Artificial Intelligence

UNI FREIBURG

Albert-Ludwigs-Universität Freiburg

Thorsten Schmidt

Abteilung für Mathematische Stochastik

www.stochastik.uni-freiburg.de
thorsten.schmidt@stochastik.uni-freiburg.de
SS 2017

Our goal today

Understand Support Vector Machines on a deeper level !

Support vector machines

- The statistical classification problem
  - Support Vector Classifier
  - The kernel trick
- A short excursion to convex optimization
- The kernel trick
- Reproducing kernel Hilbert spaces
  - Kernel examples
- MNIST - data: an example

Literature (incomplete, but growing):

- Ian Goodfellow, Yoshua Bengio und Aaron Courville (2016). **Deep Learning**. http://www.deeplearningbook.org. MIT Press
- D. Barber (2012). **Bayesian Reasoning and Machine Learning**. Cambridge University Press
- Richard S. Sutton und Andrew G. Barto (1998). **Reinforcement Learning : An Introduction**. MIT Press
- Gareth James u. a. (2014). **An Introduction to Statistical Learning: With Applications in R**. Springer Publishing Company, Incorporated. ISBN: 1461471370, 9781461471370
- T. Hastie, R. Tibshirani und J. Friedman (2009). **The Elements of Statistical Learning**. Springer Series in Statistics. Springer New York Inc. URL: https://statweb.stanford.edu/~tibs/ElemStatLearn/

Support vector machines

- The first example of a tool we visit but which is not typical for classical statistics is **Support Vector Machines**.
- For the introduction we mainly follow Hastie et. al. (2009) and Steinwart & Covel[1].

[1] I. Steinwart und C. Scovel (2007). „Fast rates for support vector machines using Gaussian kernels“. In: **Ann. Statist.** 35.2, S. 575–607.

- We start by formally introducing the **statistical classification problem**.
- We have a finite training set

$$T = ((x_1, y_1), \dots, (x_n, y_n)) \in (X \times Y)^n,$$

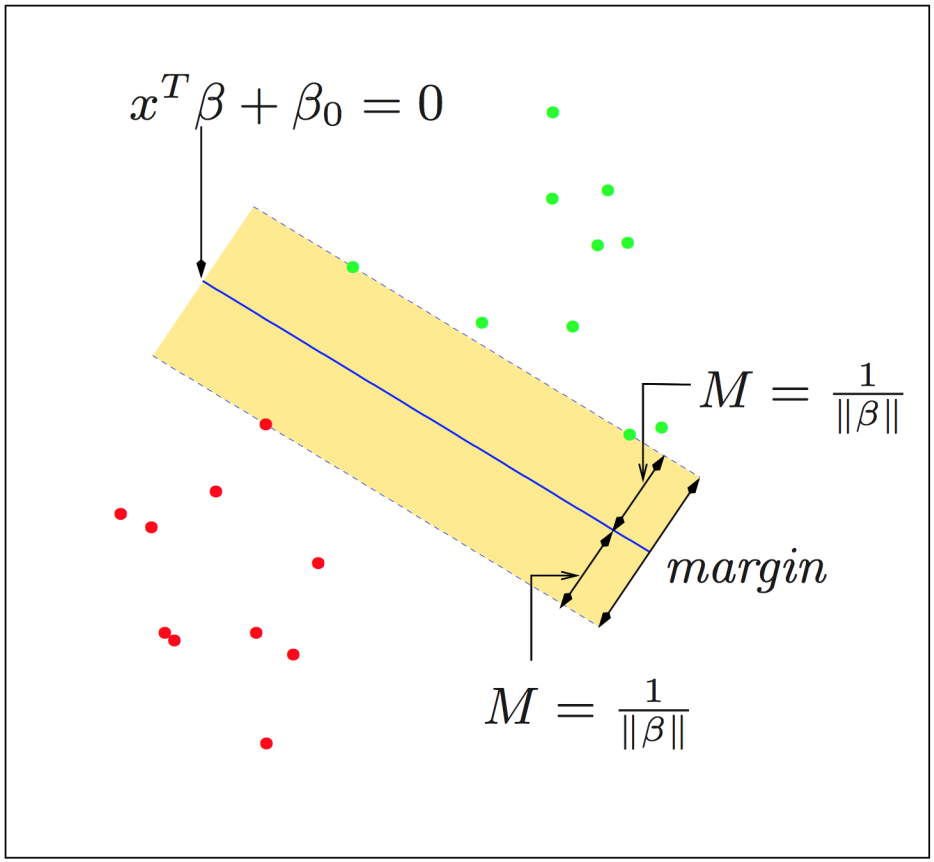
  where $X \subset \mathbb{R}^d$ and $Y = \{-1, 1\}$.
- The standard **batch model** assumes that the samples $(x_i, y_i)_{1 \le i \le n}$ are i.i.d. according to an unknown probability measure P on $X \times Y$. Furthermore, a new sample (x, y) is drawn from P independently of T.
- A **classifier** $\mathscr{C}$ assigns to every T a measurable function $f = f_T : X \to \mathbb{R}$.
- The prediction of $\mathscr{C}$ for y is

$$\operatorname{sign} f(x)$$

  with the convention $\operatorname{sign}(0) := 1$.
- We measure the quality of the classification f by the **classification risk**

$$\mathscr{R}(f) := P(\{(x, y) : \operatorname{sign} f(x) \neq y\}).$$

- Clearly, it is the goal to achieve the smallest possible risk, the so-called **Bayes risk**
$$\mathscr{R}_P := \inf\{\mathscr{R}(f) | f : X \to \mathbb{R} \text{ measurable}\}.$$
- A function which attains this level is called a **Bayes decision function**.
- Let us start with an illustrative introduction to SVM (Pictures taken from Hastie et.al. (2009))



