Rheinisch-Westfälische Technische Hochschule Aachen
Lehrstuhl für Informatik 6
Prof. Dr.-Ing. Hermann Ney

Selected Topics in Human Language Technology and Pattern Recognition (WS 2013)

Sparse Kernel Machines (incl. SVMs)

Tobias Pohlen

Matriculation number 308743

02/10/2014

Supervisor: Simon Wiesler

Contents

List of Tables

List of Figures

1 Introduction

The field of machine learning has grown to become one of the most important areas of modern computer science. There are many practical applications such as speech recognition or email filters that rely heavily on machine learning techniques. The general goal is to infer a pattern from *training data* and use this pattern in order to deal with unknown data. In this paper we specifically concentrate on the task of *classification* where we want to infer a decision rule from our training data. An example for a classification problem is an email spam filter: Decide whether or not an incoming email is spam. The training data consists of emails that are marked either as spam or not-spam. We want to infer a decision rule from this data that helps us categorize incoming emails. In order for us to deal with classification in a formal manner, we have to define it as a mathematical problem. Most of the results we present in this paper are based on the following formal definition of the *binary classification problem*:

Definition 1.1 (Binary classification problem)
Given a training set $X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\} \subset \mathbb{R}^m \times \{-1, 1\}$ *of* N training examples $\mathbf{x}_i$ *and* class labels y_i *one seeks to assign the unclassified vector* $\mathbf{x} \in \mathbb{R}^m$ *distinctly to one of the classes* -1 *or* 1. $\mathbb{R}^m$ *is called the* input space.

Note that we use the bold type face $\mathbf{x}_i$ to distinguish the vector $\mathbf{x}_i$ from the i-th entry x_i of the vector $\mathbf{x}$.

The binary classification problem is illustrated in figure 1. In order to solve the classification problem, we will first introduce *linear classifiers*. These classifiers use a very simple linear model to *separate* the two classes. To further extend their classification abilities, we will formulate the original model in the *dual formulation*. This will allow us to apply the theory of *kernel functions* which significantly enlarges the possible areas of application. Before we introduce *support vector machines (SVMs)* as the most popular example of *sparse kernel machines*, we will reason about the theory behind some of the the design choices that have been made during their development. Finally, we will analyze the learning process of SVMs which uses some results from optimization theory.

The theory of linear classifiers dates back to the 1930s where Fisher used linear discriminant functions in order to distinguish plants based on some measurements [Fis36]. The concept has been reused in the 1950s by Frank Rosenblatt who developed the popular perceptron learning rule [Ros58]. In the 1970s Vapnik did extensive research in the area of *statistical learning theory*. The results he obtained led to the development of support vector machines (summarized in [Vap95]).

2 Linear classifiers

The decision rule we mentioned in the introductory paragraph can be seen as a function h that maps our data to one of the classes. Such a function is called *classifier*. In case of the binary classification problem as defined in 1.1 a classifier is a function $h : \mathbb{R}^m \mapsto \{-1, 1\}$. In this paper we will focus on *linear classifiers* that use simple linear models in order to make a decision for one of the classes. These linear models are called *linear discriminant functions*. The following definition introduces linear classifiers and linear discriminant functions formally.

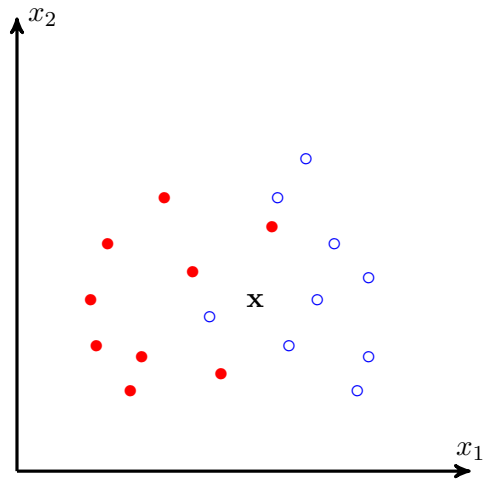

Figure 1: Assign the vector $\mathbf{x}$ to one of the two classes represented by the two kinds of circles.

Definition 2.1 (Linear discriminant functions and classifiers)
A linear discriminant function *in the* primal form *is defined as*

$$f : \mathbb{R}^m \mapsto \mathbb{R}, \mathbf{x} \mapsto \mathbf{w}^T\mathbf{x} + b = \sum_{i=1}^{m} w_i x_i + b \tag{1}$$

where $\mathbf{w} \in \mathbb{R}^m$ *is called* weight vector *and* $b \in \mathbb{R}$ *is called* bias. *The* linear classifier h_f *decides the class label based on the sign of* f*:*

$$h_f : \mathbb{R}^m \mapsto \{-1, 1\}, \mathbf{x} \mapsto sign(f(\mathbf{x})) = \begin{cases} 1 & \text{if } f(\mathbf{x}) \geq 0 \\ -1 & \text{otherwise} \end{cases} \tag{2}$$

Obviously, not all training sets can be perfectly classified by a linear classifier. Those training sets that can are said to be *linearly separable*.

Definition 2.2 (Linearly separable)
A linear classifier h_f *is said to* separate *the training set* $X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\} \subset \mathbb{R}^m \times \{-1, 1\}$ *if* $h_f(\mathbf{x}_i) = y_i,\ \forall i = 1, ..., N$. *A training set* X *is called* linearly separable *if there exists a linear classifier that separates the set.*

Learning or *training* a linear classifier essentially means that the parameters $\mathbf{w}$ and b of the discriminant function f have to be determined. We will see in the section about support vector machines how the parameters should be chosen theoretically and how they can be calculated practically.

Some authors refer to classifiers as hypotheses (e.g. [Vap95] or [CST00]). Then, learning means that a proper hypothesis h has to be chosen from the set of all hypotheses $\mathcal{H} = \{h_f \mid h_f \text{ is a linear classifier}\}$. We will avoid this terminology to be consistent throughout the paper.

Every classifier h divides the input space into *decision regions* $C_i \subset \mathbb{R}^m$ in which all vectors are assigned to the same class i. The boundary between those decision regions is called *decision surface*. When dealing with linear classifiers h_f, the decision surface is

defined by all $\mathbf{x} \in \mathbb{R}^m$ for which the equation $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0$ holds. These vectors effectively form an $(m-1)$ dimensional hyperplane in the input space. The situation is illustrated in figure 2.

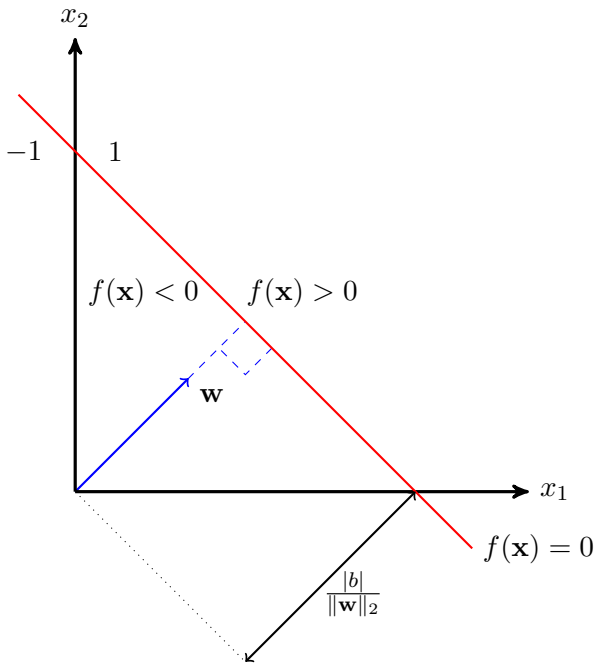

Figure 2: Decision surface (hyperplane) of a linear classifier h_f in $\mathbb{R}^2$.

In the section about support vector machines we will discuss what properties qualify the *best* linear classifier for a given training set. The discussion is based on the concept of *margins*. There are two kinds of margins. The first one is the so called *functional margin* that corresponds to the output of the linear discriminant function f.

Definition 2.3 (Funcional margin)
Let $X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\} \subset \mathbb{R}^m \times \{-1, 1\}$ *be a training set and let* h_f *be a linear classifier. The quantity* $\gamma_i^{func} = y_i f(\mathbf{x}_i)$ *is called* functional margin *of* $\mathbf{x}_i$.

The second one is the *geometric margin* that is the signed euclidean distance from a point $\mathbf{x}_i$ to the decision surface of h_f. The functional margin is not necessarily equal to the geometric margin as the following corollary illustrates.

Corollary 2.1 (Geometric margin)
Let $X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\} \subset \mathbb{R}^m \times \{-1, 1\}$ *be a training set and let* h_f *be a linear classifier. The signed euclidean distance* γ_i^{geo} *from a point* $\mathbf{x}_i$ *to the decision surface of* h_f *is given by*

$$\gamma_i^{geo} = \frac{y_i f(\mathbf{x}_i)}{\|w\|_2} \tag{3}$$

We refer to γ_i^{geo} *as the* geometric margin of $\mathbf{x}_i$.

