# MADS-ML – Machine Learning

## Support Vector Machines

Prof. Dr. Stephan Doerfel

FACHHOCHSCHULE KIEL
University of Applied Sciences

Moodle (WiSe 2024/25)

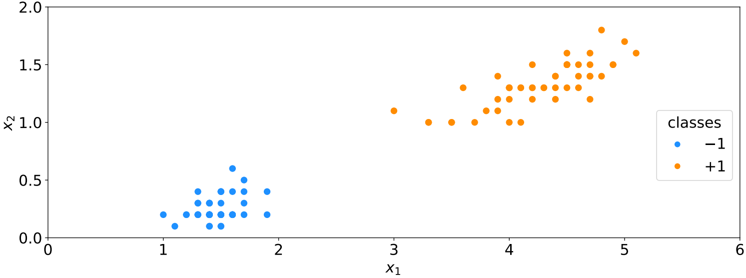
"The line must be drawn here!"

# Motivation

- Find an actual model and learn its parameters (in contrast to kNN an decision trees)
- use and build upon an intuitive concept of separability
- Support Vector Machines (SVMs) by Vapnik et al.
- basics 1963
- important extensions: 1993 Soft Margin, 1995 Kernel Trick
- versatile, robust, effective in high-dimensional spaces

# Outline

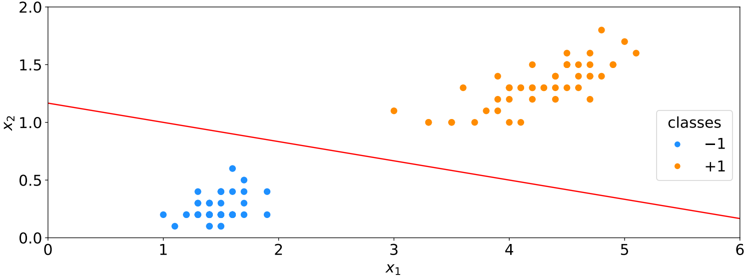
# Example Dataset 2D




Consider

- two classes: $c_1 = -1$ und $c_2 = +1$ ➔ (binary classification)
- two features: $x_1$ and $x_2$
- data shows part of the Iris dataset

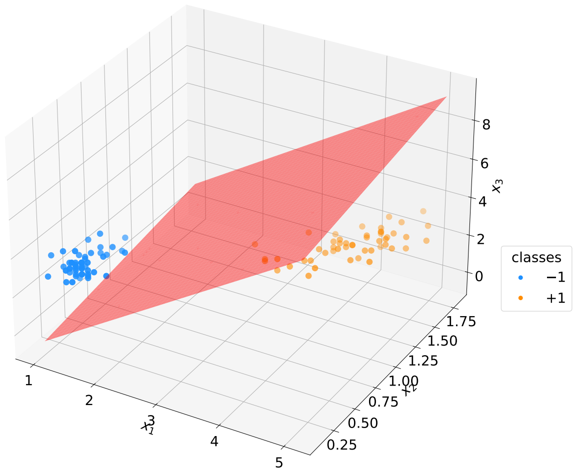
# Example Dataset 2D




Consider

- two classes: $c_1 = -1$ und $c_2 = +1$ ➔ (binary classification)
- two features: $x_1$ and $x_2$
- data shows part of the Iris dataset

# Example dataset 3D



# Outline

# Hyperplane

**Definition 1 (Hyperplane)**

A **Hyperplane** in an $n$-dimensional vector space $\mathbb{R}^n$ is an $(n-1)$-dimensional affine subspace.

- n=1 → point
- n=2 → line
- n=3 → (regular) plane

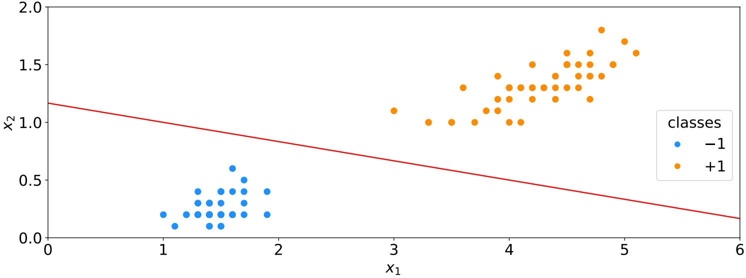
# Hyperplane – Mathematical Representation