Consider the hyperplane described by $x^\top \beta + \beta_0 = 0$, with $\| \beta \| = 1$ and the classification $\mathcal{G}$:

$$\text{sign}(x^\top \beta + \beta_0).$$

The maximal margin is obtained by the following optimization problem

$$\max_{\beta, \beta_0 : \|\beta\| = 1} M$$

subject to $y_i(x_i^\top \beta + \beta_0) \geq M, \ i = 1, \ldots, n$

- Such classifiers, computing a linear combination of the input and returning the sign were called **perceptrons** in the late 1950s (Rosenblatt, 1958) and set the foundations for later models of neural networks in the 80s and the 90s.
- We reformulate this criterion as follows: first, we get rid of $\| \beta \| = 1$ by considering

$$\frac{1}{\| \beta' \|} y_i (x_i^\top \beta' + \beta_0') \geq M$$

or, equivalently

$$y_i (x_i^\top \beta' + \beta_0') \geq M \| \beta' \|.$$

- With β', β_0' satisfying these equations, any (positive) multiple will also satisfy these, we rescale to $\| \beta' \| = M^{-1}$ and arrive at

$$\begin{aligned} &\min_{\beta, \beta_0} \frac{1}{2} \| \beta \|^2 \\ &\text{subject to } y_i (x_i^\top \beta + \beta_0) \geq 1, \ i = 1, \dots, n. \end{aligned} \tag{1}$$

- This is a convex optimization problem and can be solved via the classical Karush-Kuhn-Tucker conditions.
- It should be noted that the solution does only depend on a small amount of the data, and hence has a certain kind of robustness. On the other side, it will possibly not be optimal under additional information on the underlying distribution.

Non-linearly separable data

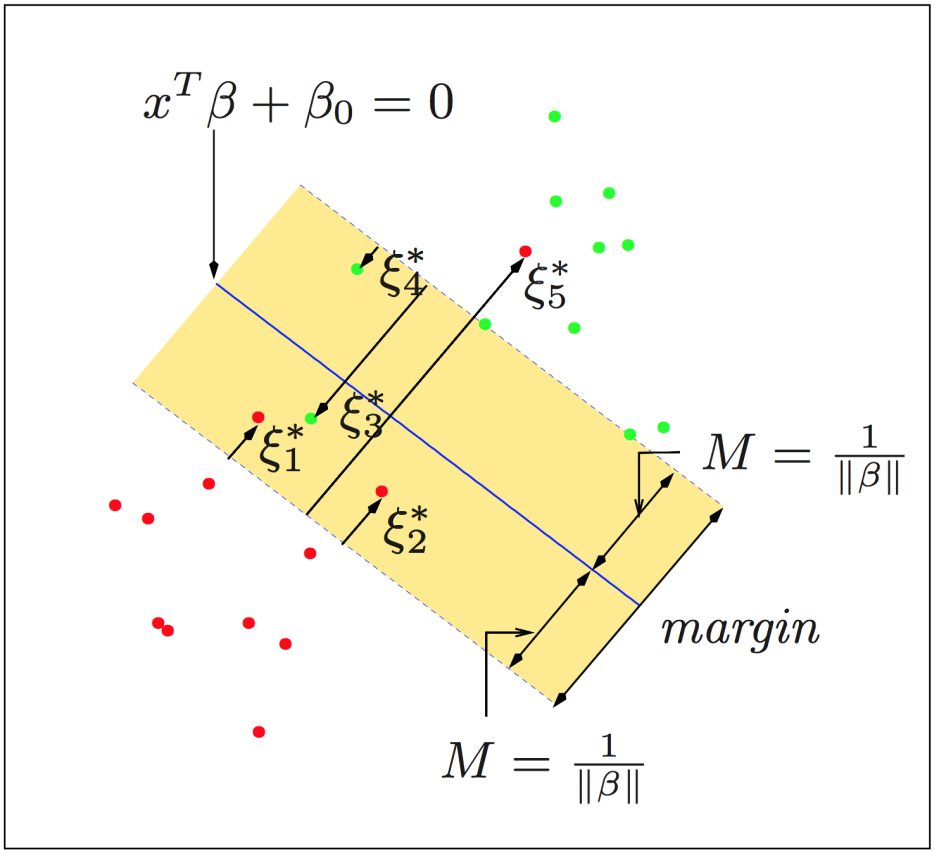
- When the data does not separate fully we will allow some points to be on the wrong side.
- In this regard, define the **slack variables** $\xi_1, \dots, \xi_n$ and consider

$$y_i(x_i^\top \beta + \beta_0) \geq M - \xi_i \tag{2}$$

or

$$y_i(x_i^\top \beta + \beta_0) \geq M(1 - \xi_i) \tag{3}$$

  with $\xi_i \geq 0$, $\sum \xi_i \leq K$ with a constant K.
- The first conditions seems more natural, while the second choice measures the overlap in relative distance, which chances with the width of the margin, M. However, (2) leads to a non-convex optimization problem. The second problem is convex and is the ”standard” support vector classifier.



The case for data which is not fully linearly separable.
Misclassification occurs when $\xi_i > 1$.

- Summarizing we arrive at the following minimization problem (again choosing $\| \beta \| = M^{-1}$) for the **support vector classifier**.

$$\min \| \beta \| \text{ subject to } \begin{cases} y_i(x_i^\top \beta + \beta_0) \geq (1 - \xi_i), \forall i \\ \xi_i \geq 0, \sum \xi_i \leq K. \end{cases}$$