Proof Let $\mathbf{x}_i = \mathbf{x}_i^\perp + \mu_i \frac{\mathbf{w}}{\|\mathbf{w}\|_2}$ be an orthogonal decomposition of $\mathbf{x}_i$ with $f(\mathbf{x}_i^\perp) = 0$ (i.e. $\mathbf{x}_i^\perp$ lies on the decision surface of h_f). Then μ_i is the signed distance of $\mathbf{x}_i$ to the decision surface. It follows

$$f(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i + b \tag{4}$$

$$= \underbrace{\mathbf{w}^T\mathbf{x}_i^\perp + b}_{=0} + \mu_i\|\mathbf{w}\|_2 \tag{5}$$

$$\Leftrightarrow \mu_i = \frac{f(\mathbf{x}_i)}{\|\mathbf{w}\|_2} \tag{6}$$

The margin, as defined so far, is a property of a single point $\mathbf{x}_i$. We want, however, to make it a property of the classifier in order to make use of the concept when we compare the quality of different classifiers later. This can easily be done by choosing one characteristic margin:

Definition 2.4 (Margin)
Let $X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\} \subset \mathbb{R}^m \times \{-1, 1\}$ be training set and let h_f be a linear classifier. The functional margin *γ^{func} of h_f on the training set X is defined as*

$$\gamma^{func} = \min_{1=1,...,N} y_i f(\mathbf{x}_i) = \min_{i=1,...,N} \gamma_i^{func} \tag{7}$$

Analog, the geometric margin *is defined as*

$$\gamma^{geo} = \frac{\gamma^{func}}{\|w\|_2} \tag{8}$$

2.1 Dual formulation

In the next section we will introduce the concept of kernel functions. Kernel functions can be used in a wide variety of algorithms by formulating them in the so called *dual formulation*. The general idea behind the dual formulation is that the input vectors only occur as parameters of some inner product $\langle \cdot, \cdot \rangle$. This formulation allows us to easily replace the inner product and hence make use of the so called *kernel trick*.

Theorem 2.1
Let $X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\} \subset \mathbb{R}^m \times \{-1, 1\}$ be a training set. For every linear classifier h_f there exists a linear classifier $h_{\tilde{f}}$ which has the same functional margin on X and whose weight vector $\tilde{\mathbf{w}}$ can be expressed as a linear combination of the training examples $\mathbf{x}_1, ..., \mathbf{x}_N$.

Proof Let $h_f, f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$, be an arbitrary linear classifier. Consider an orthogonal decomposition of the weight vector $\mathbf{w} = \mathbf{w}^\perp + \mathbf{w}_0$ with $\mathbf{w}_0 \in span\{\mathbf{x}_1, ..., \mathbf{x}_N\}$ and $(\mathbf{w}^\perp)^T\mathbf{w}_0 = 0$. Then

$$\gamma^{func} = \min_{i=1,...,N} y_i f(\mathbf{x}_i) = \min_{i=1,...,N} y_i(\underbrace{(\mathbf{w}^\perp)^T\mathbf{x}_i}_{=0} + \mathbf{w}_0^T\mathbf{x}_i + b) = \min_{i=1,...,N} y_i \left(\sum_{j=1}^{N} \alpha_j \mathbf{x}_j^T\mathbf{x}_i + b \right) \tag{9}$$

Hence, the linear classifier $h_{\tilde{f}}$ with $\tilde{f}(\mathbf{x}) = \sum_{i=1}^N \alpha_i \mathbf{x}_i^T\mathbf{x} + b = \tilde{\mathbf{w}}^T\mathbf{x} + b$, has the same functional margin and its weight vector $\tilde{\mathbf{w}}$ can be expressed as a linear combination of the training examples.

Theorem 2.1 justifies the following definition of the dual formation of a linear discriminant function.

Definition 2.5 (Dual formulation)
Let $X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\} \subset \mathbb{R}^m \times \{-1, 1\}$ *be a training set and let* $\langle \cdot, \cdot \rangle$ *be the standard inner product on* $\mathbb{R}^m$*. Then* f *is a linear discriminant function expressed in the* dual formulation

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b \tag{10}$$

During training the parameters $\alpha_1, ..., \alpha_N, b \in \mathbb{R}$ *have to be determined.*

As you might have noticed, there is one problem with the dual formulation: The computational complexity of the evaluation of the discriminant function in the primal form is linear in the dimension m of the input space due to the inner product $\langle \mathbf{w}, \mathbf{x} \rangle$. Whereas the complexity of the evaluation of the discriminant function in the dual representation is $\mathcal{O}(Nm)$ where N is the number of training examples. So it seems like we would punish large training sets that we usually desire when we want to learn something from data. In the light of sparse kernel machines we will see that the coefficients α_i, which are learned during training, are zero for most of the input vectors $\mathbf{x}_i$ and we, therefore, only need to compute the inner product $\langle \mathbf{x}_i, \mathbf{x} \rangle$ for a small subset of the training set. This means our learned model is *sparse*.

2.2 Multiclass problem

Up until now we only considered the binary classification problem. In practice however, we have to deal with *multiclass classification problems*.

Definition 2.6 (Multiclass classification problem)
Given a set of N *vectors* $X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\} \subset \mathbb{R}^m \times \{1, ..., C\}$ *one seeks to assign the the unclassified vector* $\mathbf{x} \in \mathbb{R}^m$ *distinctly to one of the classes* $1, .., C$*.*

Following the discussion in [CST00], we use multiple linear discriminant functions to solve this problem by converting it into C binary classification problems. For each class i we define a linear discriminant function f_i for which $f_i(\mathbf{x}_j) \geq 0$ if $y_j = i$ and $f_i(\mathbf{x}_j) < 0$ otherwise for all $(\mathbf{x}_j, y_j) \in X$. Our multiclass linear classifier $h_{f_1,...,f_C}$ is then defined as

$$h_{f_1,...,f_C}(\mathbf{x}) = \arg\max_{i=1,...,C} f_i(\mathbf{x}) \tag{11}$$

Hence, a new data point is assigned to the class whose decision surface is furthest from it. The decision surfaces between two classes are again hyperplanes and the decision regions are convex and simply connected [CST00].

3 Feature space mappings and kernel functions

So far, we only considered training sets which are linearly separable in the input space. Unfortunately, most training sets that come from practical problems do not have this property. In order to extend the range of learning sets that can be separated by linear classifiers, we introduce the concept of *features space mappings*. Feature space mappings

will allow us to separate the training data not in the input space, but in a high or even infinite dimensional *feature space*.

DEFINITION 3.1 (FEATURE SPACE MAPPINGS)
Let $\mathcal{H}$ be a D-dimensional Hilbert space with inner product $\langle \cdot, \cdot \rangle$. A non-linear function

$$\boldsymbol{\phi} : \mathbb{R}^m \mapsto \mathcal{H}, \mathbf{x} \mapsto (\phi_i(\mathbf{x}))_{i=1}^{D} \tag{12}$$

is called feature space mapping. *The ϕ_i are called* basis functions *and $\mathcal{H}$ is called* feature space.

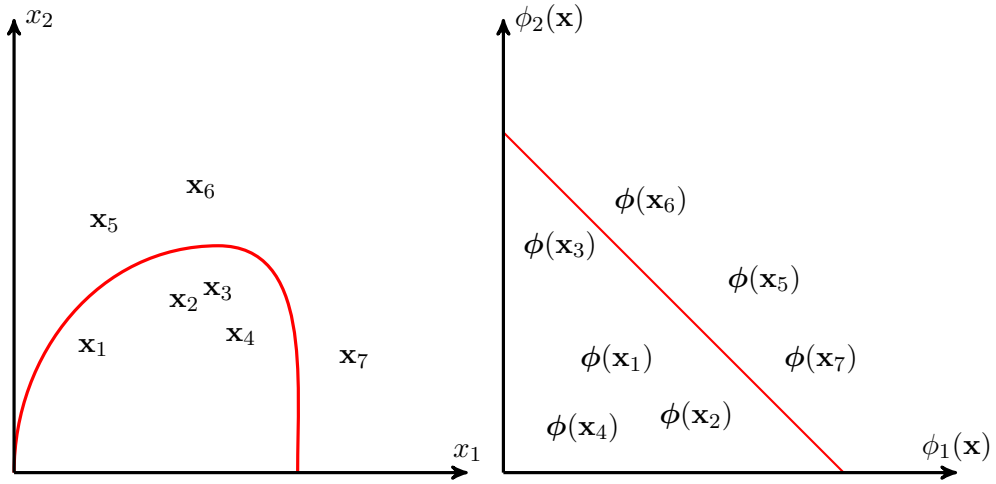

Figure 3: Linearly inseparable training set in the input space (left) and mapped training set in the feature space (right).

The general idea behind a feature space mapping goes as follows: If the training set is not separable in the input space, we transform it using a non-linear function into feature space in which it then is linearly separable. Such a mapping is illustrated in figure 3.

Our linear classifier now operates in the feature space $\mathcal{H}$ instead of the input space $\mathbb{R}^m$. Hence, the underlying linear discriminant function has the form

$$f : \mathbb{R}^m \mapsto \mathbb{R}, \mathbf{x} \mapsto \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}) \rangle + b \tag{13}$$

where $\mathbf{w} \in \mathcal{H}$ and $\langle \cdot, \cdot \rangle$ is the inner product of $\mathcal{H}$. We will see, however, that the dual formulation of the discriminant function is much more convenient in the context of feature space mappings because it allows us to use so called *kernel functions*.

$$f : \mathbb{R}^m \mapsto \mathbb{R}, \mathbf{x} \mapsto \sum_{i=1}^{N} \alpha_i \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}) \rangle + b \tag{14}$$

We essentially replaced the inner product of the vectors in the input space $\mathbb{R}^m$ by the inner product of the images of the vectors under the feature space mapping. This procedure is known as the *kernel trick* which leads us to the definition of *kernel functions*.