A hyperplane in $\mathbb{R}^n$ is described by a vector $\boldsymbol{w} \in \mathbb{R}^n$ of (**weights**), $\boldsymbol{w} \neq 0$, a scalar $b \in \mathbb{R}$ (**bias**) and the equation:

$$\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0.$$

- denoted as $\mathcal{H}(\boldsymbol{w}, b)$
- $\langle \cdot, \cdot \rangle$ is the scalar product
- representation in coordinates::
  $w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b = 0$

# Hyperplane – Mathematical Representation – Example




Description of the hyperplane $\mathcal{H}(\boldsymbol{w}, b)$:

- $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0$ with $\boldsymbol{w} = \begin{pmatrix} 1 \\ 6 \end{pmatrix}, b = -7$

# Properties of a Hyperplane

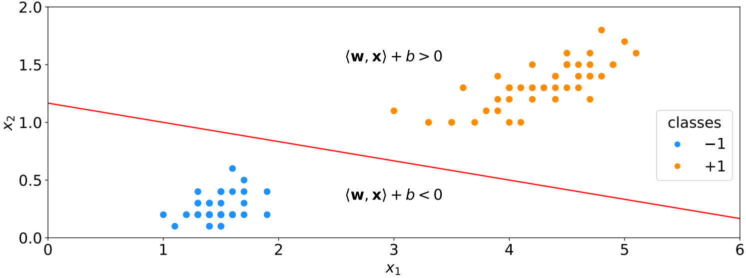
Let $\mathcal{H}(\boldsymbol{w}, b)$ be the hyperplane $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0$ and define

$$f_{\boldsymbol{w},b} : \mathbb{R}^n \to \mathbb{R} : \boldsymbol{x} \mapsto \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b.$$

Properties:

- $\mathcal{H}(\boldsymbol{w}, b)$ divides the space $\mathbb{R}^n$ into to parts (half-spaces).
- $f_{\boldsymbol{w},b}(\boldsymbol{x}) = 0 \iff \boldsymbol{x}$ in $\mathcal{H}(\boldsymbol{w}, b)$
- $\boldsymbol{x} \in \mathbb{R}^n$ with $f_{\boldsymbol{w},b}(\boldsymbol{x}) > 0$ is in the positive half-space
- $\boldsymbol{x} \in \mathbb{R}^n$ with $f_{\boldsymbol{w},b}(\boldsymbol{x}) < 0$ is in the negative half-space
- $|f_{\boldsymbol{w},b}(\boldsymbol{x})| > |f_{\boldsymbol{w},b}(\boldsymbol{y})|$ means, $\boldsymbol{x}$ is further from $\mathcal{H}(\boldsymbol{w}, b)$ than $\boldsymbol{y}$
- $|f_{\boldsymbol{w},b}(\boldsymbol{x})|$ is called functional distance from $\boldsymbol{x}$ to $\mathcal{H}(\boldsymbol{w}, b)$.

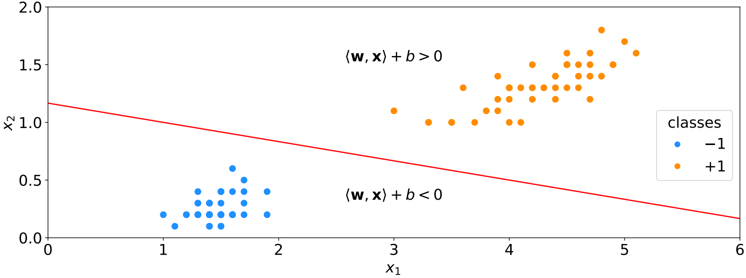
# Separating Hyperplane




$\mathcal{H}(\boldsymbol{w}, b)$ divides the space $\mathbb{R}^n$ into two parts (positive und negative).

$$\text{For } \boldsymbol{x} \in \mathbb{R}^n\text{:}\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b \begin{cases} = 0 \Rightarrow & \boldsymbol{x} \text{ in } \mathcal{H}(\boldsymbol{w}, b) \\ > 0 \Rightarrow & \boldsymbol{x} \text{ is in the positive half-space} \\ < 0 \Rightarrow & \boldsymbol{x} \text{ is in the negative half-space} \end{cases}$$

