characteristics a unique codeword is established for each of the 10 digits as shown in Table **??** [16].

[16] Until now, we have labelled the binary classes by -1 and $+1$. For a better overview, we defined the two classes in this example to be 0 and 1. Otherwise we would not have a 6-bit string, as a sign also occupies a bit, but a 12-bit string which makes the codewords harder to read.

Column position	Abbreviation	Meaning
1	vl	contains vertical line
2	hl	contains horizontal line
3	dl	contains diagonal line
4	cc	contains closed curve
5	ol	contains curve open to left
6	or	contains curve open to right

Table 1: Digit characteristics [**?**]

	Code Word					
Class	vl	hl	dl	cc	ol	or
0	0	0	0	1	0	0
1	1	0	0	0	0	0
2	0	1	1	0	1	0
3	0	0	0	0	1	0
4	1	1	0	0	0	0
5	1	1	0	0	1	0
6	0	0	1	1	0	1
7	0	1	1	0	0	0
8	0	0	0	1	0	0
9	0	0	1	1	0	0

Table 2: Codewords for the digits 0 to 9 [**?**]

For the testing part, each of the 6 classifiers is evaluated for an unclassified digit, producing a sequence of numbers such as 110001. Afterwards this so called "6-bit string" is compared to the 10 codewords, assigning the test digit to the class with the nearest codeword according to the Hamming distance [17], i.e. the minimal loss. For a testing pattern which was assigned the code 110001 by the six binary Support Vector Machines, the codeword 110000 has the smallest Hamming distance, thus it is predicted to be the digit 4 [**?**].

This method is called the **Error Correcting Output Codes** (ECOC), as it can correct at least $\lfloor \frac{d-1}{2} \rfloor$ bit errors if the minimum Hamming distance is d. For instance for the OVR approach, Table **??** would have 10 rows and 10 columns, the characteristic would be the digit itself and the codeword would have a 1 in column $i+1$ and zeros elsewhere for digit i.

Allwein et al. [**?**] extended the work of Dietterich and Bakiri to the extent that they replaced the Hamming distance by a measure that takes margins into account [**?**]. For classifying a pat-

[17] The Hamming distance counts the number of bits that differ [**?**].

tern, they introduced a margin based loss function that also takes the magnitude of the loss into account and which does not only calculate the number of different bits in the output string. They called this approach loss-based decoding [**?**].

In this framework the output code for the training data is given a priori. Crammer and Singer [**?**] later developed methods to design good output codes for discrete and continuous codes using Support Vector Machines.

4.2 Global Support Vector Machines

Although the expansion of binary Support Vector Machines to multicategory Support Vector Machines is a popular way of dealing with more than two classes in pattern recognition, there are more direct ways of formulating optimization problems for multiclass classification. In the following section we will discuss various approaches.

4.2.1 J. Weston and C. Watkins

The method that was proposed by Weston and Watkins [**?**] is similar to the OVR approach but the hyperplanes are obtained by solving only one and not k optimization problems [**?**]. They construct a piecewise linear separation obtained by the following optimization problem by generalizing the binary optimization problem:

Model 7 (Primal optimization problem for multiclass SVM: Weston and Watkins)**.**

$$\begin{aligned} \min_{w,\xi,b} \quad & \frac{1}{2}\sum_{m=1}^{k}||w_m||^2 + C\sum_{i=1}^{n}\sum_{m\neq y_i}\xi_i^m \\ \text{subject to} \quad & \langle w_{y_i}, x_i\rangle + b_{y_i} \geq \langle w_m, x_i\rangle + b_m + 2 - \xi_i^m \\ & \xi_i^m \geq 0, \quad i = 1,\ldots,n, \quad m \in \{1,\ldots,k\}\backslash\{y_i\} \end{aligned} \tag{4.3}$$

The resulting classification rule is then evaluated at the decision function

$$f(x) = \arg\max_l [\langle w_l, x\rangle + b_l], \;\; l = 1,\ldots,k. \tag{4.4}$$

Remark. *As pointed out in [**?**], for $k = 2$ this would be equal to the optimization problem in Model **??** if we take $w_1 = -w_2$, $b_1 = -b_2$ and $\xi_i = \frac{1}{2}\xi_i^1$ for pattern i in class 1, and $\xi_i = \frac{1}{2}\xi_i^2$ for pattern i in class 2.*

To get to the dual formulation of Model **??** we proceed as in the binary case by computing the Lagrangian

$$
\begin{aligned}
L(w,b,\xi,\alpha,\beta) = \frac{1}{2}\sum_{m=1}^{k}||w_m||^2 + C\sum_{i=1}^{n}\sum_{m\le y_i}\xi_i^m \\
-\sum_{i=1}^{n}\sum_{m=1}^{k}\alpha_i^m[\langle w_{y_i}-w_m,x_i\rangle + b_{y_i}-b_m-2+\xi_i^m \\
-\sum_{i=1}^{n}\sum_{m=1}^{k}\beta_i^m\xi_i^m
\end{aligned}
\tag{4.5}
$$

with dummy variables $\alpha_i^{y_i}=0$, $\xi_i^{y_i}=2$ and $\beta_i^{y_i}=0$ for $i=1,\ldots,n$, the constraints for the Lagrangian variables $\alpha_i^m \ge 0$, $\beta_i^m \ge 0$, and the slack variables $\xi_i^m \ge 0$, where $i=1,\ldots,n$, and $m \in \{1,\ldots,k\}\backslash\{y_i\}$. Equation (**??**) has to be maximized with respect to the Lagrangian variables and minimized with respect to w and ξ in order to obtain the coefficients for later evaluating the decision function (**??**).

After multiple manipulations [**?**], the dual problem is formulated as follows.