Definition 3.2 (Kernel functions)
Let $\boldsymbol{\phi} : \mathbb{R}^m \mapsto \mathcal{H}$ be a feature space mapping. Then

$$k(\mathbf{x}, \mathbf{y}) = \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{y}) \rangle \tag{15}$$

is called kernel function*, where $\langle \cdot, \cdot \rangle$ is the inner product of $\mathcal{H}$.*

We will see that often the kernel function can be computed much more efficiently than the explicit inner product of the images under the feature space mapping. Example 3.1 illustrates the idea. It is important to notice that in the context of kernel functions we can only use the dual formulation of the linear discriminant function. Hence, the linear discriminant function has the following form:

$$f : \mathbb{R}^m \mapsto \mathbb{R}, \mathbf{x} \mapsto \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b \tag{16}$$

Example 3.1 (Kernel function)
It seems rather counterintuitive that a kernel function can be computed more efficiently than the inner product under the feature space mapping. To illustrate this effect we choose the feature space $\mathcal{H} \equiv \mathbb{R}^3$ and the input space $\mathbb{R}^2$. We define the feature space mapping $\boldsymbol{\phi}$ as follows

$$\boldsymbol{\phi} : \mathbb{R}^2 \mapsto \mathbb{R}^3, \mathbf{x} \mapsto \begin{pmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{pmatrix} \tag{17}$$

Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2$. Then

$$\begin{aligned} \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{y}) \rangle =& \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{y}) && (18) \\ =& \sum_{i=1}^{3} \phi_i(\mathbf{x}) \phi_i(\mathbf{y}) && (19) \\ =& x_1^2 y_1^2 + 2 x_1 x_2 y_1 y_2 + x_2^2 y_2^2 && (20) \\ =& (\mathbf{x}^T \mathbf{y})^2 && (21) \end{aligned}$$

This means that the kernel function $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y})^2$ is just the squared dot product of $\mathbf{x}$ and $\mathbf{y}$ in the input space. Obviously, it is more efficient to compute the kernel function directly than to first compute the explicit feature space mapping and then the inner product of the images.

There are two major ways to get a kernel function:

1. Define an explicit feature space mapping $\boldsymbol{\phi}$ and try to find an efficiently computable form of $k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$
2. Define a function $k(\mathbf{x}, \mathbf{y})$ and show it corresponds to an inner product in some feature space $\mathcal{H}$

Example 3.1 already gave us the idea that the first approach can be challenging. We want to define some function $k(\mathbf{x}, \mathbf{y})$ and then show it corresponds to an inner product in some feature space. In the next section we will derive a way to do this.

3.1 Characterizing kernels

Throughout this section we will assume that we are given a function $k(\mathbf{x}, \mathbf{y})$. We will derive a sufficient condition for k to be a *valid kernel function over* $\mathbf{X} \times \mathbf{X}$, where $\mathbf{X} \subset \mathbb{R}^m$ is a compact subset of $\mathbb{R}^m$. Because our training set X is finite, we can always find a compact set $\mathbf{X}$ such that $X \subset \mathbf{X}$. Therefore, the requirement will not limit the practical areas of application.

Definition 3.3 (Valid kernel function over $\mathbf{X} \times \mathbf{X}$)
A function $k : \mathbf{X} \times \mathbf{X} \mapsto \mathbb{R}$ is called valid kernel function over $\mathbf{X} \times \mathbf{X}$ *if there exists a Hilbert space $\mathcal{H}$ and a feature space mapping $\boldsymbol{\phi} : \mathbf{X} \mapsto \mathcal{H}$ such that*

$$k(\mathbf{x}, \mathbf{y}) = \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{y}) \rangle, \ \forall \mathbf{x}, \mathbf{y} \in \mathbf{X} \tag{22}$$

where $\mathbf{X}$ is a compact subset of $\mathbb{R}^m$.

The theorem that gives us the sufficient condition utilizes the so called *kernel matrix* of k. The Kernel matrix is a generalization of the *Gram matrix* which you know from linear algebra.

Definition 3.4 (Kernel matrix)
Let $X = X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\} \subset \mathbf{X} \times \{-1, 1\}$. We define the kernel matrix $\mathbf{K}(X)$ *of k as follows*

$$\mathbf{K}(X) = (k(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1}^N \tag{23}$$

We will now follow the discussion in [Mer09] and [CST00] in order to derive Mercer's theorem.

The theorem is based on a generalized inner product in a D-dimensional Hilbert space $\mathcal{H}$ that has weightings $\lambda_i \geq 0$ for all dimensions: Let $\psi, \theta \in \mathcal{H}$

$$\langle \psi, \theta \rangle := \sum_{i=1}^{D} \lambda_i \psi_i \theta_i \tag{24}$$

Using this definition of an inner product, the theorem will give us necessary and sufficient conditions to admit such a representation

$$k(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{D} \lambda_i \phi(\mathbf{x})_i \phi(\mathbf{y})_i \tag{25}$$

where $\boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), ...)$ is a feature space mapping.

Theorem 3.1 (Mercer [CST00])
Let $\mathbf{X}$ be a compact subset of $\mathbb{R}^m$. Suppose k is a continuous symmetric function such that the integral operator $T_k : L_2(\mathbf{X}) \mapsto L_2(\mathbf{X})$,

$$(T_k f)(\cdot) = \int_{\mathbf{X}} k(\cdot, \mathbf{x}) f(\mathbf{x}) d\mathbf{x} \tag{26}$$

is non negative, that is

$$\int_{\mathbf{X} \times \mathbf{X}} k(\mathbf{x}, \mathbf{y}) f(\mathbf{x}) f(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0 \tag{27}$$

for all $f \in L_2(\mathbf{X})$. Then we can expand $k(\mathbf{x}, \mathbf{y})$ in a uniformly convergent series (on $\mathbf{X} \times \mathbf{X}$) in terms of T_k's eigen-functions $\phi_j \in L_2(\mathbf{X})$, normalized in such a way that $\|\phi_j\|_{L_2} = \sqrt{\int_{\mathbf{X}} \phi^2(\mathbf{x}) d\mathbf{x}} = 1$, and positive associated eigenvalues $\lambda_i \geq 0$,

$$k(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \phi_i(\mathbf{y}) = \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{y}) \rangle \tag{28}$$

where

$$\boldsymbol{\phi}(\mathbf{x}) := (\phi_i(\mathbf{x}))_{i=1}^{\infty} \tag{29}$$

Proof A proof can be found in [Mer09]

Definition 3.5 (Mercer kernel)
A function $k(\mathbf{x}, \mathbf{y})$ that satisfies the conditions of theorem 3.1 is called Mercer kernel over $\mathbf{X} \times \mathbf{X}$.

Let's recap the true meaning of theorem 3.1. It states that if the preconditions are fulfilled (the integral operator is non negative), then the function k corresponds to the inner product of T_k's eigen-functions. This result is useful for us because in order to verify if k is a valid kernel function, we neither have to find an appropriate Hilbert space $\mathcal{H}$ nor do we have to define a feature space mapping $\boldsymbol{\phi}$. By showing the conditions are fulfilled, we directly know $\mathcal{H}$ and $\boldsymbol{\phi}$ exist. The last question that remains open is: Given a symmetric continuous function k. How do we show the conditions are fulfilled? The following theorem gives us a condition which we can easily be verified in practice.

Theorem 3.2 (Mercer (modern version))
Let $k(\mathbf{x}, \mathbf{y})$ be a continuous symmetric function. Then

$$\int_{\mathbf{X} \times \mathbf{X}} k(\mathbf{x}, \mathbf{y}) f(\mathbf{x}) f(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0 \tag{30}$$

for all $f \in L_2(\mathbf{X})$ if and only if the kernel matrix $\mathbf{K}(X)$ is symmetric positive semi-definite for all finite subsets $X \subset \mathbf{X}$.

Proof A proof can be found in [CST00].

The following example illustrates how we can use theorems 3.2 and 3.1 to prove $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y})^2$ is a valid kernel function over $\mathbf{X} \times \mathbf{X}$, where $\mathbf{X}$ is an arbitrary compact subset of $\mathbb{R}^m$. In contrast to example 3.1 we now don't need to define a feature space or a feature space mapping.

Example 3.2 (Mercer's theorem)
Using Mercer's theorem we want to show that $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y})^2$ is a valid kernel function over $\mathbf{X} \times \mathbf{X}$, where $\mathbf{X}$ is an arbitrary compact subset of $\mathbb{R}^m$. Obviously, k is symmetric and continuous. All we have to prove is that the kernel matrix $\mathbf{K}(X)$ is positive semi-definite for all finite subsets $X \subset \mathbf{X}$. Since $\mathbf{x}^T \mathbf{y}$ is an inner product on $\mathbb{R}^m$, we know that the kernel matrix $\tilde{\mathbf{K}}(X)$ of $\tilde{k}(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y}$ is always positive semi-definite. Let $X \subset \mathbf{X}$ be an arbitrary finite subset. Then

$$\mathbf{x}^T \mathbf{K}(X) \mathbf{y} = \sum_{i=1}^{N} \sum_{j=1}^{N} x_i \mathbf{K}(X)_{i,j} y_i = \sum_{i=1}^{N} \sum_{j=1}^{N} x_i (\tilde{\mathbf{K}}(X)_{i,j})^2 y_i \geq 0$$

for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$. Therefore, the conditions of Mercer's theorem are fulfilled and it follows that k is a valid kernel function over $\mathbf{X} \times \mathbf{X}$ for all compact subsets $\mathbf{X} \subset \mathbb{R}^m$.

3.2 Constructing kernels

Using theorem 3.2, we can easily construct new kernels from existing ones. The following rules can simply be proved by showing that the kernel matrices are positive semi-definite.

THEOREM 3.3 (KERNEL CONSTRUCTION [CST00])
Let k_1 and k_2 be Mercer kernels over $\mathbf{X} \times \mathbf{X}, a > 0, f(\cdot)$ a real-valued function on $\mathbf{X}, \boldsymbol{\phi} : \mathbf{X} \mapsto \mathbb{R}^l$ with k_3 a kernel function over $\mathbb{R}^l \times \mathbb{R}^l$ and B a symmetric positive semi-definite matrix. Then the following functions are valid kernel functions:

1. $k(\mathbf{x}, \mathbf{y}) = k_1(\mathbf{x}, \mathbf{y}) + k_2(\mathbf{x}, \mathbf{y})$
2. $k(\mathbf{x}, \mathbf{y}) = ak_1(\mathbf{x}, \mathbf{y})$
3. $k(\mathbf{x}, \mathbf{y}) = k_1(\mathbf{x}, \mathbf{y})k_2(\mathbf{x}, \mathbf{y})$
4. $k(\mathbf{x}, \mathbf{y}) = f(\mathbf{x})f(\mathbf{y})$
5. $k(\mathbf{x}, \mathbf{y}) = k_3(\phi(\mathbf{x}), \phi(\mathbf{y}))$
6. $k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T B\mathbf{x}$

Proof A proof can be found in [CST00].