# Separating Hyperplane

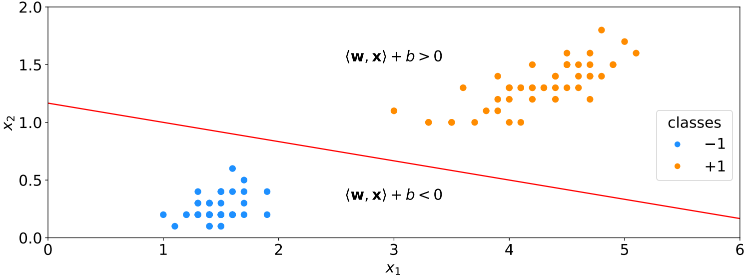



## Definition 2 (Separating Hyperplane)

For a dataset $D$ with two classes $(-1, +1)$, $\mathcal{H}(\boldsymbol{w}, b)$ is called a **separating hyperplane**, if for each instance $(\boldsymbol{x}, c) \in D$ holds

$$c = -1 \iff \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b < 0.$$

# Binary Classification using a Hyperplane




Approach:

- learning phase: determine separating hyperplane $\mathcal{H}(\boldsymbol{w}, b)$
- classification: $\hat{c}(\boldsymbol{x}) := \begin{cases} +1 \text{ for } \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b \geq 0 \\ -1 \text{ for } \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b < 0 \end{cases}$

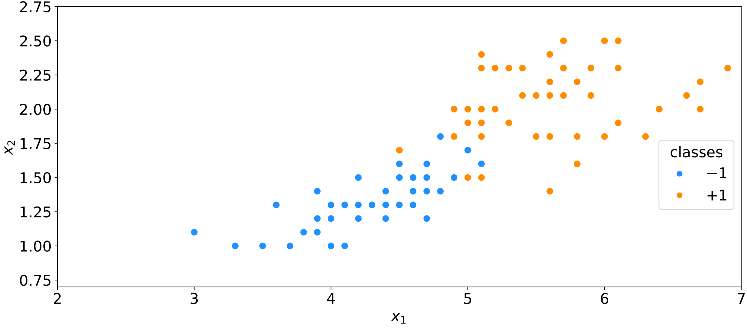
# Open Issues

1. **Existence:** Can we always find a separating hyperplane?

2. **Uniqueness:** Is the hyperplane unique or are there many?

3. **Optimality:** Which is the best hyperplane?

# Existence of a Separating Hyperplane




There are datasets for which no separating hyperplane exists.

# Linear Separable

**Definition 3 (Linear Separable)**

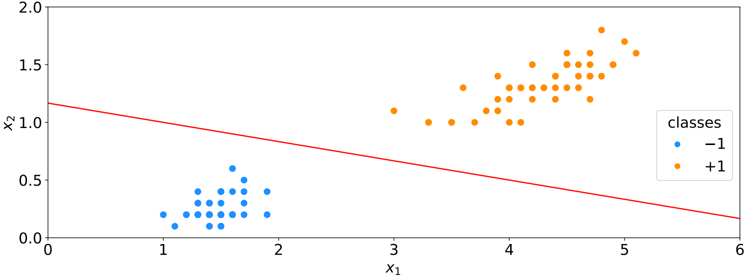
Two sets $A, B \subseteq \mathbb{R}^n$ are called **linear separable**, if there exist $\boldsymbol{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$, such that

- for all $\boldsymbol{x} \in A$ holds $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b \geq 0$ and
- for all $\boldsymbol{x} \in B$ holds $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b < 0$.

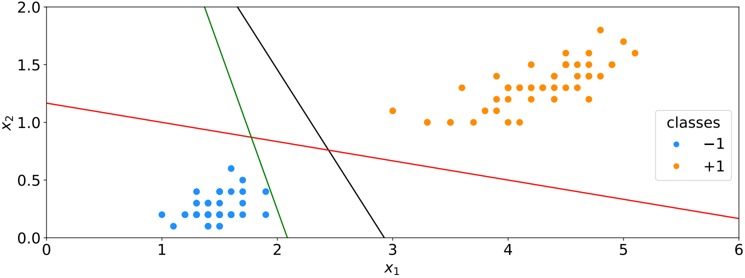
The requirement of linear separability is a strong restriction on the applicability of this approach.[1]

[1] Soft-Margin SVMs and the kernel trick are workarounds for this issue.