Model 8 (Dual optimization problem for multiclass SVM: Weston and Watkins)**.**

$$
\begin{aligned}
\max \quad & W(\alpha) = 2\sum_{i,m}\alpha_i^m + \sum_{i,j,m}[-\frac{1}{2}c_j^{y_i}A_iA_j + \alpha_i^m\alpha_j^{y_i} - \frac{1}{2}\alpha_i^m\alpha_j^m]\langle x_i,x_j\rangle \\
\text{subjec to} \quad & \sum_{i=1}^{n}\alpha_i^l = \sum_{i=1}^{n}c_i^lA_i, \;\; l=1,\ldots,k \\
& 0 \le \alpha_i^m \le C, \quad \alpha_i^{y_i}=0, \quad i=1,\ldots,n, \quad m\in\{1,\ldots,k\}\backslash y_i
\end{aligned}
\tag{4.6}
$$

This is a quadratic function in terms of α with linear constraints, where A_i and c_i are short notation for

$$
A_i = \sum_{m=1}^{n}\alpha_i^m \text{ and } c_i^l = \begin{cases} 1 & \text{if } y_i = l \\ 0 & \text{if } y_i \ne l \end{cases}.
\tag{4.7}
$$

New points are then assigned to a group according to the following decision function

$$
f(x) = \arg\max_l[\sum_{i=1}^{n}(c_i^lA_i - \alpha_i^l)\langle x_i,x\rangle + b_l],
\tag{4.8}
$$

where b_l, $l = 1,\ldots,k$, can be obtained by simultaneously solving a set of equations from the KKT-conditions. As in the binary case, the inner product can be replaced by any kernel function that satisfies the Mercer Theorem [**?**].

4.2.2 Guermeur's Generalization of Vapnik, Bredensteiner, Bennett, Weston and Watkins Support Vector Machine

Also Vapnik [**?**] and Bredensteiner and Bennett [**?**] proposed similar methods for the direct approach in k-classification problems, where in general the main differences lie in the definition of the objective functions [18]. Guermeur [**?**] showed that these three variants are essentially equivalent and that the resulting multicategory Support Vector Machines can be trained by solving the following optimization problem [**?**].

Model 9 (Primal optimization problem for multiclass SVM: Guermeur)**.**

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\sum_{1\le m<l\le k} ||w_m - w_l||^2 + C\sum_{i=1}^{n}\sum_{m=1}^{k}\xi_i^m \\ \text{subject to} \quad & \langle w_{y_i} - w_m, x_i\rangle + b_{y_i} - b_m \ge 1 - \xi_i^m, \\ & \xi_i^m \ge 0, \quad i = 1,\ldots,n, \quad m \in \{1,\ldots,k\}\backslash\{y_i\} \end{aligned} \tag{4.9}$$

Uniqueness is guaranteed by adding the constraints $\sum_{m=1}^{k} w_m = 0 \in \mathbb{R}^d$ assuming $x_i \in \mathbb{R}^d$ and $\sum_{m=1}^{k} b_m = 0$. A new pattern is again assigned to the group g with the highest score on

$$f_g(x) = \langle w_g, x\rangle + b_g. \tag{4.10}$$

By changing the metric in the objective function, also other models, which may be more appropriate for the domain in which the data lives, can be formulated [**?**], [**?**].

4.2.3 Crammer and Singer

After picking up the idea of the Error Correcting Output Codes of Dietterich and Bakiri [**?**] and the developments made by Allwein et al. [**?**], Crammer and Singer also proposed a method for multicategory Support Vector Machines [**?**].

On the whole they had a similar approach as Weston and Watkins, but one main difference is that they did not include a bias term into the decision function.

[18] The objective functions can be found in [**?**] S. 173, Table 1.

Therefore we focus on the following set classifiers:

$$H_W(x) = \arg\max_{m=1}^{k} \langle w_m, x \rangle. \tag{4.11}$$

As in many methods before, the predicted label is the group m that attains the highest value on (**??**) amongst all $m = 1, \ldots, k$, which is again called the similarity score.

Example. Applying this to the binary case x would have the label 1 if $\langle w, x \rangle > 0$ and the label -1 otherwise. This could be implemented by defining $w_1 = w$ and $w_2 = -w$ [**?**].

Another difference to the method of Weston and Watkins is that Crammer and Singer have a smaller set of inequality constraints in the optimization problem as they do not compare the similarity score of the predicted class $\langle w_{y_i}, x_i \rangle$ of training pattern (x_i, y_i) to all other scores $\langle w_m, x_i \rangle$, where $m = \{1, \ldots, k\} \backslash \{y_i\}$, but only to the maximum within all confidences.

Therefore we end up with a piecewise linear upper bound on the empirical error

$$R_{emp} \leq \frac{1}{n} \sum_{i=1}^{n} \max_m \left[\langle w_m, x_i \rangle + 1 - \delta_{y_i,m} \right] - \langle w_{y_i}, x_i \rangle, \tag{4.12}$$

where $\delta_{p,q}$ is equal to 1 if $p = q$ and 0 otherwise. For each of the summands it holds that they are 0 if the confidence level for the correct label is larger by at least 1 than the confidences for the rest of the labels [**?**].

If the empirical loss is equal to zero, i.e.

$$\max_m \left[\langle w_m, x_i \rangle + 1 - \delta_{y_i,m} \right] - \langle w_{y_i}, x_i \rangle = 0, \quad \forall i = 1, \ldots, n, \tag{4.13}$$

we say that the training sample $(x_1, y_1), \ldots, (x_n, y_n)$ is linearly separable by a multiclass machine [**?**].

As shown before, the generalization ability of a binary and also multiclass Support Vector Machine depends on the margin, consequently also Crammer and Singer are minimizing the norms of $w_1, \ldots, w_k$, thus end up with the following minimization problem:

Model 10 (Primal optimization problem for multiclass SVM: Crammer and Singer)**.**

$$\begin{aligned} \min_{w,\xi} \quad & \frac{1}{2}\beta \sum_{m=1}^{k} ||w_m||^2 + \sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & \langle w_{y_i}, x_i \rangle + \delta_{y_i,m} - \langle w_m, x_i \rangle \geq 1 - \xi_i \end{aligned} \tag{4.14}$$

where the slack variables ξ_i are defined to be

$$\xi_i = \max_m \left[\langle w_m, x_i\rangle + 1 - \delta_{y_i,m}\right] - \langle w_{y_i}, x_i\rangle \quad \forall i = 1,\dots n \tag{4.15}$$

and $\beta > 0$ is a regulization constant.