Two of the most used kernels in practice are the *polynomial kernel* and the *Gaussian kernel*.

COROLLARY 3.1 (POLYNOMIAL KERNEL)
Let $k_1(\mathbf{x}, \mathbf{y})$ be a kernel function over $\mathbf{X} \times \mathbf{X}, \mathbf{x}, \mathbf{y} \in \mathbf{X}$ and p a polynomial with positive coefficients. Then

$$k(\mathbf{x}, \mathbf{y}) = p(k_1(\mathbf{x}, \mathbf{y})) \tag{31}$$

is a valid kernel function over $\mathbf{X} \times \mathbf{X}$ for all compact subsets $\mathbf{X} \subset \mathbb{R}^m$.

Proof Using the rules stated in theorem 3.3 we derive:

a) Rule 3. $\Rightarrow k_1(\mathbf{x}, \mathbf{y})^i$ is a valid kernel for all $i > 0$

b) Rule 4. $\Rightarrow k_1(\mathbf{x}, \mathbf{y})^0 = 1$ is a valid kernel

Together with rules 1 and 2 it follows that

$$k(\mathbf{x}, \mathbf{y}) = p(k_1(\mathbf{x}, \mathbf{y})) = \sum_{i=0}^{l} a_i k_1(\mathbf{x}, \mathbf{y})^i \tag{32}$$

is a valid kernel function for $a_i \geq 0, i = 0, ..., l$.

COROLLARY 3.2 (GAUSSIAN KERNEL)
Let $\sigma > 0$. Then

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{-\|\mathbf{x} - \mathbf{y}\|^2}{\sigma^2}\right) \tag{33}$$

is a valid kernel function over $\mathbf{X} \times \mathbf{X}$ for all compact subsets $\mathbf{X} \subset \mathbb{R}^m$.

Proof Let $\mathbf{x}, \mathbf{y} \in \mathbf{X}$ arbitrary but fixed. Then $k(\mathbf{x}, \mathbf{y})$ can be written as

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{-\|\mathbf{x}-\mathbf{y}\|^2}{\sigma^2}\right) = \exp\left(\frac{-\|\mathbf{x}\|^2}{\sigma^2}\right)\exp\left(\frac{-\|\mathbf{y}\|^2}{\sigma^2}\right)\exp\left(\frac{2\mathbf{x}^T\mathbf{y}}{\sigma^2}\right) \tag{34}$$

Rule 4. already implies that the first two factors form a kernel. In order to show that k is a valid kernel function, we have to prove that $\exp\left(\frac{2\mathbf{x}^T\mathbf{y}}{\sigma^2}\right)$ is a valid kernel function. We define the polynomial p_N as

$$p_N(x) := \sum_{k=0}^{N} \frac{x^k}{k!} \tag{35}$$

Since $\left(\frac{2\mathbf{x}^T\mathbf{y}}{\sigma^2}\right)$ is a kernel function, corollary 3.1 implies $p_N\left(\frac{2\mathbf{x}^T\mathbf{y}}{\sigma^2}\right)$ is a kernel functions. too. Hence, also the limit of $\left(p_N\left(\frac{2\mathbf{x}^T\mathbf{y}}{\sigma^2}\right)\right)_{N\in\mathbb{N}}$ is a kernel. With $\lim_{N\to\infty} p_N(x) = \exp(x)$, it follows that $\exp\left(\frac{2\mathbf{x}^T\mathbf{y}}{\sigma^2}\right)$ is a valid kernel function.

4 Support Vector Machines (SVMs)

So far, we have seen seen linear discriminant functions and feature space mappings which can implicitly be used by enabling kernel functions. In this section we will introduce the concept of *support vector machines*(SVMs) as one very popular example of sparse kernel machines. To do so, we will first look at the *generalization* performance of linear classifiers and the use of *Structural Risk Minimization (SRM)* which will lead us to the formulation of *maximum margin classifiers*. Then we will give a brief introduction to optimization theory where we present the concept of *Lagrange multipliers* that can be used to derive *necessary conditions* for optima of *constrained optimization problems*. Using the results from these two sections, we can formulate the optimization problem of a maximum margin classifier whose solution yields the support vector approach.

4.1 Generalization error

In this section we want to define what linear classifier h_f should be chosen during training for a given training set X. Obviously, there are many possible choices for h_f as illustrated in figure 4. So the question we need to answer is: What is the *optimal* linear classifier h_f^* for a given training set X?

In order for us to answer this question, we first have to define what properties determine the quality of a linear classifier. To do so, we introduce the concept of the *expected generalization error*.

Definition 4.1 (Expected generalization error)
Assume our data (points and labels) is generated independently and identically according to a fixed but unknown distribution $\mathcal{D}$ over $\mathbb{R}^m \times \{-1, 1\}$. Then the expected generalization error *or* actual risk $R(h_f)$ *of a linear classifier h_f is defined as:*

$$R(h_f) = \sum_{y\in\{-1,1\}} \int_{\mathbb{R}^m} \frac{1}{2}|1 - yh_f(\mathbf{x})|p(\mathbf{x}, y)d\mathbf{x} \tag{36}$$

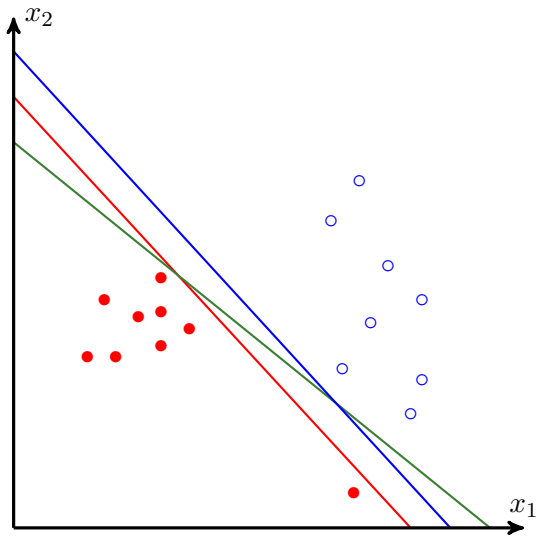

Figure 4: Which hyperplane corresponds to the decision surface of the optimal classifier h_f^* for this training set?

Using this concept, we can formally define the optimal linear classifier for a given training set.

Definition 4.2 (Optimal linear classifier)
Given a training set X that is generated according to a fixed but unknown distribution $\mathcal{D}$. The optimal *linear classifier h_f^* minimizes the expected generalization error.*

Unfortunately, computing the expected generalization error proposed in definition 4.1 is infeasible - even if we knew the distribution $\mathcal{D}$. To use this criterion as an optimality criterion nevertheless, we introduce a so called *pac* bound where pac stands for *probably approximately correct*. A pac bound is an upper bound ϵ of $R(h_f)$ which holds with a probability of $1 - \delta$ for all $\delta \in (0, 1)$. Vapnik showed in [Vap95] that there is a pac bound for the expected generalization error which depends on the *empirical risk* or *training error*, the number of independently and identically generated training examples N and the so called *Vapnik-Chervonenkis-dimension (VC-dimension)*. The following definitions explain everything that is needed in order to understand the bound.

Definition 4.3 (Empirical risk)
Let $X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\} \subset \mathbb{R}^m \times \{-1, 1\}$ be a training set and let h_f be a linear classifier. The empirical risk *or* training error *$R_{emp}(h_f)$ is defined as*

$$R_{emp}(h_f) = \frac{1}{N} \sum_{i=i}^{N} \frac{1}{2} |1 - y_i h_f(\mathbf{x}_i)| \tag{37}$$

In other words, the empirical risk $R_{emp}(h_f)$ is just the relative number of data points that are misclassified by h_f. The second concept that we need is the so called *VC-dimension*. It is a little bit harder to understand than the empirical risk because it is not a property of a single classifier h_f but a property of a *class of classifiers*. Example 4.1 will give an intuition how the definition can be interpreted.

Definition 4.4 (VC-dimension [Vap95])
The VC-dimension *of a class of classifiers $\mathcal{F}$ is the greatest number h such that there exists a set of h vectors $X = \{\mathbf{x}_1, ..., \mathbf{x}_h\}$ that can be separated in all 2^h possible ways. We write*

$$VC(\mathcal{F}) = h \tag{38}$$

We say the set X is shattered *by $\mathcal{F}$*

Example 4.1 (VC-dimension)
We will show that the VC-dimension of $\mathcal{F} = \{h_f \mid \mathbf{w} \in \mathbb{R}^2, b \in \mathbb{R}, f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b\}$ is at least 3. To do so, we have to choose three points in the x_1x_2-plane and show that we can separate them in all 6 possible ways using hyperplanes (i.e. lines in $\mathbb{R}^2$). From figure 5 it

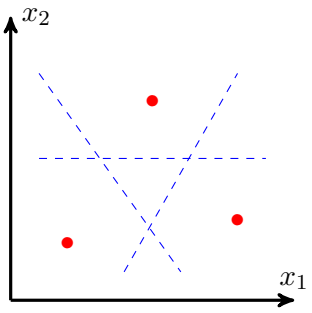

Figure 5: This set of three points is shattered by $\mathcal{F}$.

is clear that there is a set of three points that is shattered by $\mathcal{F}$. Therefore,

$$VC(\mathcal{F}) \geq 3 \tag{39}$$

The following theorem defines a pac bound on the actual risk.

Theorem 4.1 (Risk bound [Vap95])
Let $\delta \in (0, 1)$, $h_f \in \mathcal{F}$, $VC(\mathcal{F}) = h$. Then for every training set X of N independently and identically drawn points and labels the following upper bounds holds with a probability of at least $1 - \delta$.

$$R(h_f) \leq R_{emp}(h_f) + \underbrace{\sqrt{\frac{h(\log(\frac{2N}{h}) + 1) - \log(\frac{\delta}{4})}{N}}}_{\textit{VC-Confidence}} \tag{40}$$

Proof A proof can be found in [Vap95].