# Uniqueness of a Separating Hyperplane



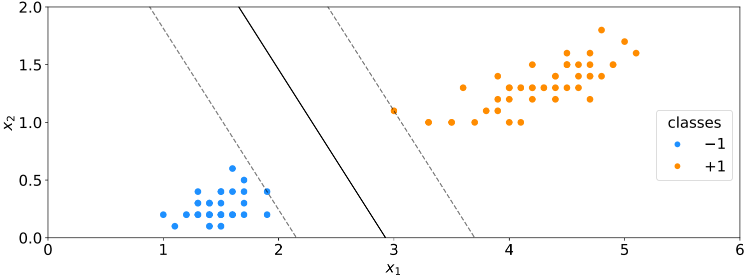
# Uniqueness of a Separating Hyperplane




If a dataset is linear separable, there can be many more hyperplanes.

Which hyperplane is the best?

# The Optimal Separating Hyperplane

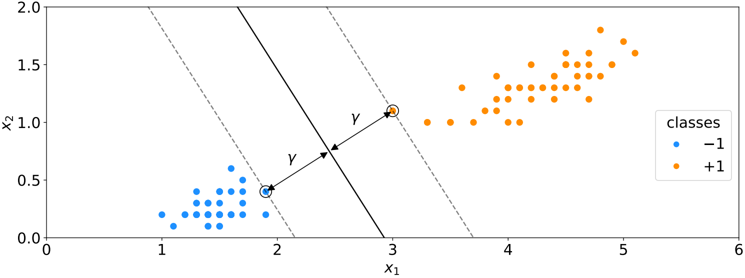



- **Generalization theory:** Estimate the risk of misclassification on unknown data

# Risk of misclassification – Maximum Margin

- Given a **probability of error** $\delta$, an upper bound $u$ for the classification error can be constructed such that with a probability of $(1 - \delta)$, no more than $u$ percent of the data are misclassified.
- This upper bound depends on
  - the radius of a ball encompassing the dataset (smaller is better),
  - the distance between data and hyperplane $\gamma$ (higher is better),
  - the number of training instances $\ell$ (higher is better) and
  - the probability of error $\delta$ (higher is better).
- **Result:** The higher the distance between hyperplane and data, the lower the risk.

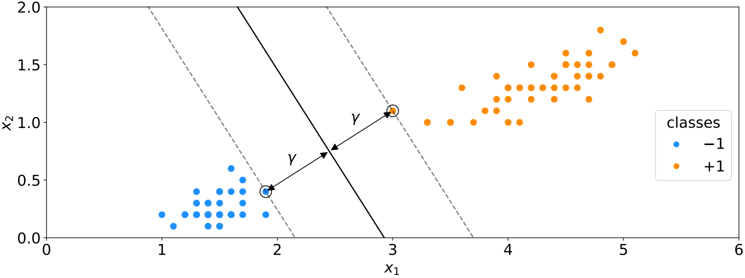
# The Optimal Separating Hyperplane




**Goal:** Determine the hyperplane $\mathcal{H}(\boldsymbol{w}, b)$, which has the highest possible distance $\gamma$ to the data.

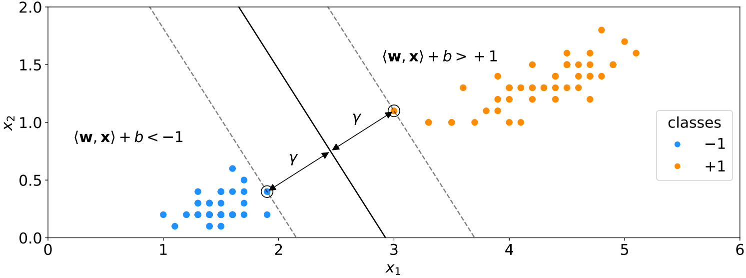
- Compute $\gamma$ from $\boldsymbol{w}$ and $b$
- Optimize $\gamma \to \max$

# The distance $\gamma$




- $\lambda \in \mathbb{R}, \lambda > 0 : \mathcal{H}(\lambda \boldsymbol{w}, \lambda b) = \mathcal{H}(\boldsymbol{w}, b)$
- If the dataset is linear separable, we can choose $\boldsymbol{w}$ and $b$ of the optimal hyperplane such that for each instance $\boldsymbol{x}$ with the minimal distance $\gamma$ holds: $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = \pm 1$.
- In this case: $\gamma = \frac{1}{\sqrt{\langle \boldsymbol{w}, \boldsymbol{w} \rangle}}$. ➔ $\gamma \to \max \iff \langle \boldsymbol{w}, \boldsymbol{w} \rangle \to min$