Solving this problem we proceed as in the binary case. First we compute the Lagrangian L with positive Lagrangian multipliers $\alpha_{i,m} \geq 0$

$$L(w,\xi,\alpha) = \frac{1}{2}\beta \sum_m ||w_m||^2 + \sum_{i=1}^{n} \xi_i + \sum_{i,m} \alpha_i^m \left[\langle w_m, x_i\rangle - \langle w_{y_i}, x_i\rangle - \delta_{y_i,m} + 1 - \xi_i\right]. \tag{4.16}$$

Setting its derivatives with respect to x_i and w_m equal to zero yields

$$\sum_{m=1}^{k} \alpha_i^m = 1 \quad \text{and} \quad w_m = \frac{1}{\beta}\sum_i (\delta_{y_i,m} - \alpha_i^m) x_i. \tag{4.17}$$

In this case the contribution of a pattern to (**??**) is $\delta_{y_i,m} - \alpha_i^m$, therefore a vector x_i is called Support Vector if there exists a w_m for $m = 1,\dots,k$, for which this coefficient does not equal zero.

Substituting (**??**) into the Lagrangian yields the dual optimization problem [**?**]

Model 11 (Dual optimization problem for multiclass SVM: Crammer and Singer)**.**

$$\begin{aligned} \max_{\alpha} \quad & W(\alpha) = -\frac{1}{2\beta}\sum_{i,j}\langle x_i, x_j\rangle \sum_{m=1}^{k} (\delta_{y_i,m} - \alpha_i^m)(\delta_{y_j,m} - \alpha_j^m) - \sum_{i,m} \alpha_i^m \delta_{y_i,m} \\ \text{subject to} \quad & \sum_{m=1}^{k} \alpha_i^m \geq 0 \quad \forall i = 1,\dots,n \\ & \sum_{m,i} \alpha_i^m = 1. \end{aligned} \tag{4.18}$$

A new test pattern x is classified according to

$$H(x) = \arg\max_{m=1}^{k} \left[\sum_i (\delta_{y_i,m} - \alpha_i^m)\langle x_i, x\rangle\right]. \tag{4.19}$$

Remark. *Since we are working with inner products again, also in this case they can be replaced by any Kernel function that satisfy the Mercer Theorem [**?**].*

In their paper Crammer and Singer also introduce a method how to decompose the quadratic program given by Model **??** in order to get a simple and memory efficient algorithm for solving the quadratic optimization problem [**?**].

4.2.4 Yang, Shao and Zhang: Multiple Birth SVM

Yang, Shao and Zhang [**?**] found another way of doing multiclass classification with an idea that is rather dissimilar to the ones we have seen before. They also construct a classifier that solves k quadratic problems simultaneously but their approach is different to the others. For their so called multiple birth support vector machine (MBSVM) the idea can be explained by the following toy 3-class classification problem in $\mathbb{R}^2$, shown in Figure **??**.

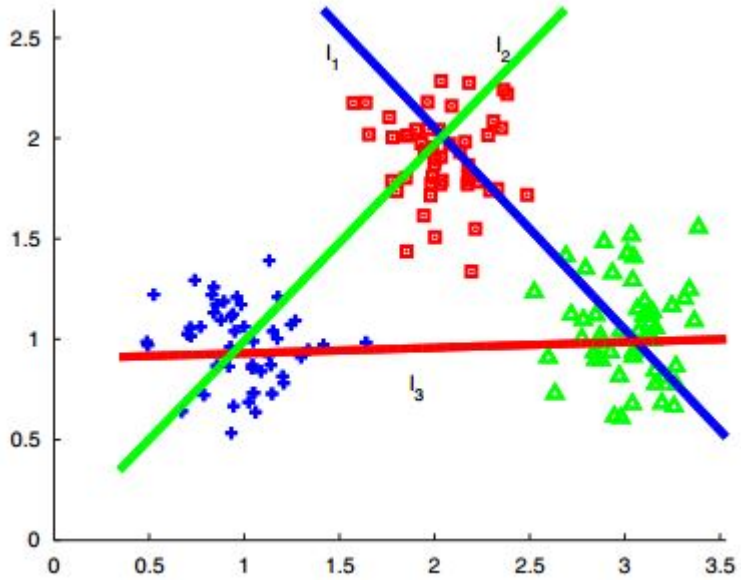


Figure 4.2: A toy example learned by the linear MBSVM [**?**]

Example. The classifier finds a decomposition in three lines l_1, l_2 and l_3 corresponding to the points whose class-membership is shown with the different shapes in such a way that the points marked as crosses are at maximum distance to the line l_1 while the points belonging to the other two groups are proximal to the line l_1. In the same way the other two lines are found. A new pattern is then assigned to the class whose line it lies farthest to [**?**].

The resulting optimization problem is the following.

Model 12 (Primal optimization problem for multiclass SVM: Z.Yang et al.'s Multiple Birth SVM)**.**

$$\begin{aligned} \min_{w_j,\, b_j,\, \xi_j} \quad & \frac{1}{2}||B_j w_j + e_{j1} b_j||^2 + C_k e'_{j2} \xi_j \\ \text{subject to} \quad & (A_j w_j + e_{j2} b_j) + \xi_j \geq e_{j2} \\ & \xi_j \geq 0, \end{aligned} \tag{4.20}$$

where A_j is the matrix of all points belonging to class j, $B_j = [A'_1, \ldots, A'_{j-1}, A'_{j+1}, \ldots, A'_k]'$ is the matrix of all patterns that do not belong to class j, e_{j1} is the vector of ones with $n - n_k$ entries, where n_j is the number of training points belonging to class j, $n = \sum_{i=1}^{j} n_i$, e_{j2} is the vector of ones with n_j entries, ξ_j are the slack variables and $C_j > 0$ the penalty parameter as in Chapter **??**.