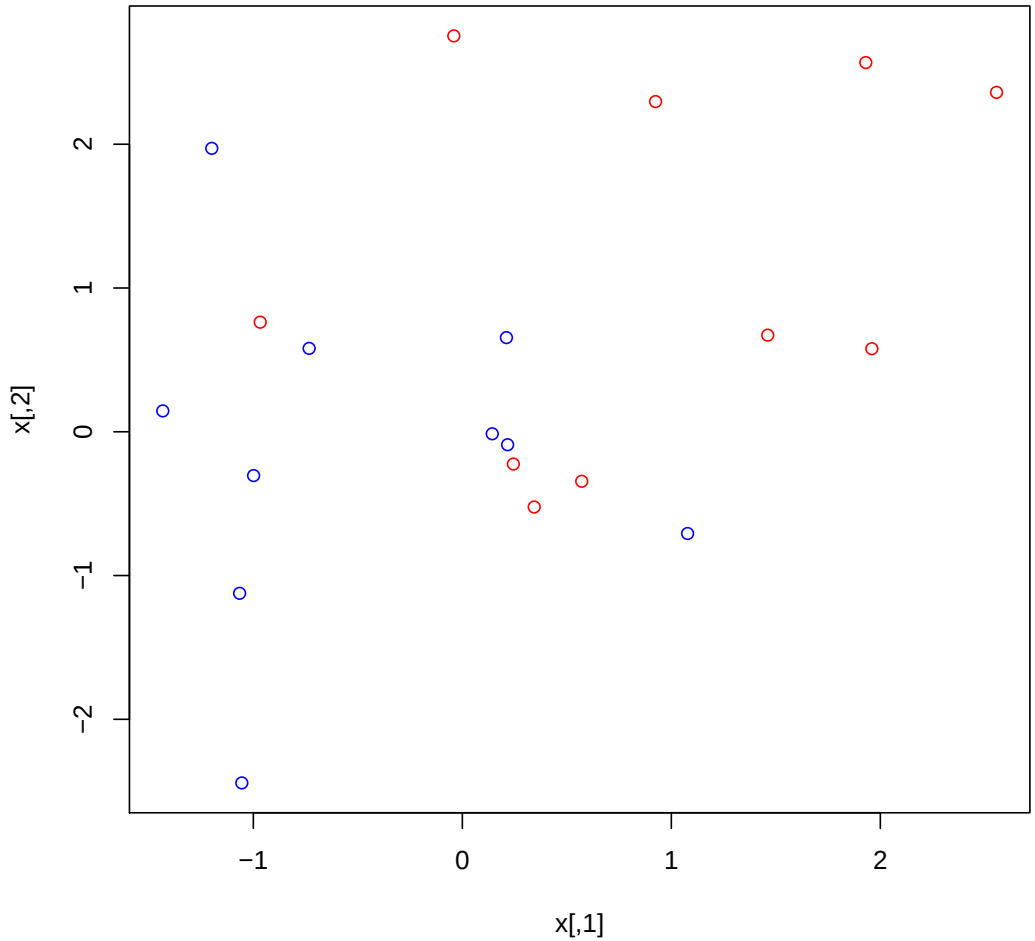
- For a detailed description we will revisit the Properties from Jungnickel (2014) in the following. Beforehand we illustrate the SVM with a small R example.

A toy example

The classical statistical example is to consider two groups of normal distributions with different mean.

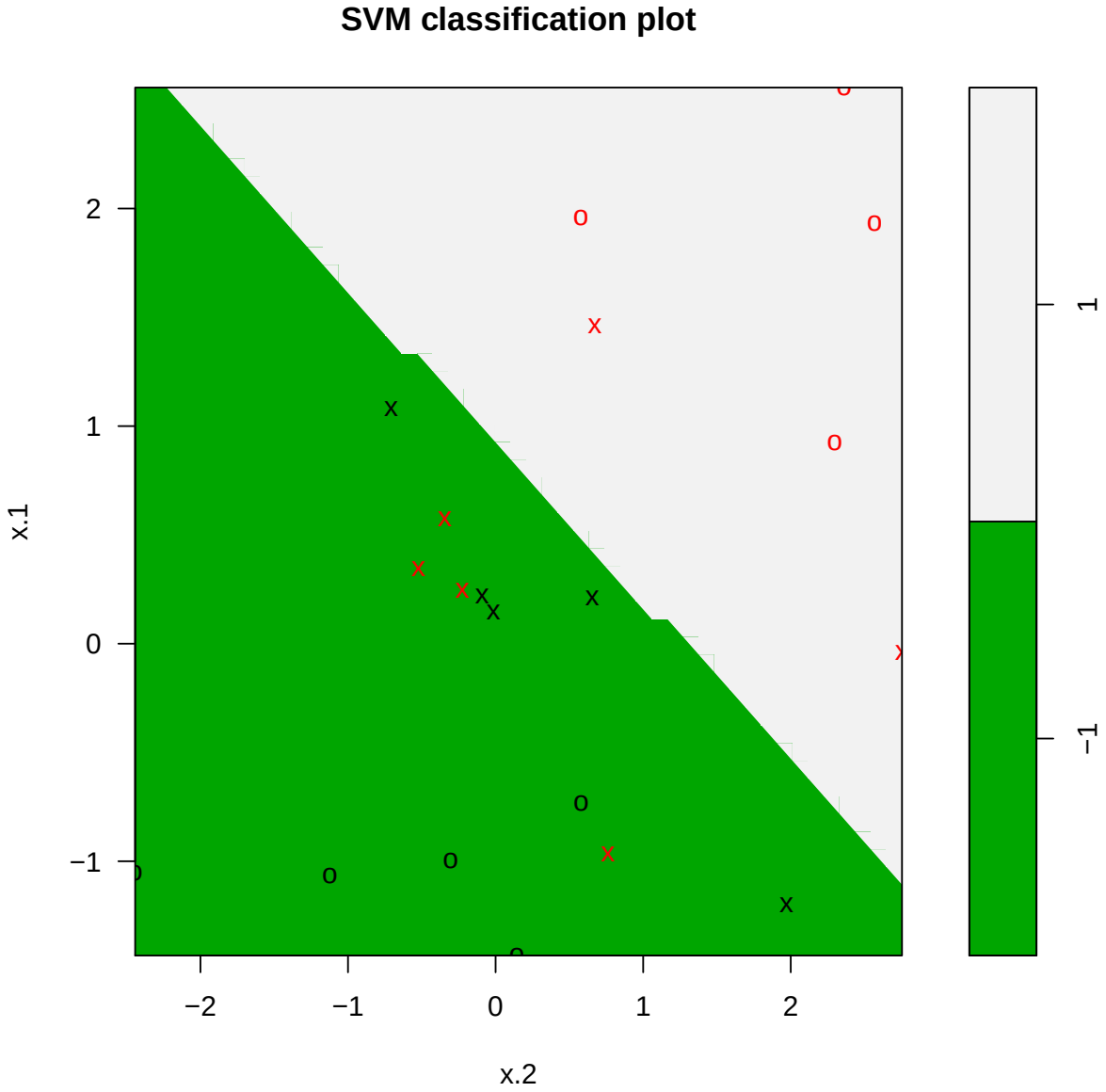
- Consider (X_i, Y_i) which are standard normal in group one and normal with mean $(1, 1)$ but same (identity) covariance matrix.
- The example is taken from Chapter 9 of James et al. (2013)

```
library(e1071)
%  N=20
%  x=matrix(rnorm(N*2), ncol=2)   # data - x corresponds to 2-dimensional data
%  y=c(rep(-1,N/2), rep(1,N/2))      # data - y has the classifier
%  x[y==1,]=x[y==1,] + 1           # We shift the mean for the second class by (1,1)
%  plot(x, col=(3-y))
```