# Optimization task for the Maximum Distance Classifier

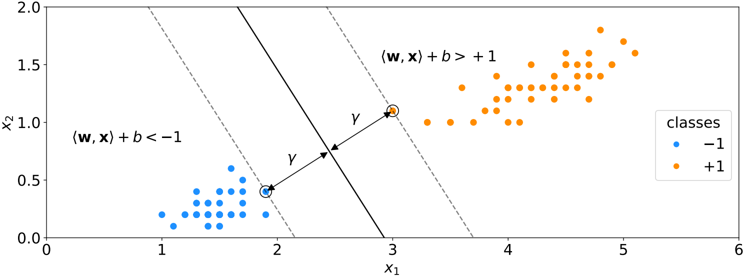



**Given:** linear separable dataset $\{(\boldsymbol{x}_1, c_1), (\boldsymbol{x}_2, c_2), \ldots, (\boldsymbol{x}_\ell, c_\ell)\}$

**Target:** Minimize $\langle \boldsymbol{w}, \boldsymbol{w} \rangle$

- for $c_i = +1 : \langle \boldsymbol{w}, \boldsymbol{x_i} \rangle + b \geq +1$
- for $c_i = -1 : \langle \boldsymbol{w}, \boldsymbol{x_i} \rangle + b \leq -1$

# Optimization task for the Maximum Distance Classifier




**Given:** linear separable dataset $\{(\boldsymbol{x}_1, c_1), (\boldsymbol{x}_2, c_2), \ldots, (\boldsymbol{x}_\ell, c_\ell)\}$

**Target:** Minimize $\langle \boldsymbol{w}, \boldsymbol{w} \rangle$

**Conditions:** $c_i(\langle \boldsymbol{w}, \boldsymbol{x_i} \rangle + b) - 1 \geq 0$ for $i = 1, ..., \ell$

# Solution of the Optimization Task

## Approach

- mathematical background: Lagrange-Theory, Karush-Kuhn-Tucker-Conditions
- problem is convex ➔ globally optimal solution
- Lagrange function:

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w} \rangle - \sum_{i=1}^{\ell} \alpha_i \left[c_i(\langle \boldsymbol{w}, \boldsymbol{x_i} \rangle + b) - 1\right] \rightarrow \min$$

  with $\alpha_i \geq 0$ for $1 \leq i \leq \ell$.
- The $\alpha_i$ are called **Lagrange-Multipliers**.
- Result: $\boldsymbol{w}^*, b^*, \boldsymbol{\alpha}^*$

# Observations 1/2

- $b$ can be computed as

$$b^* := -\frac{\max_{c_i=-1}(\langle \boldsymbol{w^*}, \boldsymbol{x_i} \rangle) + \min_{c_i=1}(\langle \boldsymbol{w^*}, \boldsymbol{x_i} \rangle)}{2}$$

- the distance $\gamma$ between hyperplane and data is

$$\gamma = \left( \sum_{i:\alpha_i \neq 0} \alpha_i^* \right)^{-\frac{1}{2}} .$$

# Observations 2/2

- In the solution, we yield

$$\alpha_i^*(c_i(\langle \boldsymbol{w^*}, \boldsymbol{x_i} \rangle + b^*) - 1) = 0 \text{ for } 1 \leq i \leq \ell,$$

  and thus $\alpha_i^* \neq 0 \Longrightarrow \langle \boldsymbol{w^*}, \boldsymbol{x_i} \rangle + b^* = \pm 1$ ($\boldsymbol{x_i}$ has minimum distance $\gamma$).
- The vectors $\boldsymbol{x_i}$ with $\alpha_i^* \neq 0$ are called **support vectors**.
- $\boldsymbol{w}^*$ is a linear combination of support vectors:

$$\boldsymbol{w}^* := \sum_{i=1}^{\ell} c_i \alpha_i^* \boldsymbol{x_i} = \sum_{i:\alpha_i^* \neq 0} c_i \alpha_i^* \boldsymbol{x_i}.$$

- usually, the number of support vectors is way smaller than the number of training instances $\ell$

# Classification

To classify an instance $\boldsymbol{x}$, we compute

$$\hat{c}(\boldsymbol{x}) := \begin{cases} +1 \text{ for } \langle \boldsymbol{w^*}, \boldsymbol{x} \rangle + b^* \geq 0 \\ -1 \text{ for } \langle \boldsymbol{w^*}, \boldsymbol{x} \rangle + b^* < 0 \end{cases}$$

$$\langle \boldsymbol{w^*}, \boldsymbol{x} \rangle = \langle \sum_{i:\alpha_i^* \neq 0} c_i \alpha_i^* \boldsymbol{x_i}, \boldsymbol{x} \rangle = \sum_{i:\alpha_i^* \neq 0} c_i \alpha_i^* \langle \boldsymbol{x_i}, \boldsymbol{x} \rangle$$

Thus, classification is computed directly by computing a couple of scalar products between the instance to classify and only those instances of the training data that are support vectors.