The main objective therefore is to minimize the sum of squared distances from the hyperplane to the points of the $k-1$ classes that do not belong to class k, plus a term that represents the sum of error variables.

A new pattern is then assigned to the class [**?**] that maximizes

$$f(x) = \arg\max_{j=1,\ldots,k} \frac{|\langle w_j, x \rangle + b_j|}{||w_j||}. \tag{4.21}$$

Like in the other cases, we obtain the solution to this model by first computing the Lagrangian with Lagrangian variables α_i and β_i

$$\begin{aligned} L(w_j, b_j, \xi_j, \alpha_j, \beta_j) = & \frac{1}{2}||B_j w_j + e_{j1} b_j||^2 + c_j e'_{j2} \xi_j \\ & - \alpha'_j((A_j w_j + e_{j2} b_j) + \xi_j - e_{j2}) - \beta'_j \xi_k. \end{aligned} \tag{4.22}$$

It is then maximized with respect to w and b and minimized with respect to ξ, α and β in order to obtain the dual problem of the Multiple Birth Support Vector Machine:

Model 13 (Dual optimization Problem for multiclass SVM: Z. Yang et al's Multiple birth SVM)**.**

$$\begin{aligned} \max_{x_j} \quad & e'_{j2} \alpha_j - \frac{1}{2} \alpha_j G_j (H'_j H_j + \varepsilon I)^{-1} G'_j \alpha_j \\ \text{s.t.} \quad & 0 \leq \alpha_j \leq C_j \end{aligned} \tag{4.23}$$

where $H_j = [B_j e_{j1}]$, $G_j = [A_j e_{j2}]$, $C_j > 0$. The term εI with ε a small scalar and I the identity matrix ensures that the inverted matrix is not ill-conditioned. [**?**].

Extending the linear case by regarding rather kernel-generated surfaces $k(x,E')u_j + b_j = 0$ for $j = 1,\ldots,k$ and $E' = (A'_1,\ldots,A'_k)$ than hyperplanes allows again to map the data that is non-separable in the original feature space into a higher dimensional feature space where linear separation is possible, see Chapter **??** [**?**].

4.3 Summary

Extending the binary Support Vector Machines to k-class Support Vector Machines yields many practical applications, e.g. optical character recognition, and can be done in various ways of which we introduced eight approaches.

The first four applied multiple binary Support Vector Machines in order to decompose the multicategory labelling decision into further smaller problems. To this extent we have seen different methods of using binary classifiers for multiclass problems as the One Versus the Rest approach, where we constructed k binary Support Vector Machines, one for each class, and the Pairwise Classification where we had $\frac{k(k-1)}{2}$ smaller binary problems, comparing each pair of classes. Both methods assign a new pattern according to the number of votes one class is achieving on the decision function. Like in the pairwise approach, in the Decision Directed Acyclic Graph method we trained $\frac{k(k-1)}{2}$ binary Support Vector Machines, but the classification was done according to a path through a directed, acyclic graph, resulting in only $k-1$ and not $\frac{k(k-1)}{2}$ classifying decisions. Finally, the idea of the Error Correcting Output Coding was to train a certain number of binary classifiers where each produces a bit for a codeword assigned to a pattern. For the classification decision the class for which the code is the most similar to the produced codeword is chosen.

Solving multiple binary problems is an easy way to construct multiclass Support Vector Machines, but there are also methods where we only have to solve one optimization problem to do a multiclass classification. To this extent we discussed four ways of constructing such optimization problems. The one by Weston and Watkins [**?**] was already developed in the 1990s, and later be proven by Guermeur [**?**] to be similar to the one promoted by Vapnik [**?**] and Bredensteiner and Bennett [**?**]. Crammer and Singer [**?**] thought these models to be rather lager and complex, therefore they proposed another approach with a smaller set of constraints and also an efficient algorithm that solves their optimization problem. At last, Yang, Shao and Zhang [**?**] proposed a way of doing the classification decision by not looking at the optimal hyperplanes, but by looking

for the hyperplane that is farthest away from the one class while being near all other classes.

All of them, both the multiple binary Support Vector Machines as the global multiclass Support Vector Machines, have advantages and disadvantages that are mainly driven by the number of quadratic programs that have to be solved, the number of variables and the number of constraints [19]. Which method is taken for solving certain multiclass pattern recognition problems depends on the number and the nature of the feature patterns.

[19] A table with these instances can be found in [**?**] p. 158, Table 1.

5 Implementation in Matlab

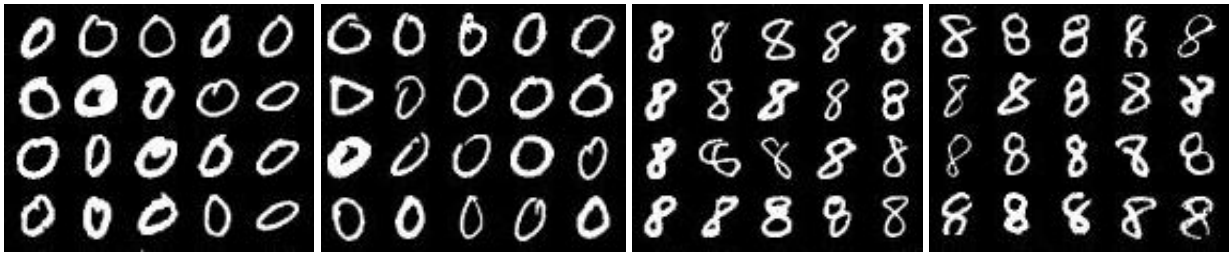
Having studied the theoretical part of binary and multicategory Support Vector Machines, we now want to apply this theory to real world data. A popular pattern recognition problem analysed many times to test the accuracy and generalization ability of Support Vector Machines is the handwritten digit recognition problem. First we will discuss the structure of the data which is then used in the implementation and evaluation of the binary and some of the multicategory Support Vector Machines that were introduced in Chapter **??** and Chapter **??**. For the programming part the software Matlab is used.

5.1 Handwritten Digit Data

From the various databases for handwritten digits we will do our research on the MNIST-database [**?**] [20] which is constructed as follows.