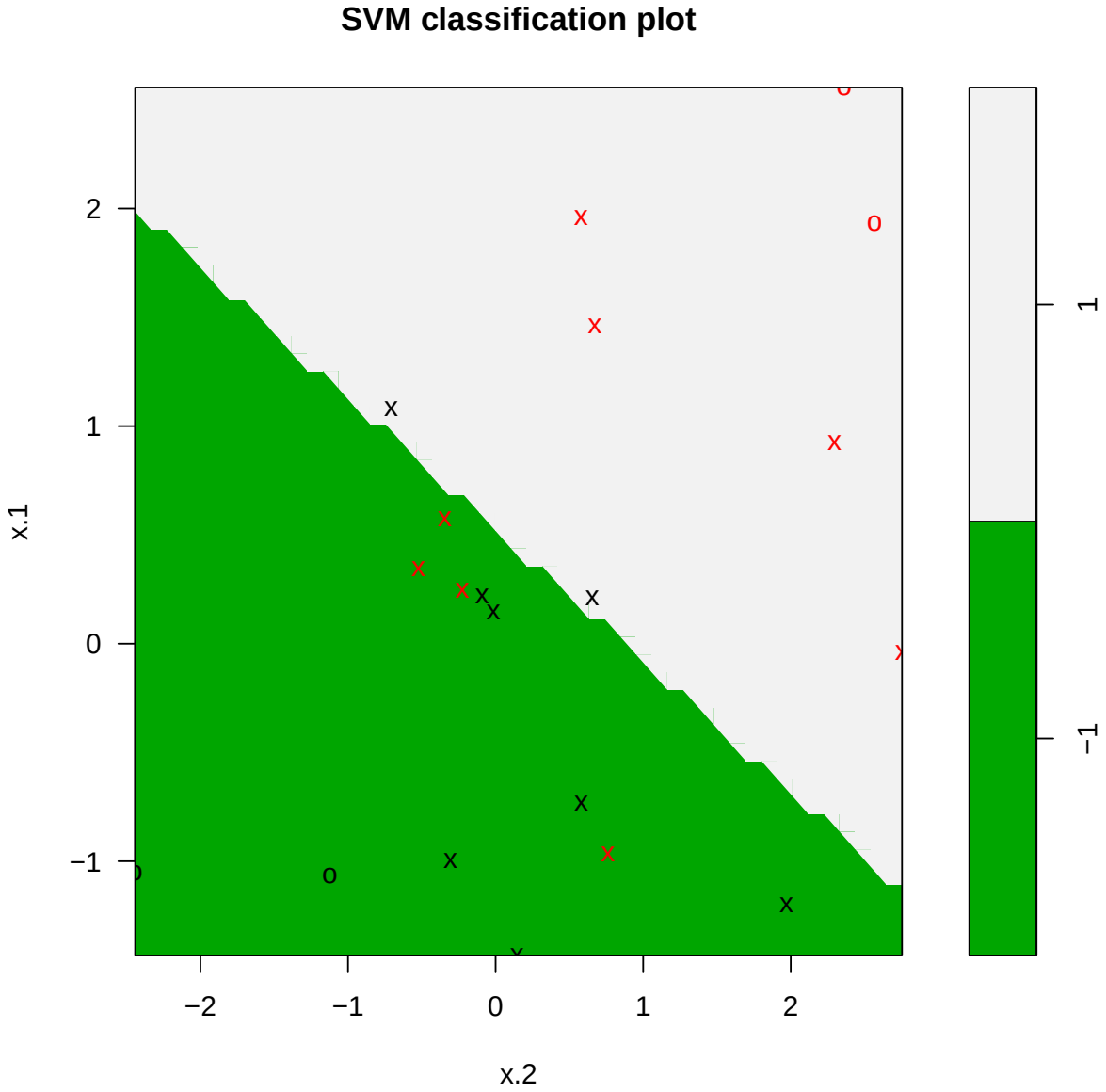


```
dat = data.frame(x=x, y=as.factor(y))
svmfit = svm( y ~ ., data=dat, kernel="linear", cost=10, scale=FALSE)

svmfit$index         # Identities of the support vectors
summary(svmfit)      # Summary
plot(svmfit,dat,color = terrain.colors) # Plot
```

- This small code fits a linear SVM with cost factor 10 to the data.
- Its output is the number and identities of the support vectors (unfortunately not the classification rule)



The summary plot: red / black decode the classes and the crosses are the support vectors. Note that we have four misclassifications.



A smaller parameter C leads to more misclassfications. What could be an optimal choice ?

The library provides a connection to cross-validation for the choice of parameters.

```
tune.out=tune(svm,y ~.,data=dat,kernel="linear",ranges=list(
      cost=c(0.001, 0.01, 0.1, 1,5,10,100)))
summary(tune.out)

#Parameter tuning of ‘svm’:
#- sampling method: 10-fold cross validation
#- best parameters:
# cost
# 0.1
#- Detailed performance results:
#   cost error dispersion
#1 1e-03  0.65  0.3374743
#2 1e-02  0.65  0.3374743
#3 1e-01  0.05  0.1581139
#4 1e+00  0.10  0.2108185
#5 5e+00  0.15  0.2415229
#6 1e+01  0.15  0.2415229
#7 1e+02  0.15  0.2415229
```

The kernel trick

- It seems quite restrictive to consider only linear classification rules. We have already seen that in linear regression we were able to overcome this problem by a suitable transformation of the data. This can also be achieved here and is often called **the kernel trick**.
- We first give a rather informative introduction and thereafter discuss the mathematical properties.