We can use this pac bound to find the optimal linear classifier. Note however, that the empirical risk depends on a single choice of h_f, whereas the VC-confidence depends on the VC-dimension of the class $\mathcal{F}$ that h_f is chosen from. In order to minimize the term with respect to both, the choice of h_f and the VC-dimension h, one has to make the VC-dimension a *controlling* variable. Vapnik introduced a concept called *Structural Risk Minimization (SRM)* [Vap95]. The the general idea goes as follows:
Suppose our class $\mathcal{F}$ has a *structure* consisting of nested subsets $\mathcal{F}_k \subset \mathcal{F}$ such that

$$\mathcal{F}_1 \subset \mathcal{F}_2 \subset ... \subset \mathcal{F}_k \subset ... \tag{41}$$

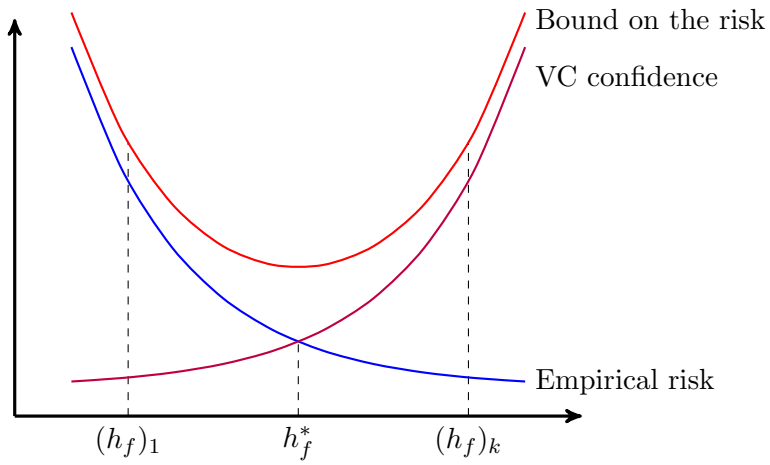

Figure 6: Finding the optimal classifier $h_f^* \in \mathcal{F}^*$.

and

$$VC(\mathcal{F}_1) \leq VC(\mathcal{F}_2) \leq ... \leq VC(\mathcal{F}_k) \leq ... \tag{42}$$

Then we can find the optimal linear classifier h_f by evaluating the VC-confidence for every $\mathcal{F}_k$ and finding the classifier $h_f \in \mathcal{F}_k$ that minimizes the empirical risk. The procedure is illustrated in figure 6. Intuitively, the class of classifiers which will minimize the pac-bound has a low VC-dimension and contains functions which yield a low empirical risk on the training set, meaning that they separate the training set (almost) perfectly. Vapnik showed in [Vap82] that the VC-dimension of a class of linear classifiers can be reduced by forcing a certain geometric margin γ^{geo} on a training set X:

$$\mathcal{F}_\gamma = \{h_f \mid h_f \text{ has geometric margin at least } \gamma \text{ on } X\} \tag{43}$$

Unfortunately, the hierarchy $\mathcal{F}_{\gamma_1} \subset \mathcal{F}_{\gamma_2} \subset ...$ cannot be used for SRM because SRM works only with hierarchies that are independent from the actual data. To solve this problem, Shawe-Tayler et. al introduced a generalization of SRM called *Data Dependent Structural Risk Minimization* which allows us to choose a data dependent hierarchy and apply SRM [STBWA98]. The key concept to this method is a generalization of the VC-dimension called *fat-shattering-dimension*. We will not go into detail here, but only point to the papers [STBWA98] and [BS99] in which the concept is explained in detail. Using their method, we get a pac bound on $R(h_f)$ which depends on the geometric margin γ^{geo} that h_f has on a training set X of N examples generated independently and identically according to $\mathcal{D}$.

THEOREM 4.2 (MARGIN BASED GENERALIZATION ERROR [STBWA98])
Define the class $\mathcal{F}$ of real-valued functions on the ball of radius R in $\mathbb{R}^m$ as

$$\mathcal{F} := \{\mathbf{x} \mapsto \mathbf{w}^T\mathbf{x} \mid \|\mathbf{w}\|_2 \leq 1, \|x\|_2 \leq R\} \tag{44}$$

There is a constant c such that, for all probability distributions, with a probability at least $1-\delta$ over N independently generated examples $\mathbf{x}_i$, the following bound holds for every

classifier $h_f, f \in \mathcal{F}$

$$R(h_f) \leq \frac{n}{m} + \sqrt{\frac{c}{N}\left(\frac{R^2}{\gamma^{geo2}}\log^2(N) + \log\left(\frac{1}{\delta}\right)\right)} \tag{45}$$

where n *is the number of training examples in* $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ *with geometric margin less than* γ^{geo}.

Proof A proof can be found in [STBWA98].

We now have an upper pac-bound on the expected generalization error that is dependent on the geometric margin. It is important to notice, that because the bound is based on the fat-shattering-dimension of $\mathcal{F}$ instead of the VC-dimension, it holds even if our class has an infinite VC-dimension. This may happen if we for example use a Gaussian kernel [Bur98]. Using the bound presented in theorem 4.2, we can define the optimal linear classifier w.r.t. a training set X.

Corollary 4.1 (Optimal linear classifier)
The optimal linear classifier *is the one with the greatest geometric margin. It is called* linear maximum margin classifier.

The decision surface of an optimal linear classifier is illustrated in figure 7.

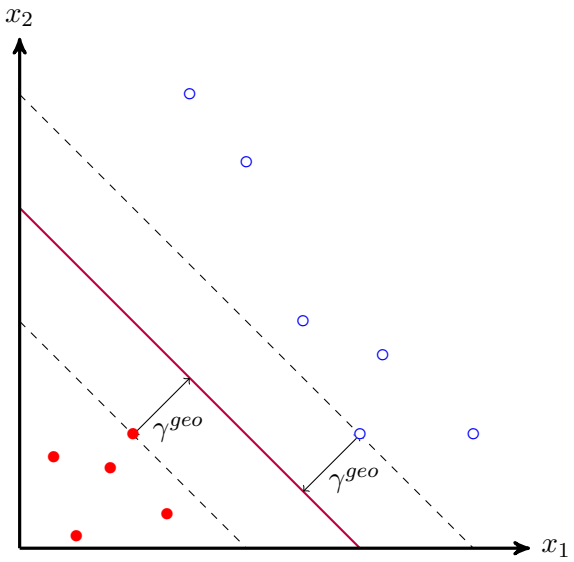

Figure 7: Decision surface of the optimal linear classifier for the given training set.

4.2 Optimization theory

When we want to use maximum margin classifiers in practice, we have to determine their parameters using numerical methods. The key tool regarding to SVM learning are the so called *Lagrange multipliers*. Lagrange multipliers can be used to tackle a wide range of different constrained optimization problems. In this chapter we will give a very brief

introduction to the topic such that the following explanations about SVM training can be understood. However, very basic concepts of unconstrained optimization are necessary in order to follow the discussion. For an in-depth introduction we refer to [NW06].

Definition 4.5 (Constrained optimization problem [NW06])
Let f and c_i be smooth, real-valued functions on a subset of $\mathbb{R}^m$, and $\mathcal{I}$ and $\mathcal{E}$ be two finite sets of indices. Then the constrained optimization problem *is defined as*

$$\min_{\mathbf{x}\in\mathbb{R}^m} f(\mathbf{x}) \text{ subject to} \begin{cases} c_i(\mathbf{x}) = 0, i \in \mathcal{E} \\ c_i(\mathbf{x}) \geq 0, i \in \mathcal{I} \end{cases} \tag{46}$$

$\mathcal{E}$ and $\mathcal{I}$ indicate the equality *and* inequality constraints *respectively. The set $\Omega \subset \mathcal{R}^m$ of all points which satisfy the constraints is called* feasible region *and f is called* objective function.

We only define the optimization problem as a minimization problem, because maximizing f is equivalent to minimizing $-f$. Very similar to unconstrained optimization, there are necessary conditions any local optimum $\mathbf{x}^*$ of f that satisfies the constraints has to fulfill. These are called the *Karush-Kuhn-Tucker conditions*. Before we can cite the theorem that states them, we have to define *active constraints* that will play an important role when it comes to the concept of sparsity in SVM training.

Definition 4.6 (Active constraints [NW06])
The active set $\mathcal{A}(\mathbf{x})$ at any point $\mathbf{x} \in \Omega$ consists of the equality constraint indices from $\mathcal{E}$ together with the indices of the inequality constraints i for which $c_i(\mathbf{x}) = 0$; that is

$$\mathcal{A}(\mathbf{x}) = \mathcal{E} \cup \{i \in \mathcal{I} \mid c_i(\mathbf{x}) = 0\} \tag{47}$$

We say the constraint $c_i, i \in \mathcal{I}$, is *active* at $\mathbf{x} \in \Omega$ if $c_i(\mathbf{x}) = 0$ and inactive if $c_i(\mathbf{x}) > 0$ is satisfied. The inequality constraints only put a lower bound of 0 on the values of c_i. An active inequality constraint can be understood as a constraint that actively forces the value c_i to be non-negative.

The final theorem about the necessary conditions of a local optimum requires one more important definition that characterizes a point $\mathbf{x} \in \Omega$.

Definition 4.7 (LICQ [NW06])
Given the point $\mathbf{x} \in \Omega$ and the active set $\mathcal{A}(\mathbf{x})$, we say that the linear independence constraint qualification (LICQ) *holds if the set of active constraint gradients $\{\nabla c_i(\mathbf{x}) \mid i \in \mathcal{A}(\mathbf{x})\}$ is linearly independent.*

This rather technical requirement will later assure that the Lagrange multipliers exist for the optimum we are searching. The following theorem will be the base result from optimization theory that is used to derive the dual optimization problem for support vector machines.