For each digit from 0 to 9 there are two sets of data available, a training set and a test set. Each of these sets contains a certain amount of handwritten digits where each of these digits is a 28×28-pixel image. Therefore the data can be stored as a 28×28-dimensional matrix that has entries between 0 and 255 where 0 means that the pixel is black, 255 means that the pixel is white and the pixels with numbers in between are shades of grey.

Extracts of examples for the test and the training data respectively are shown in Figure **??**



(a) Training data 0 (b) Test data 0 (c) Training data 8 (d) Test data 8

Figure 5.1: Examples of Training and Test Data from MNIST-database

For the purpose of using the pictures in order to test the implemented Support Vector Machines, the pictures first have to be translated into matrices and the matrices have to be reshaped into vectors. This work could be done by oneself but for convenience we will work with the data from [**?**] where one can import the MNIST-database directly into Matlab in the desired shape.

[20] For more information about the MNIST database see http://yann.lecun.com/exdb/mnist [**?**].

5.2 Binary Support Vector Machine

For the binary classification problem we will train Support Vector Machines with different settings on 1000 training patterns and test it on 300 testing patterns for each group. We will evaluate the findings for the digits 0 and 8 regarding the performance of the Support Vector Machines measured on the percentage of errors. Furthermore we will analyse the influence of different kernels as well as various settings for the soft margin parameter C. We will also look at the number of Support Vectors that are used for the different kinds of Support Vector Machines in some cases to get a notion which settings need the most Support Vectors, as their number acts as an indicator for the generalization ability of the Support Vector Machine [21].

First of all we will do this analysis training and testing the Support Vector Machines with functions provided by Matlab, but we will also compare the results to a self-implemented Support Vector Machine.

5.2.1 Binary Support Vector Machine with Functions Provided by Matlab: `binSVM`

Matlab provides the functions `svmtrain` and `svmclassify` [**?**] for training and testing Support Vector Machines. The two functions will now be evaluated for the different kinds of settings that were mentioned above. The default settings of `svmtrain` are shown in Table **??**.

Optimization Method	SMO
Kernel function	linear
Soft Margin parameter	$C = 1$

Table 3: Default settings for `svmtrain`

Doing this analysis aims to find the optimal settings to train a Support Vector Machine that has small error ratios and also a good generalization ability.

Different Optimization Methods

First of all one can vary the optimization method for the training part of the Support Vector Machine, but as the Sequential Minimal Optimization (SMO) method by John C. Platt was particularly developed to solve optimization problems regarding Support Vector Machines [**?**] this one turns out to return the lowest error ratios in comparison to Least Square (LS) and Quadratic Programming (QP) as one can see in Table **??**.

[21] See Leave-One-Out-Bound, Chapter **??**.

	SMO	LS	QP
Support Vectors	117	2000	713
Error Ratio 0	2.00	1.00	2.00
Error Ratio 8	2.33	4.00	2.33
Average	2.17	2.50	2.17

Table 4: Errors in % for different optimization methods

The first thing to mention is that the method LS achieves the highest error ratio and also results in the highest number of Support Vectors that is possible. This means that in the classification part, each training pattern is used for evaluating the decision function. The methods SMO and QP score the same error ratio, but QP is not only slower but also classifies the data with 713 Support Vectors and not only 117 as the method SMO does. Regarding these findings we will use the SMO method unless stated otherwise.

Different Kernels

As mentioned in Chapter **??**, one cannot only use the dot product as a similarity measure but various different kernel functions to map non-separable input data into a higher dimesional space where a linear separation may be possible. In Table **??** one can see the error ratios for the four different kernel functions we have already introduced in Chapter **??**.

	linear	quadratic	polynom. $d = 3$	rbf
Support Vectors	117	621	10	2000
Error Ratio 8	2.00	2.00	28.00	1
Error Ratio 0	2.33	2.666	1.67	0
Average	2.17	2.33	14.83	50.00

Table 5: Errors in % for different kernels

For the chosen feature data it turns out that the lowest error ratios are achieved with the default option linear which corresponds to the dot product as kernel. But the unusual high error ratio for the RBF kernel needs further investigation.

Using the RBF kernel we will now analyse the resulting error ratios using different values for the parameter σ as shown in Figure **??**.

For values of σ from 1 to 100 we get the best error ratio (1.5%) for $\sigma = 36$ but also for smaller σ, for instance for $\sigma = 18$, we get a error ratio of 1.83%.

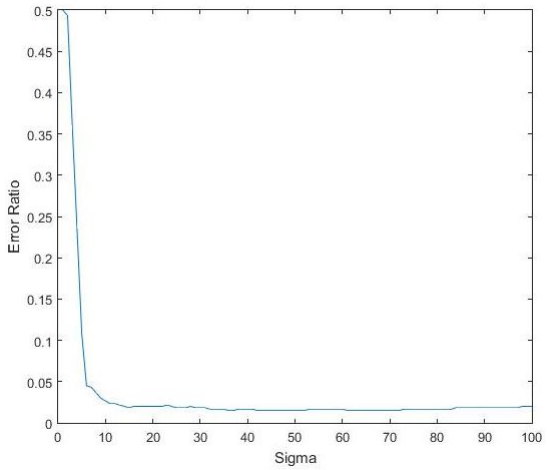


Figure 5.2: Error ratios for different values of σ

Settings for the Soft Margin: C

In this section we will now vary the parameter C in the soft margin optimization in Model **??**. As one can see in Table **??**, for small C we get slightly different error ratios, where analysis has shown that increasing C to more than the default value of 1 does not provide any further improvement on the error ratio.

Kernel	**C**	**Error Ratio 0**	**Error Ratio 8**	Average
Linear	0.005	0.33	2.00	1.17
	0.1	1.00	2.33	2.167
Quadratic	1	2.00	2.267	2.33
Polynomial	1	2.80	1.67	1.48
RBF	0.074	4.00	1.33	2.667
	12	1.00	1.67	1.33

Table 6: Errors in % for different kernels and different values of C

As can be seen in Table **??**, different values of the soft margin parameter C can influence the error ratio quite much. Though, for the quadratic and polynomial kernel, analysis has shown that it does not matter which value for C is taken.