A short excursion to convex optimization

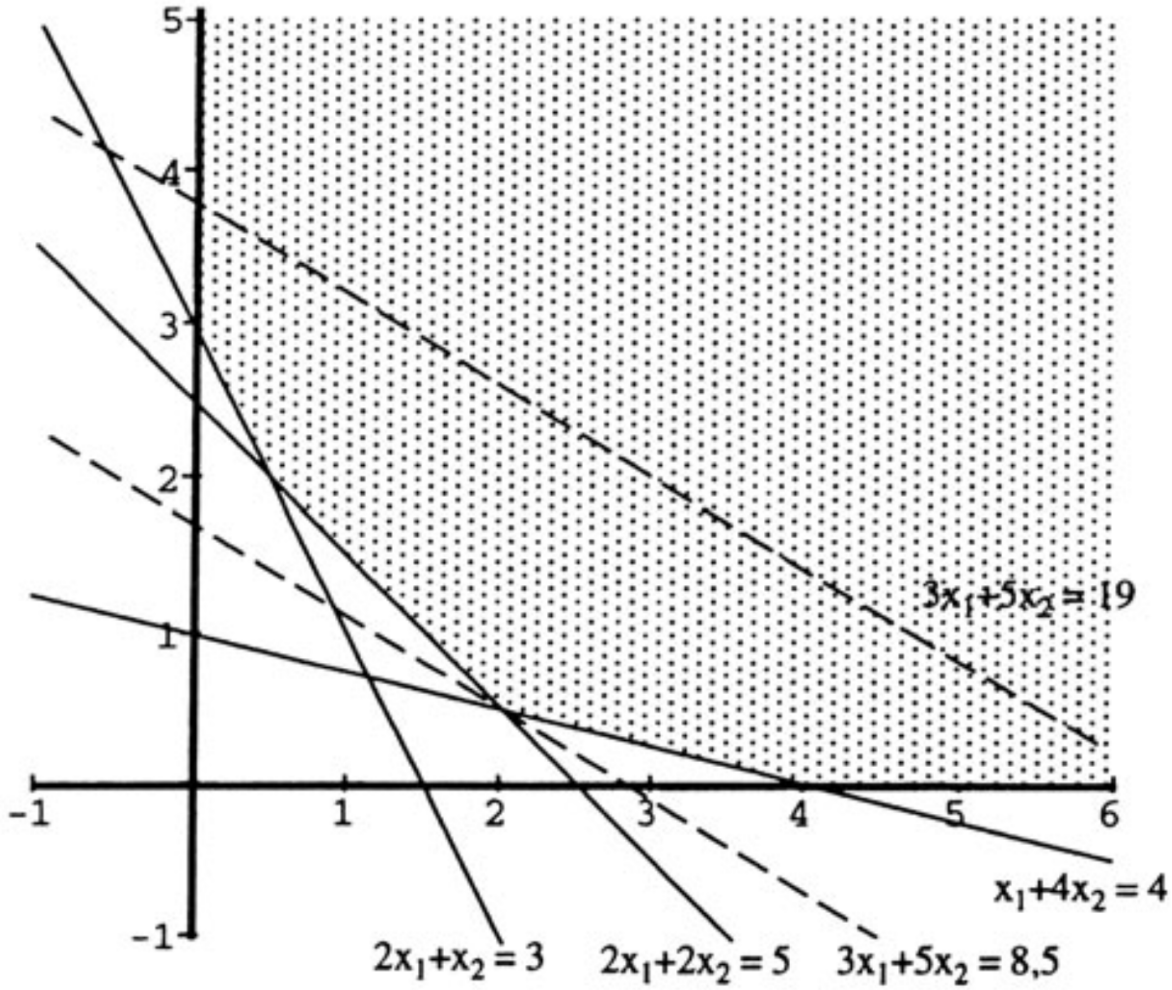
We follow Jungnickel[2]. Consider the **linear optimization problem**, also called **linear programm**

$$\begin{aligned} \min \quad & 3x_1 + 5x_2 \\ \text{subject to} \quad & 2x_1 + x_2 \geq 3 && (1) \\ & 2x_1 + 2x_2 \geq 5 && (2) \\ & x_1 + 4x_2 \geq 4 && (3) \\ & x_1 \geq 0 && (4) \\ & x_2 \geq 0 && (5) \end{aligned}$$

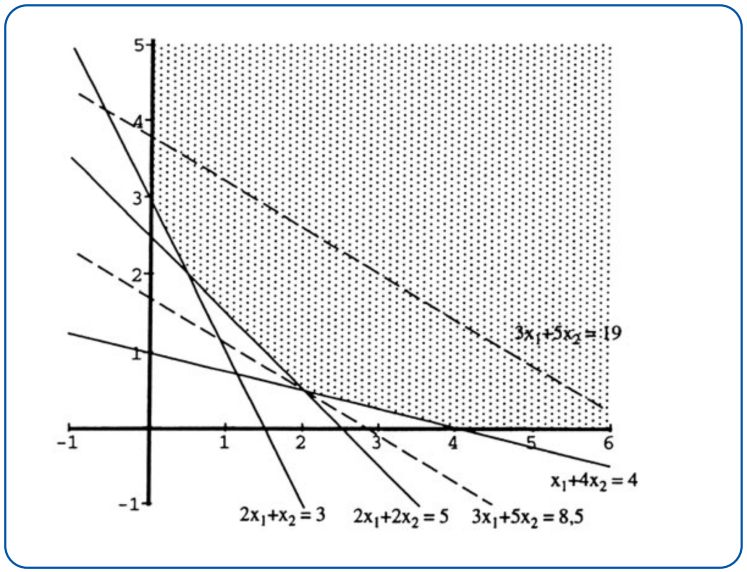
We illustrate the setting in the following picture. Besides this, we plot the target function.

$$3x_1 + 5x_2 = 19.$$

[2] D. Jungnickel (2014). **Optimierungsmethoden: Eine Einführung**. Springer-Verlag.