Definition 4.8 (Lagrange function)
Given the constrained optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^m} f(\mathbf{x}) \text{ subject to} \begin{cases} c_i(\mathbf{x}) = 0, i \in \mathcal{E} \\ c_i(\mathbf{x}) \geq 0, i \in \mathcal{I} \end{cases} \tag{48}$$

we define the Lagrange function *(or* Lagrangian*)* $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$ *as follows*

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i c_i(\mathbf{x}) \tag{49}$$

Minimizing f *is equivalent to minimizing* $\mathcal{L}$ *with respect to* f *and maximizing* $\mathcal{L}$ *with respect to* $\boldsymbol{\lambda}$*.*

THEOREM 4.3 (FIRST-ORDER NECESSARY CONDITIONS [NW06])
Suppose $\mathbf{x}^* \in \Omega$ *is a local optimum of (49), that the functions* f *and* c_i *are continuously differentiable, and that the LICQ holds at* $\mathbf{x}^*$*. Then there is a Lagrange multiplier vector* $\boldsymbol{\lambda}^* = (\lambda_i^*)_{i \in \mathcal{E} \cup \mathcal{I}}$ *such that the following conditions are satisfied at* $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$

$$\nabla_x \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) = 0 \tag{50}$$
$$c_i(\mathbf{x}^*) = 0, \quad \forall i \in \mathcal{E} \tag{51}$$
$$c_i(\mathbf{x}^*) \geq 0, \quad \forall i \in \mathcal{I} \tag{52}$$
$$\lambda_i^* \geq 0, \quad \forall i \in \mathcal{I} \tag{53}$$
$$\lambda_i^* c_i(\mathbf{x}^*) = 0, \quad \forall i \in \mathcal{E} \cup \mathcal{I} \tag{54}$$

Proof A proof can be found in [NW06].

These conditions are known as the *Karush-Kuhn-Tucker conditions* or short *KKT* conditions. The conditions (54) are called *complementary conditions* and will later be used to determine the so called *support vectors*.

4.3 Training

We now have all the knowledge we need to focus on the actual topic: Support vector machines. SVMs are maximum margin classifiers whose training algorithm gives a sparse model such that they can be evaluated very quickly. In this chapter we will define the optimization problem and show how it can be reduced to a quadratic programming problem (quadratic objective function, linear constraints) by the use of Lagrange multipliers. Throughout this chapter, we assume that $k(\mathbf{x}, \mathbf{y}) = \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{y}) \rangle$ is a valid kernel function over $\mathbf{X} \times \mathbf{X}$, where $\mathbf{X}$ is an arbitrary compact subset of $\mathbb{R}^m$.

4.3.1 Linearly separable case

First, we discuss the case when the data is linearly separable. In most practical applications, this is not the case, but the derivations are a foundation for later discussions.

We defined the decision surface of a linear classifier h_f as the hyperplane induced by the solution of the equation $f(\mathbf{x}) = 0$. Obviously, this hyperplane does not change when we rescale the weight vector $\mathbf{w}$ and the bias b with a $\lambda \in \mathbb{R}^+$. Hence the classifier $h_{\tilde{f}}$ with $\tilde{f}(\mathbf{x}) = \lambda \mathbf{w}^T \mathbf{x} + \lambda b$ has the same decision surface as h_f and therefore the same geometric margin γ^{geo} on all training sets X. However, the functional margin as defined in 2.3 (i.e. the output of $\tilde{f}$) changes. This means there is a degree of freedom with respect to the functional margin of the separating hyperplane. We can therefore fix the functional margin to 1 and maximize the geometric margin under this constraint [CST00].

In chapter 2 we showed that the geometric margin can be obtained by computing the functional margin of the linear discriminant function with a normalized weight vector $\mathbf{w}$. The following corollary gives us an idea how to maximize the geometric margin under the constraint that the functional margin is equal to 1.

COROLLARY 4.2
Let $f(\mathbf{x}) = \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}) \rangle + b$ be a linear discriminant function with $\gamma^{func} = 1$ on the training set $X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_n)\} \subset \mathbb{R}^m \times \{-1, 1\}$. Then

$$\gamma^{geo} = \frac{1}{\|\mathbf{w}\|_2} \tag{55}$$

Proof Let $\mathbf{w}$ and b realize $\gamma^{func} = 1$. Then

$$\gamma^{geo} = \min_{i=1,...,N} \frac{\gamma_i}{\|\mathbf{w}\|_2} \tag{56}$$

$$= \frac{\overbrace{\min_{i=1,...,N} \gamma_i}^{=\gamma^{func}=1}}{\|\mathbf{w}\|_2} \tag{57}$$

$$= \frac{1}{\|\mathbf{w}\|_2} \tag{58}$$

Corollary 4.2 says, that we can maximize the geometric margin, by fixing the functional margin to 1 and minimizing the l_2-norm of $\mathbf{w}$. We use this to define the corresponding optimization problem.

COROLLARY 4.3 ([CST00])
Let $X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\} \subset \mathbb{R}^m \times \{-1, 1\}$ be a linearly separable training set. The hyperplane $(\mathbf{w}^*, b^*)$ that solves the optimization problem

$$\min_{\mathbf{w} \in \mathbb{R}^m, b \in \mathbb{R}} \quad \|\mathbf{w}\|_2 \tag{59}$$

$$\textit{subject to} \quad \underbrace{y_i(\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i) \rangle + b)}_{=\gamma_i^{func}} \geq 1, i = 1, ..., N \tag{60}$$

realizes the maximum margin hyperplane with geometric margin $\gamma^{geo} = \frac{1}{\|w^*\|_2}$.

Note that the constraints given in 4.3 also make sure that the hyperplane separates the training set. To apply theorem 4.3 later, we have to reformulate the inequality constraints. The following definition is equivalent to the one given in corollary 4.3:

$$\min_{\mathbf{w} \in \mathbb{R}^m, b \in \mathbb{R}} \quad \|\mathbf{w}\|_2 \tag{61}$$

$$\text{subject to} \quad y_i(\langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle + b) - 1 \geq 0, i = 1, ..., N \tag{62}$$

Using this definition, we can make use of theorem 4.3 in order to derive the necessary conditions for a maximum margin hyperplane. To do so, we first define the Lagrange function. It is easy to see that instead of optimizing a function f directly, we can also optimize the function $g \circ f$, where g is a monotonic smooth, continuous function. Hence, given the training set $X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\} \subset \mathbb{R}^m \times \{-1, 1\}$ we define the Lagrange function $\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha})$ as follows, where $\boldsymbol{\alpha} = (\alpha_i)_{i=1}^N$ are the Lagrange multipliers:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|w\|_2^2 - \sum_{i=1}^{N} \alpha_i (y_i(\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i) \rangle + b) - 1) \tag{63}$$

This form of the Lagrangian is known as the *primal form*. In chapter 2.1 we introduced the dual formulation of a linear discriminant function. The dual formulation has the nice

property, that all the vectors only appear as arguments of an inner product $\langle \cdot, \cdot \rangle$. We find the dual formulation of the Lagrange function by making use of the KKT-conditions:

$$\frac{\partial}{\partial \mathbf{w}} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \mathbf{w} - \sum_{i=1}^{N} y_i \alpha_i \boldsymbol{\phi}(\mathbf{x}_i) \stackrel{!}{=} 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{N} y_i \alpha_i \boldsymbol{\phi}(\mathbf{x}_i) \tag{64}$$

$$\frac{\partial}{\partial b} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^{N} y_i \alpha_i \stackrel{!}{=} 0 \tag{65}$$

Substituting these relations back into the Lagrangian yields

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|w\|_2^2 - \sum_{i=1}^{N} \alpha_i (y_i(\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i) \rangle + b) - 1) \tag{66}$$

$$= \frac{1}{2} \left\langle \sum_{i=1}^{N} y_i \alpha_i \boldsymbol{\phi}(\mathbf{x}_i), \sum_{i=1}^{N} y_i \alpha_i \boldsymbol{\phi}(\mathbf{x}_i) \right\rangle$$
$$- \sum_{i=1}^{N} \alpha_i \left(y_i \left(\left\langle \sum_{i=j}^{N} y_j \alpha_j \boldsymbol{\phi}(\mathbf{x}_j), \boldsymbol{\phi}(\mathbf{x}_i) \right\rangle + b \right) - 1 \right) \tag{67}$$

$$= \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle$$
$$- \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle - b \underbrace{\sum_{i=1}^{N} \alpha_i y_i}_{=0} + \sum_{i=1}^{N} \alpha_i \tag{68}$$

$$= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \underbrace{\langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle}_{=k(\mathbf{x}_i, \mathbf{x}_j)} \tag{69}$$

The training vectors now only occur as arguments of an inner product $\langle \cdot, \cdot \rangle$ and we therefore can make use of kernel functions. This is known as the *dual formulation* of the Lagrangian.

THEOREM 4.4 (DUAL FORMULATION [CST00])
Let $X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\} \subset \mathbb{R}^m \times \{-1, 1\}$ be a linearly separable training set. Suppose $\boldsymbol{\alpha}^*$ solve the following optimization problem:

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^N} \quad \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{70}$$

$$\textit{subject to} \quad \sum_{i=1}^{N} y_i \alpha_i = 0 \tag{71}$$

$$\alpha_i \geq 0, i = 1, ..., N \tag{72}$$

Then the weight vector $\mathbf{w}^* = \sum_{i=1}^{N} y_i \alpha_i^* \boldsymbol{\phi}(\mathbf{x}_i)$ realizes the maximum margin hyperplane with geometric margin

$$\gamma^{geo} = \frac{1}{\|\mathbf{w}^*\|_2} \tag{73}$$

Proof We now have a maximization problem, because the Lagrange function was defined as a minimization w.r.t. $\mathbf{x}$ and a maximization w.r.t. $\boldsymbol{\alpha}$. The first constraint is a consequence of the necessary condition

$$\frac{\partial}{\partial b}\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^{N} y_i \alpha_i \stackrel{!}{=} 0 \tag{74}$$

just as the formulation of the weight vector is a consequence of the necessary condition

$$\frac{\partial}{\partial \mathbf{w}}\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \mathbf{w} - \sum_{i=1}^{N} y_i \alpha_i \boldsymbol{\phi}(\mathbf{x}_i) \stackrel{!}{=} 0 \tag{75}$$

The other constraints are the KKT conditions for inequality constraints as stated in theorem 4.3.