Best Settings for Binary Support Vector Machines

Regarding the different settings that have been analysed, the best one for training the Support Vector Machine for digits 0 and 8 are the optimization method SMO, a linear kernel and a soft

margin parameter $C = 0.005$. Though, in the literature the best results were achieved with an RBF kernel in most instances.

For the purpose of this thesis we will not make any further investigation to improve the results that were presented but will work with the settings that were found out to fit the best for our data.

In Figure **??** we can see which test patterns were not classified correctly under these settings.

Figure 5.3: Patterns that were not categorized correctly

5.2.2 Own Binary Support Vector Machine: `ownbinSVM` and `ownbinSVMslack`

Using the functions `svmtrain` and `svmclassify` one can easily implement Support Vector Machines and play with their settings, but as the training of a Support Vector Machines is just a quadratic convex optimization problem one can implement it on ones own without too much effort. This was done for a Support Vector Machine without and with a soft margin term and the results are shown in Table **??**.

	$C = 0$	$C = 0.005$	$C = 1$
Error Ratio 0	0.67	0.67	0.67
Error Ratio 8	4.33	8.67	3.33
Average	2.50	4.67	2.00

Table 7: Errors in % for own implemented Support Vector Machine

Also evaluating different kernel functions on this own implemented Support Vector Machine could be done by replacing the inner product calculations by the respective kernel function. But as it turns out in the tests with the Matlab function `svmtrain`, the linear kernel with a soft margin constraint of $C = 0.005$ provides the lowest error ratio. Though, the results for the own implemented Support Vector Machine with the soft margin constraint $C = 1$ are better than the ones for $C = 0.005$ contrary to former findings. One reason could be the used optimization method, as `svmtrain` works with SMO whereas in the own implementation QP was used.

5.3 Multicategory Support Vector Machine

We now want to extend the knowledge of binary Support Vector Machines to classify more than two digits. For this purpose we will analyse three different methods explained in Chapter **??**: One vs. the Rest (OVR), the Pairwise which is also called One vs. One (OVO) and the Decision Directed Acyclic Graph (DDAG) on the MNIST database.

5.3.1 One vs. the Rest: `M_ovrSVM`

For training a Multicategory Support Vector Machine according to the OVR method, each class is once treated as the main group with labelling $y_i = 1$ and the rest as a group with label $y_i = -1$. Therefore k binary classifiers are trained on the training set. A new test pattern is then assigned to the class which has the highest confidence score according to (**??**).

The most apparent drawback of this method is that it has to deal with much more feature data than the other two methods because we are regarding training samples of *all* digits at the same time. Consequently one has to reduce the number of training pattern from 1000 we had for the binary Support Vector Machine to a smaller number, in our case 300, otherwise the function `svmtrain` will not converge unless one allows many KKT conditions to be violated. The huge number of training data leads to the problem that classifiers are trained on way more "negative" samples than positive ones, leading to a really high error ratio of an overall of 20.97%. The confusion matrix can be seen in Table **??**.

		true class									
		0	1	2	3	4	5	6	7	8	9
predicted class	0	90.33	0	2.67	1.00	0.67	3.00	2.33	0.67	1.33	1.00
	1	0	94.33	3.00	0.67	0.33	0	1.33	2.33	2.67	1.00
	2	1.67	0.67	79.67	3.67	1.33	1.00	4.67	5.67	3.00	2.33
	3	0.33	1.00	2.33	75.67	1.00	7.33	0	1.67	6.33	4.33
	4	0	0.33	1.00	0.33	83.00	2.67	2.67	0.67	4.67	8.00
	5	3.67	2.00	2.00	10.00	0	71.33	3.33	1.67	7.00	3.00
	6	2.00	0.33	1.33	0.67	2.00	2.33	81.67	0.33	2.00	0
	7	0.33	0	2.33	4.00	1.33	1.67	1.67	79.67	2.00	7.00
	8	1.33	1.33	4.67	2.00	1.67	9.00	1.33	0.33	3.67	2.33
	9	0.33	0	1.00	2.00	8.67	1.67	1.00	7.00	7.33	71.00
Error %		9.67	5.67	20.33	24.33	17.00	28.67	18.33	20.33	36.33	29.00

Table 8: Selected classes in % for the OVR Support Vector Machine

In the case of OVR multicategory Support Vector Machines, different parameter values for the soft margin appear to have no influence on the error ratios.

One idea to get better results was to try to enlarge the "positive" training group of each classifier and reduce the number of patterns of the other group such that each of the two groups have the same size as it was the case in the binary Support Vector Machine. Unfortunately also this attempt resulted in very bad error ratios.

5.3.2 Pairwise Classification: `M_PairSVM`

For the OVO approach we trained 45 classifiers, each separating one class from each other class. A new test pattern is assigned to the class which was selected from the most of the 45 classifiers. Applying this method to train a multicategory Support Vector Machine for the digit recognition problem has the advantage that one can train it with much more feature data because the single optimization problems are smaller than in the OVR case. Therefore we get a smaller error ratio of 9.37%. The confusion matrix is shown in Table **??**.

		true class									
		1	2	3	4	5	6	7	8	9	0
predicted class	1	98.33	0.33	0	0	0	1.00	1.00	1.33	1.33	0
	2	0.33	90.00	1.00	1.67	0.33	2.33	2.67	2.00	0.33	1.67
	3	0	0.33	91.00	0	4.33	0	1.67	3.00	2.00	0
	4	0.33	0.33	0.67	91.33	0.67	1.33	2.00	1.67	3.33	0
	5	0	0.33	3.00	0	87.67	2.00	0	4.33	0.67	0.67
	6	1.00	1.33	0.33	1.67	1.33	90.67	0	0.33	0	1.00
	7	0	1.00	1.67	0.33	0	0	88.00	1.33	2.33	0
	8	0	4.00	2.00	0.33	3.33	1.33	0.33	84.00	0.33	0
	9	0	0.67	0.33	4.67	1.00	0	4.00	1.00	88.67	0
	0	0	1.67	0	0	1.33	1.33	0.33	1.00	1.00	96.67
Error %		1.67	10.00	9.00	8.67	12.33	9.33	12.00	16.00	11.33	3.33

Table 9: Selected classes in % for the OVO Support Vector Machine

Using the OVO method the value C for the soft margin influences the results. As in the binary case we get better results when changing the value for C from the default setting $C = 1$ to the value that achieved the best results in the binary case $C = 0.005$ as can be seen in Table **??**.