The optimal solution can be found by moving $3x_1 + 5x_2 = 19$ towards the origin $(0,0)$.



- We realize that the optimal solution will be on one of the intersection points (this important observation leads to the **Simplex algorithm**)
- The optimal solution is $x^* = (2, 0.5)$ representing $3x_1 + 5x_2 = 8.5$
- As we obtained this geometrically, are we able to prove that this solution is optimal ?

$$
\begin{aligned}
\min \quad & 3x_1 + 5x_2 & \\
\text{subject to} \quad & 2x_1 + x_2 \geq 3 & (1)\\
& 2x_1 + 2x_2 \geq 5 & (2)\\
& x_1 + 4x_2 \geq 4 & (3)\\
& x_1 \geq 0 & (4)\\
& x_2 \geq 0 & (5)
\end{aligned}
$$

- Observe, that such an equation might not always have a finite solution (eg. if one of the coefficients is negative)
- Any of the boundary conditions can be multiplied with positive constants. The boundary conditions can also be added. We could, for example add (1) to $1.5 \cdot$(2) and obtain

$$3x_1 + 3x_2 \geq 7.5$$

- As, moreover $x_2 \geq 0$, we obtain necessarily $3x_1 + 5x_2 \geq 7.5$
- Can we do better ? Are we able to reach, e.g. 8.5 ??

The dual problem

$$
\begin{aligned}
\min \quad & 3x_1 + 5x_2 & \\
\text{subject to} \quad & 2x_1 + x_2 \geq 3 & (1)\\
& 2x_1 + 2x_2 \geq 5 & (2)\\
& x_1 + 4x_2 \geq 4 & (3)\\
& x_1 \geq 0 & (4)\\
& x_2 \geq 0 & (5)
\end{aligned}
$$

- Multiplying the inequalities with y_i, we arrive at the following scheme: let us find multipliers $y_1,\ y_2,\ y_3 \geq 0$, such that

$$
\begin{aligned}
2y_1 + 2y_2 + y_3 &\leq 3\\
y_1 + 2y_2 + 4y_2 &\leq 5\\
y_1, y_2, y_3 &\geq 0.
\end{aligned}
$$

- The r.h.s. should be **maximized**, to be as close to x^* as possible, such that the objective function is

$$
\max \quad 3y_1 + 5y_2 + 4y_3.
$$

- This is the so-called dual problem. We can easily verify that $x^* = (2, 0.5)$ verifies the dual problem, hence is optimal.

Arguing similar, but more general we arrive at the following result.

Proposition (Weak duality)

Consider $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$ and the two problems

$$\min c^\top x, \quad \text{subject to} \quad Ax \geq b, x \geq 0 \tag{P}$$

$$\max y^\top b, \quad \text{subject to} \quad y^\top A \leq c^\top, y \geq 0. \tag{D}$$

Then, for any x_0 satisfying the constraints in (P) and y_0 the ones in (D) it holds that

$$y_0^\top b \leq c^\top x_0.$$

The proof is fairly easy:

$$y_0^\top b \leq y_0^\top (Ax_0) = (y_0^\top A)x_0 \leq c^\top x_0.$$

It can even be shown that in this case $y^{*\top} = c^\top x^*$, see Satz 2.9.17 in Jungnickel (2014).

Convex optimization

Actually we are interested in convex optimization problems. More precisely, we consider the following **minimization problem (MP)**

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to} \quad & g(x) \leq 0 \\ & h(x) = 0, \end{aligned} \tag{MP}$$

where $x \in X \subset \mathbb{R}^n$ and $g : X \to \mathbb{R}^m$, $h : X \to \mathbb{R}^p$. We denote by S the set where the constraints in (MP) are satisfied. Define the **Lagrange function**

$$\ell(x, \lambda, \mu) := f(x) + \lambda^\top h(x) + \mu^\top g(x).$$

Define the **dual function** $\theta(\lambda, \mu) := \inf\{\ell(x, \lambda, \mu) : x \in X\}$. Then the **dual problem** to (MP) is

$$\begin{aligned} \max \quad & \theta(\lambda, \mu) \\ \text{subject to} \quad & \mu \geq 0. \end{aligned} \tag{DP}$$

Again, we have strong duality under additional assumptions, see Satz 6.5.10 in Jungnickel (2014).

Proposition (Strong duality)

Let X be a convex set and $f : X \to \mathbb{R}$, $g : X \to \mathbb{R}^p$ be convex functions and $h : X \to \mathbb{R}^m$ be an affine function.
If there exists $x_0 \in X$, s.t. $g(x_0) < 0$, $h(x_0) = 0$ and $0 \in \int h(X)$, then

$$\sup\{\theta(\lambda, \mu) : \mu \geq 0\} = \inf\{f(x) : x \in S\}.$$

The following necessary conditions are found in Satz 2.9.9. in Jungnickel (2014).

It is also very useful to obtain necessary conditions, however in the simpler optimization problem where we only have inequality constraints. Set $I(x) = \{i : g_i(x) = 0\}$. We consider

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to} \quad & g(x) \leq 0, \quad x \in X. \end{aligned} \tag{MPIC}$$

Proposition (Karush-Kuhn-Tucker conditions)

Let X be open and x an admissible point and g_i differentiable in x for all $i \in I(x)$ and continuous for $i \notin I(x)$. Moreover, let

$$\nabla g_i(x), i \in I(x) \text{ be linearly independent.} \tag{LICQ}$$

If x is a local minimum, then there exist μ_i, $i \in I(x)$, s.t.

$$\nabla f(x) + \sum_{i \in I(X)} \mu_i \nabla g_i(x) = 0.$$

If g_i are differentiable for all i, we obtain

$$\nabla f(x) + \sum_i \mu_i \nabla g_i(x) = 0,$$

$$\mu_i \geq 0, \text{ and } \mu_i g_i(x) = 0, \quad i = 1, \ldots, p.$$

Back to the kernel trick

We are interested in the following convex problem:

$$\begin{aligned} \min_{\beta,\beta_0,\xi} \quad & \frac{1}{2} \parallel \beta \parallel^2 + C \sum_{i=1}^{N} \xi_i \\ \text{subject to} \quad & y_i(x_i^\top \beta + \beta_0) \geq (1-\xi_i), \forall i \\ & \xi_i \geq 0. \end{aligned} \tag{MP}$$

The Lagrange function is

$$\frac{1}{2} \parallel \beta \parallel^2 + C \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i (y_i(x_i^\top \beta + \beta_0) - (1-\xi_i)) - \sum_{i=1}^{N} \mu_i \xi_i$$

with positivity constraints $\alpha, \mu, \xi \geq 0$.