This formulation of the problem is a so called *quadratic programming problem (QP)*: We maximize a quadratic function of the form $f(\mathbf{x}) = \mathbf{x}^T G\mathbf{x} + \mathbf{c}^T\mathbf{x} + \mathbf{b}$ subject to linear constraints. Those problems can efficiently be solved using standard solvers as described in [NW06]. Note however, there are specialized solvers for SVM training (e.g. [Pla98]) that typically are much more efficient than generic QP solvers.

The following theorem implies that every local optimum of the optimization problem is a global optimum. This is an advantage over other classifiers such as neural networks which are very hard to train because of many local minima.

THEOREM 4.5
The optimization problem stated in corollary 4.6 is is a convex minimization problem.

Proof We can reformulate the maximization problem as a minimization problem by switching the sign of the objective function.

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^N} \quad -\sum_{i=1}^{N}\alpha_i + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{76}$$

$$\text{subject to} \quad \sum_{i=1}^{N} y_i \alpha_i = 0 \tag{77}$$

$$\alpha_i \geq 0, i = 1, ..., N \tag{78}$$

A sufficient condition for convexity a positive semi-definite hessian [NW06]. Let $\mathbf{K}(X)$ be the kernel matrix of k, where X is an arbitrary training set. Then the objective function can be stated as

$$f(\tilde{\boldsymbol{\alpha}}) = \frac{1}{2}\tilde{\boldsymbol{\alpha}}^T \mathbf{K}(X)\tilde{\boldsymbol{\alpha}} - (y_1, ..., y_N)^T\tilde{\boldsymbol{\alpha}} \tag{79}$$

where $\tilde{\boldsymbol{\alpha}} = (y_1\alpha_1, ..., y_N\alpha_N)^T$. Obviously

$$\nabla^2_{\tilde{\boldsymbol{\alpha}}} f(\tilde{\boldsymbol{\alpha}}) = \mathbf{K}(X) \tag{80}$$

Theorem 3.2 implies that $\mathbf{K}(X)$ is positive semi-definite. Therefore, the optimization is convex.

Once we have solved the optimization problem, we can evaluate the linear discriminant function, using either the primal or the dual formulation. When using a kernel function, the calculation of the weight vector $\mathbf{w}$ is not possible without knowing the exact underlying feature space mapping. In this case, we can only use the dual formulation.

COROLLARY 4.4
Let $X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\} \subset \mathbb{R}^m \times \{-1, 1\}$ be a linearly separable training set. If $\boldsymbol{\alpha}^*$ solves the optimization problem in 4.6, then the linear discriminant function can be evaluated for a new point $\mathbf{x} \in \mathbb{R}^m$ using the dual formulation:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b \tag{81}$$

Before we continue with the case when the training set is not linearly separable, there is one question we have to answer: Do we punish large training sets? We always want to get as much training examples as possible when we train a classifier. But the proposition of corollary 4.4 states, that for every evaluation of the linear discriminant function, we have to calculate the inner product of $\mathbf{x}$ and all the training vectors $\mathbf{x}_i$. When the training set is very large, which is usually what we desire, this is infeasible. Fortunately, there is a solution to this problem.

We already gave several hints about the *sparsity* of the model we get from the optimization problem. We can see this by considering the complementary conditions:

$$\alpha_i^*(y_i(\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i) \rangle + b - 1) = 0, i = 1, ..., N \tag{82}$$

This means that either a training example has a functional margin of 1 (i.e. it lies on the margin) **or** the Lagrange multiplier α_i^* is equal to zero. The points whose Lagrange multiplier are not equal to zero are called *support vectors* and are the points which are closest to the decision surface. We will denote the set of support vector indices by $\mathcal{SV}$. Empirically, the number of support vectors is only a very small fraction of total size of the training set meaning that the obtained model is *sparse*. Therefore, the computation of $f(\mathbf{x})$ is very efficient because we only have to consider the support vectors. All the other terms drop out:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b = \sum_{\imath \in \mathcal{SV}} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b \tag{83}$$

The complementary conditions yield another interesting fact about the model: Only the support vectors influence the decision surface. Points that lie far away from it have no impact on the model whatsoever. This means that SVMs are generally less sensitive to outliers than many other classifiers. We can think of this as an implication of the discussion about the generalization abilities of classifiers. SVMs are designed in a way such that they give the theoretically best generalization performance. A classifier that is insensitive to outliers is also less likely to overfit the training set.

Figure 8 shows the decision surface of a maximum margin classifier with its support vectors highlighted.

As you might have noticed: The bias parameter b dropped out of the optimization problem. It is implicitly defined over the constraints. We can therefore calculate it by

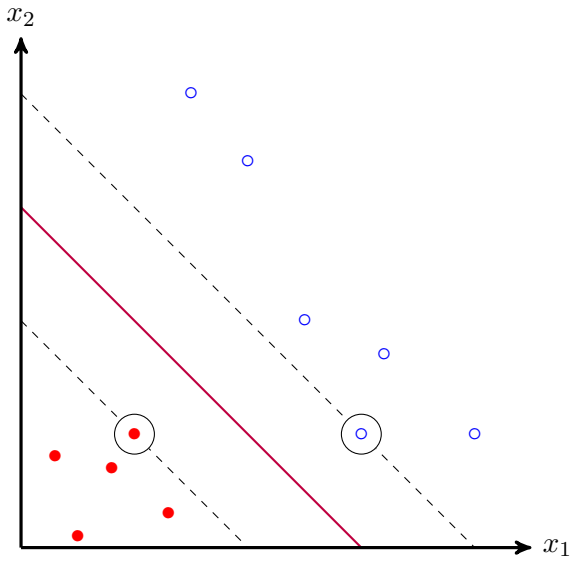

Figure 8: Decision surface of a maximum margin classifier with highlighted support vectors. The dotted lines are the so called *margin boundaries*.

choosing a support vector $\mathbf{x}$ with class label y:

$$yf(\mathbf{x}) = 1 \tag{84}$$

$$\Leftrightarrow \; y\left(\sum_{i\in\mathcal{SV}} y_i\alpha_i k(\mathbf{x}_i,\mathbf{x}) + b\right) = 1 \tag{85}$$

$$\overset{y\in\{-1,1\}}{\Leftrightarrow} \; b = y - \sum_{i\in\mathcal{SV}} \alpha_i y_i k(\mathbf{x}_i,\mathbf{x}) \tag{86}$$

Bishop proposed that the calculation is numerically more stable if we take the average over all support vectors [Bis06]. That is

$$b = \frac{1}{|\mathcal{SV}|}\sum_{n\in\mathcal{SV}}\left(y_n - \sum_{i\in\mathcal{SV}} y_i\alpha_i k(\mathbf{x}_i,\mathbf{x}_n)\right) \tag{87}$$

4.3.2 Linearly inseparable case

In this section, we will discuss the case when the training set is not linearly separable. This is usually the case we have to deal with in practice. The concept we will introduced is called *soft margin*. The general idea goes as follows: We try to maximize the margin, but allow some points to lie on the wrong side of margin boundary. In this paper we will only consider the so called 1-norm soft margin. There are, however, other concepts available such as the 2-norm soft margin [CST00].

Following the discussion in [Bis06] we introduce *slack variables* $\xi_i \geq 0, i = 1, ..., N$ that penalize points by their distance from their respective margin boundary. To define the slack variables, we have to consider two cases: For points that are classified correctly and do not lie within the margin, the slack variables are defined as $\xi_i = 0$. For all other

points (i.e. points that are inside the margin boundaries or misclassified) we define the slack variables as $\xi_i = |y_i - f(\mathbf{x}_i)|$. Using these definitions, we can propose the 1-norm soft-margin optimization problem:

$$\begin{aligned} \min_{\mathbf{w}\in\mathbb{R}^m, b\in\mathbb{R}, \boldsymbol{\xi}\in\mathbb{R}^N} \quad & \|\mathbf{w}\|_2 + C\|\boldsymbol{\xi}\|_1 && (88)\\ \text{subject to} \quad & y_i(\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i)\rangle + b) - 1 + \xi_i \geq 0, i = 1, ..., N && (89)\\ & \xi_i \geq 0, i = 1, ..., N && (90) \end{aligned}$$

where $C > 0$ is a trade off parameter that determines how strongly we penalize points that do not satisfy the maximum margin constraints. As before, we define the Lagrange function in order to make use of the KKT conditions. The KKT conditions for the Lagrangian

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + C\|\boldsymbol{\xi}\|_1 - \sum_{i=1}^{N} \alpha_i \left(y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) - 1 + \boldsymbol{\phi}(\xi_i)\right) - \sum_{i=1}^{N} \alpha_{N+i}\xi_i \quad (91)$$

are

$$\begin{aligned} \alpha_i \geq 0, \quad & \forall i = 1, ..., 2N && (92)\\ y_i(\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i)\rangle + b) - 1 + \xi_i \geq 0, \quad & \forall i = 1, ..., N && (93)\\ \alpha_i(y_i(\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i)\rangle + b) - 1 + \xi_i) = 0, \quad & \forall i = 1, ..., N && (94)\\ \xi_i \geq 0, \quad & \forall i = 1, ..., N && (95)\\ \alpha_{N+i}\xi_i = 0, \quad & \forall i = 1, ..., N && (96) \end{aligned}$$