5.3.3 Decision Directed Acyclic Graph: `M_DDAGSVM`

In the DDAG method we also trained 45 classifiers like in the OVO Support Vector Machine, but instead of evaluating each of these classifiers for categorizing a test pattern, we make decisions along a decision tree as shown in Figure **??**. While this method takes the same training time as the OVO method, the testing part is faster. Already with the first binary decision we rule out 8

Digit	Error %	
	$C = 1$	$C = 0.005$
1	1.67	1.67
2	11.67	10.00
3	12.33	9.00
4	8.00	8.67
5	18.00	12.33
6	10.33	9.33
7	12.33	12.00
8	17.33	16.00
9	15.33	11.33
0	4.33	3.33
Average	11.13	9.37

Table 10: Error in % for OVO with different Soft Margin Constraints

and with each step $i = 1,\ldots,9$ further $9-i$ classifiers. This results in a faster classification step and also the overall error ratio is smaller due to the design of the algorithm.

Different to the binary Support Vector Machine from the beginning of this chapter, for the DDAG Support Vector Machine we achieve the lowest error ratio with an RBF kernel with $\sigma = 36$ and $C = 12$. The confusion matrix with these settings can be seen in Table **??**. Comparing the error ratios for these settings (Table **??**) encourage the different choice of the kernel for this method.

		true class									
		1	2	3	4	5	6	7	8	9	0
predicted class	1	98.33	0.33	0	0	0	1.00	0.67	0.67	1.00	0
	2	0.33	92.67	0.33	1.33	1.00	1.67	2.00	2.00	0	0.33
	3	0	0.67	92.00	0.33	3.00	0	1.33	2.33	0.67	0
	4	0.33	0.33	0.67	91.67	0	0.33	2.00	0.33	2.33	0
	5	0	0.33	2.33	1.00	90.00	3.00	1.00	3.00	1.00	1.33
	6	1.00	1.00	0.33	1.67	1.33	92.00	0	0.33	0	1.33
	7	0	1.00	2.33	0.33	0	0	90.33	0.67	3.00	0
	8	0	1.33	1.67	0.67	3.00	0.67	0.33	89.00	1.33	0
	9	0	1.33	0.33	3.00	1.00	0	2.33	0.67	90.00	0.33
	0	0	1.00	0	0	0.67	1.33	0	1.00	0.67	96.67
Error %		1.67	7.33	8.00	8.33	10.00	8.00	9.67	11.00	10.00	3.33

Table 11: Selected classes in % for DDAG Support Vector Machine

	Error Ratio			
	Linear Kernel		RBF Kernel	
Digit	$C = 1$	$C = 0.005$	$C = 1$, $\sigma = 36$	$C = 12$, $\sigma = 36$
1	2.33	2.00	2.00	1.67
2	14.67	12.67	10.00	7.33
3	12.33	9.33	10.33	8.00
4	8.67	8.33	9.00	8.33
5	18.00	13.67	11.00	10.00
6	8.67	9.33	7.33	8.00
7	12.00	11.33	11.67	9.67
8	16.33	15.00	14.33	11.00
9	15.67	11.33	10.67	10.00
0	5.00	3.67	3.33	3.33
Average	11.37	9.67	8.97	7.73

Table 12: Different Errors in % for DDAG with linear and RBF kernel

5.4 Summary

In this section we discussed various implementations and settings for binary and multicategory Support Vector Machines. To this extent we found out that certain settings, as the chosen optimization algorithm, the used kernel or the values for the parameters of the soft margin and some kernel-based values can influence the results in many ways.

Not only the error ratios but also the generalization ability, i.e. the number of Support Vectors are the properties of good and less good pattern recognition algorithms. For the binary investigation, the properties of the constructed Support Vector Machines led to the result that, for the chosen feature and training data, a linear kernel and a rather small soft margin constraint is optimal.

For the multicategory case we conclude that the OVR approach is the worst and the DDAG is the best reviewed method for having small error ratios.

6 Perspectives

In order to improve the accuracy and the generalization ability of the used Support Vector Machines one could do some work on the following instances:

The MNIST-database provides about 60000 training and test points from which we only used a small part. Preparing the data before training the Support Vector Machines, e.g. via a Principal Component Analysis [**?**], could result in smaller error ratios and better generalization ability as this procedure helps to find a subset of the training set that describes the whole training patterns to that amount that training a Support Vector Machine regarding this subset results in almost the same findings as it would have been trained on the whole dataset [**?**], [**?**].

Also other preparation methods as clustering algorithms like the k-means algorithm [**?**] that prepares the data by forming a certain number of clusters where each of the clusters provides an "average" point, which are then used for the training part of a Support Vector Machine could be useful in order to reduce the dimensionality for large datasets.

There has also gone some effort in feature selection that minimizes the bounds of the leave-one-out error which could be used as another method to preprocess the used training data [**?**].

As done partly in the implementation, the used kernels and other settings for the Support Vector Machine can be investigated further. While implementing the algorithms we have only done a sloppy cross validation to obtain the best values for the soft margin constraint C or the kernel functions as the aim of this chapter was only to show how the theory we developed in Chapter **??** and Chapter **??** works for real world problems in general. Though, the results already point out areas where further improvement could be possible. Also the runtime of the different algorithms could be reviewed.

For the multiclass classification there are more approaches than discussed in Chapter **??**, making it possible to extend the knowledge of implementing a good multicategory Support Vector Machine.