The Lagrange function is

$$\frac{1}{2} \parallel \beta \parallel^2 + C \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i (y_i (x_i^\top \beta + \beta_0) - (1 - \xi_i)) - \sum_{i=1}^{N} \mu_i \xi_i.$$

We see that this function is differentiable in β, β_0 and ξ. Setting the respective derivatives to zero, we obtain

$$\beta = \sum_{i=1}^{N} \alpha_i y_i x_i, \quad 0 = \sum_{i=1}^{N} \alpha_i y_i, \quad \alpha_i = C - \mu_i. \tag{4}$$

Inserting this into the Lagrangian we obtain the Wolfe dual objective function

$$\ell_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^\top x_j,$$

giving a lower bound of the objective function of (MP).

$$\nabla f(x) + \sum_i \mu_i \nabla g_i(x) = 0,$$

$$\mu_i \geq 0, \text{ and } \mu_i g_i(x) = 0, \quad i = 1, \ldots, p,$$

x admissible.

In addition, we obtain the conditions

$$\begin{aligned} \alpha_i \left(y_i (x_i^\top \beta + \beta_0) - (1 - \xi_i) \right) &= 0, \\ \mu_i \xi_i &= 0, \\ y_i (x_i^\top \beta + \beta_0) - (1 - \xi_i) &\geq 0. \end{aligned}$$

- We can also make a further observation: from (4), we see that the solution for β has the form

$$\beta^* = \sum_{i=1}^{N} \alpha_i^* y_i x_i,$$

with nonzero coefficients α_i^* for those observations i which lie on the margin (the **support vectors**).

- β_0^* can be computed by the average of all solutions and we arrive at the decision function

$$\hat{G}(x) = \text{sign}[x^\top \beta^* + \beta_0^*].$$

- An important observation is that the optimization problem now only depends on the scalar product

$$x_i^\top x_j$$

which opens the scenery for reproducing kernel Hilbert spaces.

Reproducing kernel Hilbert spaces

For details on this subject, consider C. Berg, J. P. R. Christensen und P. Ressel (1984). **Harmonic analysis on semigroups**. Springer-Verlag.

Definition

Let X be a nonemtpy set. A function $K : X \times X \to \mathbb{C}$ is called a **kernel**, if

$$\sum_{i,j=1}^{n} c_i \bar{c}_j K(x_i, x_j) \geq 0$$

for all $n \in \mathbb{N}$, $\{x_1, \ldots, x_n\} \subset X$ and $\{c_1, \ldots, c_n\} \subset \mathbb{C}$. If we replace $\mathbb{C}$ by $\mathbb{R}$, we call the kernel real-valued.

There are also negative definite kernels (which is not of interest here), and more precisely one calls the above kernel a positive definite kernel.

Now consider such a kernel K and the linear subspace $H_0 \subset \mathbb{C}^X$ generated by the functions

$$\{K_x : x \in X\},$$

where $K_x(y) = K(x, y)$.

We introduce a (well-defined) scalar product for $f = \sum a_i K_{x_i}$, $g = \sum b_j K_{x_j}$ by

$$\langle f, g \rangle := \sum_{i,j} a_i \bar{b}_j K(x_i, x_j).$$

Then, H_0 is a pre-Hilbert space and its completion H is a Hilbert space.

Most importantly, the scalar product has the **reproducing property**

$$\langle f, K_x \rangle = \sum_i a_i K(x_i, x) = f(x)$$

for all $f \in H_0$ and $x \in X$. This is why the space H is called **reproducing kernel Hilbert space**.

Mercer's theorem allows to obtain a different view on the kernel K. Associate the integral operator

$$T_K f := \int_X K(.,x) f(x) d\nu(x)$$

to the kernel K (in $L^2(X)$). The the spectracl decomposition of T_K yields

$$K(x_1,x_2) = \sum_n \lambda_n \phi_n(x_1) \bar{\phi}_n(x_2). \tag{5}$$

The RKHS is given by

$$H = \{f \in L_2(X) : \sum_n \lambda_n^{-1} \langle f, \phi_n \rangle^2 < \infty\}.$$

This yields in turn the representation

$$f(x) = \sum_n a_n \phi_n(x).$$

On the other side, any kernel satisfying (5) is non-negative definite and thus leads to a RKHS.

Summarizing, we obtain the following:

- A RKHS allows a representation

$$f(x) = \sum_n a_n \phi_n(x)$$

with an associated kernel K satisfying

$$K(x_1, x_2) = \sum_n \lambda_n \phi_n(x_1) \bar{\phi}_n(x_2).$$

- One can think of transforming x to $\phi_n(x)$ such that

$$\{(\phi_n(x))_{n=1}^{\infty} : x \in X\}$$

is called the feature space. This can in principle be an infinite dimensional space.

Kernel examples

From T. Evgeniou, M. Pontil und T. Poggio (2000). „Regularization networks and support vector machines“. In: **Advances in computational mathematics** 13.1, S. 1–50 we cite the following list of kernels:

Table 1

Some possible kernel functions. The first four are radial kernels. The multiquadric and thin plate splines are positive semidefinite and thus require an extension of the simple RKHS theory of this paper. The last three kernels were proposed by Vapnik [96], originally for SVM. The last two kernels are one-dimensional: multidimensional kernels can be built by tensor products of one-dimensional ones. The functions B_n are piecewise polynomials of degree n, whose exact definition can be found in [85].