Again, we use the necessary first-order conditions to find the dual formulation of the optimization problem. Note that the Lagrangian is differentiable w.r.t. to ξ_i because $\xi_i \geq 0$ and therefore $\|\boldsymbol{\xi}\|_1 = \sum_{i=1}^{N} \xi_i$.

$$\frac{\partial}{\partial \mathbf{w}}\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \mathbf{w} - \sum_{i=1}^{N} y_i\alpha_i\mathbf{x}_i \stackrel{!}{=} 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{N} y_i\alpha_i\mathbf{x}_i \quad (97)$$

$$\frac{\partial}{\partial b}\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^{N} y_i\alpha_i \stackrel{!}{=} 0 \quad (98)$$

$$\frac{\partial}{\partial \xi_k}\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = C - \alpha_k - \alpha_{N+k} \stackrel{!}{=} 0 \Rightarrow C = \alpha_k + \alpha_{N+k} \quad (99)$$

Substituting these conditions back into (91) yields

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + C\|\boldsymbol{\xi}\|_1 - \sum_{i=1}^{N} \alpha_i \left(y_i(\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i)\rangle + b) - 1 + \xi_i\right) - \sum_{i=1}^{N} \alpha_{N+i}\xi_i \quad (100)$$

$$= \frac{1}{2}\left\langle \sum_{i=1}^{N} y_i\alpha_i\boldsymbol{\phi}(\mathbf{x}_i), \sum_{i=1}^{N} y_i\alpha_i\boldsymbol{\phi}(\mathbf{x}_i)\right\rangle + C\|\boldsymbol{\xi}\|_1 - \sum_{i=1}^{N} \alpha_i \left(y_i\left(\left\langle \sum_{k=1}^{N} y_k\alpha_k\boldsymbol{\phi}(\mathbf{x}_k), \boldsymbol{\phi}(\mathbf{x}_i)\right\rangle + b\right) - 1 + \xi_i\right) - \sum_{i=1}^{N} \alpha_{N+i}\xi_i \quad (101)$$

$$= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_iy_j\alpha_i\alpha_j\langle\boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j)\rangle + C\|\boldsymbol{\xi}\|_1 - \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j y_iy_j\langle\boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j)\rangle + b\underbrace{\sum_{i=1}^{N}\alpha_i y_i}_{=0} + \sum_{i=1}^{N}\alpha_i - \underbrace{\sum_{i=1}^{N}\xi_i\underbrace{(\alpha_i + \alpha_{N+i})}_{=C}}_{=C\|\boldsymbol{\xi}\|_1} \quad (102)$$

$$= \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_iy_j\alpha_i\alpha_j\underbrace{\langle\boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j)\rangle}_{=k(\mathbf{x}_i,\mathbf{x}_j)} \quad (103)$$

This means, that the soft margin optimization problem in the dual form has the same objective function as the usual maximum margin problem. Only the constraints are different - but still linear. Therefore, it is still a quadratic programming problem.

THEOREM 4.6
Let $X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\} \subset \mathbb{R}^m \times \{-1, 1\}$ be a training set. Suppose $\boldsymbol{\alpha}^*$ solve the following optimization problem:

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^N} \quad \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_iy_j\alpha_i\alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \quad (104)$$

$$\textit{subject to} \quad \sum_{i=1}^{N} y_i\alpha_i = 0 \quad (105)$$

$$0 \le \alpha_i \le C, i = 1, ..., N \quad (106)$$

Then the weight vector $\mathbf{w}^* = \sum_{i=1}^{N} y_i\alpha_i^*\boldsymbol{\phi}(\mathbf{x}_i)$ realizes the soft margin hyperplane with geometric margin

$$\gamma^{geo} = \frac{1}{\|\mathbf{w}^*\|_2} \quad (107)$$

Proof All that is left to prove is the second constraint

$$0 \le \alpha_i \le C, i = 1, ..., N \quad (108)$$

This follows directly from the constraints $\alpha_i \ge 0$ and $C - \alpha_i - \alpha_{N+i} = 0$:

$$C - \alpha_i - \alpha_{N+i} = 0 \quad (109)$$

$$\Rightarrow \alpha_i = C - \underbrace{\alpha_{N+i}}_{\ge 0} \le C \quad (110)$$

Table 1: SVM performance benchmark as summarized by Vapnik in [Vap95]

Classifier	Best parameter choice	$\lvert\mathcal{SV}\rvert$	Raw error in %
Human performance	-	-	2.5%
Decision tree, C4.5	-	-	16.2%
Best two-layer neural network	-	-	5.9%
Five-layer network (LeNet 1)	-	-	5.1%
SVM with polynomial kernel	$d = 3$	274	4.0%
SVM with Gaussian kernel	$\sigma^2 = 0.3$	291	4.1%

4.4 Comparison

In chapter 4.1 we saw that the maximum margin classifier minimizes the expected generalization error. In order to compare the performance of support vector machines to other classifiers, one has to choose some benchmark training set. One of the standard benchmark training sets is the US Postal Service (USPS) set that contains 9298 handwritten digits collected from mail envelopes. The training set is publicly available under `http://www.kernel-machines.org/data.html`. Vapnik summarized the results of various performance benchmarks in [Vap95].

For the benchmark they chose three different kernels of which we only covered two in this paper. Namely

1. A polynomial kernel

$$k(\mathbf{x}, \mathbf{y}) = \left(\frac{\langle \mathbf{x}, \mathbf{y} \rangle}{256} \right)^d, d = 1, .., 7 \tag{111}$$

2. A Gaussian kernel

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\langle \mathbf{x}, \mathbf{y} \rangle}{256\sigma^2} \right) \tag{112}$$

The results are summarized in table 1.

4.5 Limitations

So far, we only discussed the abilities of support vector machines. Unfortunately, there are some limitations. The biggest one might be the choice of the kernel function. It is still an open problem how the best kernel function for a given problem should be chosen [Bur98].

Another disadvantage of SVMs is the computationally complexity of the training as well as of the evaluation. Standard quadratic programming solvers have a computational complexity of $\mathcal{O}(N^3)$ where N is the number of variables. Therefore the computational complexity is $\mathcal{O}(m^3)$ for the primal and $\mathcal{O}(N^3)$ for the dual formulation of the optimization problem. There are, however, approaches like SMO that are specialized on SVM training and perform significantly faster than generic quadratic programming problem solver [Pla98]. The computational complexity of the evaluation of the linear discriminant in the dual form is linear in the number of support vectors. Hence, if the training yields

a very large number of support vectors, the evaluation becomes practically infeasible for real world applications. This is why some authors choose linear SVMs (i.e. support vector machines that use the dot product as kernel function) for real time applications (e.g. HoG detector [DT05]) because the weight vector can easily be computed and then be used in the primal form of the linear discriminant function. The computational complexity of the evaluation of the discriminant function in the primal form is linear in the dimension of the input space and can therefore be significantly more efficient.

5 Conclusion

In this paper, we first introduced linear discriminant functions which can be used to solve the binary classification problem. We initially defined them in the primal form and then showed that there always is a dual form that has the same margin on the training set. Then we saw how feature space mappings can be used implicitly via kernel functions and how one can determine if a given function is a valid kernel function. Using the results obtained about valid kernel functions, we derived rules to build new kernels from existing ones. However, we only looked at kernel functions for numerical data. There are other approaches that define kernels for a wide variety of structures such as texts or DNA sequences [Her01].

We started the discussion about support vector machines with some reasoning about the question, what is the best linear classifier given a training set. Using Vapnik's results from statistical learning theory, we saw that the optimal linear classifier is the maximum margin classifier. In order to calculate the parameters of the said classifier in practice we considered the training for the linearly separable and the linearly inseparable case where we made use of Lagrange multipliers and the KKT conditions to derive the dual formulation of the optimization problems which can be solved using quadratic programming solvers. The discussion about optimization also showed that the maximum margin hyperplane is only defined by the so called support vectors (i.e. those points which are closest to the hyperplane). Empirically, the number of support vectors is significantly smaller than the number of training examples which yields a sparse model that can be used in order to evaluate the linear discriminant function efficiently. Finally, we cited Vapnik's summary on the classification performance of SVMs compared to other classifiers on the USPS benchmark set.

References

[BGV92] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.

[Bis06] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[BS99] P Bartlett and J Shawe–Taylor. Generalization performance of support vector machines and other pattern classifiers. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 43–54, Cambridge, MA, 1999. MIT Press.

[Bur98] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, June 1998.

[CST00] N. Christiani and J. Shawe-Taylor. *An introduction to support vector machines*. Cambridge University Press, 2000.

[DHS00] R. O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification Second Edition*. John Wiley and Sons, Inc, 2000.

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In Cordelia Schmid, Stefano Soatto, and Carlo Tomasi, editors, *International Conference on Computer Vision & Pattern Recognition*, volume 2, pages 886–893, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005.

[Fis36] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics, 7*, pages 179–188, 1936.

[GL13] G. H. Golub and C.F. Van Loan. *Matrix Computations 4th Edition*. John Hopkins, 2013.

[Her01] R. Herbrich. *Learning Kernel Classifiers*. MIT Press, 2001.

[Mer09] J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London*, 209:415–446, 1909.

[NW06] J. Nocedal and S. J. Wrigt. *Numerical optimization*. Springer, second edition edition, 2006.

[Pla98] John C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, ADVANCES IN KERNEL METHODS - SUPPORT VECTOR LEARNING, 1998.

[Ros58] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[STBWA98] John Shawe-Taylor, Peter L. Bartlett, Robert C. Williamson, and Martin Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44(5):1926–1940, 1998.

[STC06] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2006.

[TK00] S. Tong and D. Koller. Restricted bayes optimal classifiers. pages 658–664, 2000.

[Vap82] Vladimir Vapnik. *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics (Springer Series in Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

[Vap95] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, 1995.