In order to test the generalization ability, other datasets such as the USPS database, which was already used by LeCun et al. [**?**] and Vapnik [**?**] in the 1990s, can serve as training and test data to obtain further results about error ratios.

References

[1] B. Schoelkopf, A. Smola. *Learning with Kernels : support vector machines, regularization, optimization, and beyond*. Cambridge, Mass. [u.a.] , MIT Press, New York, 2002

[2] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995

[3] https://de.mathworks.com/help/stats/svmtrain.html, accessed on 2.6.2017, 11:18

[4] G. Fischer. *Lineare Algebra. Eine Einführung für Studienanfänger*. Vieweg + Teubner, Wiesbaden 2010

[5] I. Bomze. *Lecture Notes Applied Optimization: Role of Duality*, Universität Wien, WiSe 2015/2016

[6] N. Das, J. M. Reddy, R. Sarkar, S. Basu, M. Kundu, M. Nasipuri, D. K. Basu. *A statistical–topological feature combination for recognition of handwritten numerals*. Applied Soft Computing, Volume 12, Issue 8, August 2012, Pages 2486–2495

[7] G. Gan, C. Ma, J. Wu. *Data Clustering: Theory, Algorithms and Applications*. ASA-SIAM Series on Statistics and Applied Probability, SIAM, Philadelphia, ASA, Alexandria, VA, 2007

[8] M. Miciak *Character recognition using Radon Transformation and Principal Component Analysis in postal applications*. International Multiconference on Computer Science and Information Technology, IMCSIT 2008, 2008, Pages 495–500

[9] J. Weston and C. Watkins. *Support Vector Machines for Multi-Class Pattern Recognition*. ESANN'1999 proceedings - European Symposium on Artificial Neural Networks Bruges (Belgium)

[10] N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998

[11] E.J. Bredensteiner, K.P. Bennett. *Multicategory classification by support vector machines*. Computational Optimization and Applications 12, 1999, Pages 53-79

[12] Y. Guermeur. *Combining discriminant models with new multi-class SVMs*. Pattern Analysis and Applications 5(2), 2002, Pages 168-179

[13] A.P.D. Silva. *Optimization Approaches to Supervised Classification*. European Journal of Operational Research, 2016

[14] J. C. Platt, N. Christianini, J. Shawe-Taylor. *Large Margin DAGs for Mutliclass Classification* . Advances in Neural Information Processing Systems 12, 2000, Pages 547-553

[15] Z. Yang, Y. Shao, X. Zhang. *Multiple birth support vector machine for multi-class classification*. Neural Comput and Applic, 2013, 153-161

[16] Y. LeCun, C. Cortes, C.J.C. Burges. *http://yann.lecun.com/exdb/mnist/*.

[17] S. Roweis. *http://www.cs.nyu.edu/ roweis/data.html*.

[18] M. Miciak. *Character recognition using Radon Transformation and Principal Component Analysis in postal applications, in: International Multiconference on Computer Science and Information Technology*. IMCSIT 2008, 2008, Pages 495–500

[19] J. C. Platt *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*. Technical Report MSR-TR-98-14 April 21, 1998

[20] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, V. Vapnik. *Feature Selection for SVMs*. Advances in Neural Information Processing Systems, 2001

[21] N. Cristianini, J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000

[22] T.M. Cover. *Geometrical and statistical propoerties of systems of linear inequalities with applications in pattern recognition*. IEEE Transaction on Electronic Computers 14 , 1965, Pages 326-334

[23] X. Tan , W. Bi, X. Hou, W. Wang *Reliability analysis using radial basis function networks and support vector machines*. Computers and Geotechnics 38, 2011, Pages 178–186

[24] M. Keglevic, R. Sablatnig *Master's Thesis: Automatic Recognition of Weather Records*,Computer Vision Lab Institute of Computer Aided Automation Vienna University of Technology, 2013

[25] B. Schölkopf, A.J. Smola, R.C. Williamson, and P.L. Bartlett. *New support vector algorithms*. Neural Computation 12, 2000, Pages 1207-1245

[26] B. Schölkopf, J.C. Burges, A. Smola. *Advances in Kernel Methods : Support Vector Learning* . Available from: eBook Super Collection - Austria, Ipswich, MA. Accessed May 24, 2017.

[27] J.H. Friedman. *Another approach to polychotomous classification. Technical report*. Stanford Department of Statistics, 1996

[28] T.G. Dietterich, G.Bakiri. *Solving Multiclass Learning Problems via Error-Correcting Output Codes*. Journal of Artificial Intelligence Research 2, 1995, Pages 263-286

[29] E.L. Allwein, R.E. Schapire, Y. Singer. *Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers* Journal of Machine Learning Research 1, 2000, Pages 113-141

[30] K. Crammer, Y. Singer. *On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines*. Journal of Machine Learning Research 2, 2001, Pages 265-292

[31] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.J. Jackel. *Handwritten digit recognition with backpropagation network*. Advances in Neural Information Processing System 2, Morgan Kaufmann, 1990, Pages 396-404

[32] K. Crammer, Y. Singer. *On the Learnability and Design of Output Codes for Multiclass Problems*. In Proceedings of the Thirteenth Annual Conference on Computational Learning Theory, 2000

[33] M. Yunqian, G. Guodong (Eds.). *Support Vector Machines Applications*. Springer International Publishing Switzerland, 2014

[34] http://www.svms.org/history.html, accessed on 2.6.2017, 9:38

[35] P.L. Bartlett, J. Shawe-Taylor. *Generalization performance of support vector machines and other pattern classifiers. In B. Schölkopf, C.J.C. Burges, A.J. Smola, editors, Advances in Kernel Methods - Support Vector Learning*. Cambridge, MA, 1999, MIT Press, Pages 43-54

[36] C.J.C. Burges. *A Tutorial on Support Vector Machines for Pattern Recognition*. Kluwer Academic Publishers, Boston

[37] C. Geiger, C. Kanzow. *Theorie und Numerik restringierter Optimierungsaufgaben*. Springer Verlag Berlin Heidelberg New York, 2002, Page 46