Kernel function	Regularization Network
$K(\mathbf{x}-\mathbf{y}) = \exp(-\|\mathbf{x}-\mathbf{y}\|^2)$	Gaussian RBF
$K(\mathbf{x}-\mathbf{y}) = (\|\mathbf{x}-\mathbf{y}\|^2 + c^2)^{-1/2}$	Inverse multiquadric
$K(\mathbf{x}-\mathbf{y}) = (\|\mathbf{x}-\mathbf{y}\|^2 + c^2)^{1/2}$	Multiquadric
$K(\mathbf{x}-\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|^{2n+1}$	Thin plate splines
$K(\mathbf{x}-\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|^{2n} \ln(\|\mathbf{x}-\mathbf{y}\|)$	
$K(\mathbf{x},\mathbf{y}) = \tanh(\mathbf{x}\cdot\mathbf{y}-\theta)$	(only for some values of θ) Multi-Layer Perceptron
$K(\mathbf{x},\mathbf{y}) = (1+\mathbf{x}\cdot\mathbf{y})^d$	Polynomial of degree d
$K(x,y) = B_{2n+1}(x-y)$	B-splines
$K(x,y) = \dfrac{\sin(d+1/2)(x-y)}{\sin((x-y)/2)}$	Trigonometric polynomial of degree d

- The choice of the kernel should be adapted to the question and influences the performance massively !
- Most important are probably: Gaussian (RBF), polynomial and splines. But there are many other choices, for example in speech recognition (where one uses **string kernels**...)

MNIST dataset with SVM

Lets look at an example: our aim is to classify handwritten digits with SVM.

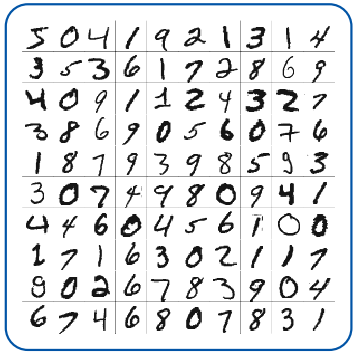
- We will use (parts of) the MNIST dataset, available at `http://yann.lecun.com/exdb/mnist/`.
- This includes 60.000 images of handwritten digits (0-9) with classification. ($\rightarrow$ wich learning ?)
- Lets get an impression of the first one hundred pictures:

First 100 digits from the MNIST dataset.

```
# Taken from https://gist.github.com/brendano/39760
# Copyright under the MIT licens:
# I hereby license it as follows. This is the MIT license.
# Copyright 2008, Brendan O'Connor
# Permission is hereby granted, free of charge, to any person obtaining a copy of this softwar
# and associated documentation files (the "Software"), to deal in the Software without restric
# including without limitation the rights to use, copy, modify, merge, publish, distribute,
# sublicense, and/or sell copies of the Software, and to permit persons to whom the Software i
# furnished to do so, subject to the following conditions: The above copyright notice and this
# permission notice shall be included in all copies or substantial portions of the Software.
```

```
library(e1071)

load_mnist <- function() {
  load_image_file <- function(filename) {
    ret = list()
    f = file(filename,'rb')
    readBin(f,'integer',n=1,size=4,endian='big')
    ret$n = readBin(f,'integer',n=1,size=4,endian='big')
    nrow = readBin(f,'integer',n=1,size=4,endian='big')
    ncol = readBin(f,'integer',n=1,size=4,endian='big')
    x = readBin(f,'integer',n=ret$n*nrow*ncol,size=1,signed=F)
    ret$x = matrix(x, ncol=nrow*ncol, byrow=T)
    close(f)
    ret
  }
  load_label_file <- function(filename) {
    f = file(filename,'rb')
    readBin(f,'integer',n=1,size=4,endian='big')
    n = readBin(f,'integer',n=1,size=4,endian='big')
    y = readBin(f,'integer',n=n,size=1,signed=F)
    close(f)
    y
  }
  train <<- load_image_file('train-images-idx3-ubyte')     # Images
  test <<- load_image_file('t10k-images-idx3-ubyte')

  train$y <<- load_label_file('train-labels-idx1-ubyte')   # Labels
  test$y <<- load_label_file('t10k-labels-idx1-ubyte')  }

show_digit <- function(arr784, col=gray(12:1/12), ...) {
  image(matrix(arr784, nrow=28)[,28:1], col=col, ...)}
```

```
setwd("somepath/MNIST")
load_mnist()

# We plot the first 100 images
par(mfrow=c(10,10),mai=c(0,0,0,0))
for (i in 1:100) {
  show_digit(train$x[i,],col=gray(12:1/12),xaxt="n", yaxt="n")
}
```



```
N=2000
dat=data.frame(x=train$x[1:N,],y=as.factor(train$y[1:N]))
svmfit = svm (y ~., data=dat,   method="class", kernel="linear", cost=10, scale=FALSE)
```

- This is the SVM-routine. Note that we chose linear and cost factor $C = 10$.
- Additionally, we only use 2000 samples for training (otherwise it takes quite long. 10.000 is feasiable...)

```
#prediction
testdat = data.frame(x=test$x[1:100,])
pred = predict(svmfit,testdat)

table(pred,test$y[1:100])
sprintf("Error rate: %f",1-sum(pred == test$y[1:100])/100)

# This gives the following result:


pred  0  1  2  3  4  5  6  7  8  9
   0  8  0  0  0  0  0  0  0  0  0
   1  0 14  0  0  0  0  0  0  0  0
   2  0  0  8  0  0  1  2  0  0  0
   3  0  0  0 11  0  0  0  0  0  0
   4  0  0  0  0 14  0  0  0  0  0
   5  0  0  0  0  0  6  0  0  0  0
   6  0  0  0  0  0  0  8  0  0  0
   7  0  0  0  0  0  0  0 14  0  0
   8  0  0  0  0  0  0  0  0  2  0
   9  0  0  0  0  0  0  0  1  0 11

"Error rate: 0.040000"
```

Which is already remarkable. Which of the images are misclassified ?

Regularization on RKHS

Classical regularization theory utilizes RKHS for the following minimization question:

$$\min_{f \in H} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \parallel f \parallel_H,$$