# Achievements (so far)

**Linear Classifier:**
SVMs learn parameters for a linear function

**Learning:**
parameters are learned as solution of a problem of quadratic programming

**Solution:**
direct solution is expansive (requires matrix inversion), efficient, specialized approximation heuristics work well

**Computational Complexity (Learning):**
$\mathcal{O}(n + \ell^3)$

**Classification:**
Very fast, as solution depends only on a couple on the scalar products with the support vectors

**Stability:**
Solution depends only on the support vectors ➔ very stable against perturbation of the training set

# Restriction (so far)

The so called **hard-margin SVM** is applicable only to linear separable datasets.

SVMs are binary classifiers.

Extensions:

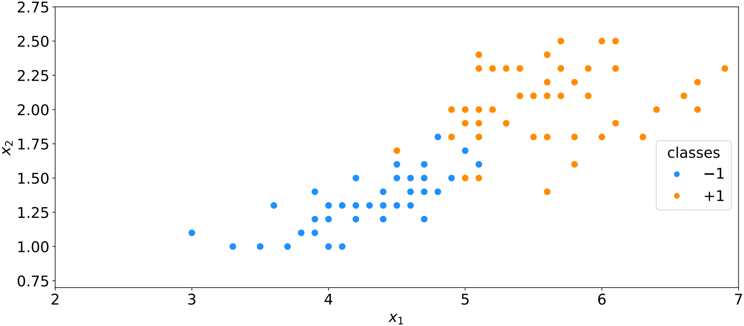
- **Soft-Margin** allows data points inside the $\gamma$-margin and even on the wrong side of the hyperplane
- **Kernel-Trick** maps the original data efficiently into a different space, where it is better separable
- **Multi-Class-SVMs** (later in this course)

Notebook 05_1_Iris_23, Cells 1–12

Exercises 1–3

# Outline

# The Maximum Distance Classifier




**Given:** dataset, not linear separable

**Idea:** modify the optimization problem

- to allow some digressions
- optimize a trade-off between digression and margin

# Soft-Margin Optimization

$$\frac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w} \rangle + C \sum_{i=1}^{\ell} \xi_i \to \min$$

with

$$c_i(\langle \boldsymbol{w}, \boldsymbol{x_i} \rangle + b) \geq 1 - \xi_i \text{ und } \xi_i \geq 0 \text{ for } i = 1, \ldots, \ell.$$

- the $\xi_i$ are called **slack variables**.
- $C$ is a hyperparameter, controlling the trade-off between margin and slack – between generalization and bias
  - higher $C$ punishes digression harder (thus lower misclassification probability)

# Soft-Margin – Lagrange Approach

Lagrange-Approach:

$$
\begin{aligned}
L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{r}) = \frac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w}\rangle &+ C\sum_{i=1}^{\ell}\xi_i \\
&- \sum_{i=1}^{\ell}\alpha_i\left[c_i(\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle + b) - 1 + \xi_i\right] \\
&- \sum_{i=1}^{\ell} r_i\xi_i \\
&\rightarrow \min
\end{aligned}
$$

with $\alpha_i \geq 0$ and $r_i \geq 0$ for $1 \leq i \leq \ell$.

# Soft-Margin Dual

In the solution of the optimization task, we yield $\boldsymbol{w}^* := \sum_{i=1}^{\ell} c_i \alpha_i^* \boldsymbol{x_i}$ and the margin

$$\gamma = \left( \sum_{i,j \in sv} c_i c_j \alpha_i^* \alpha_j^* \langle \boldsymbol{x_i}, \boldsymbol{x_j} \rangle \right)^{-\frac{1}{2}}$$

with the Box Constraint: $0 \leq \boldsymbol{\alpha} \leq \boldsymbol{C}$

# Soft-Margin KKT Conditions

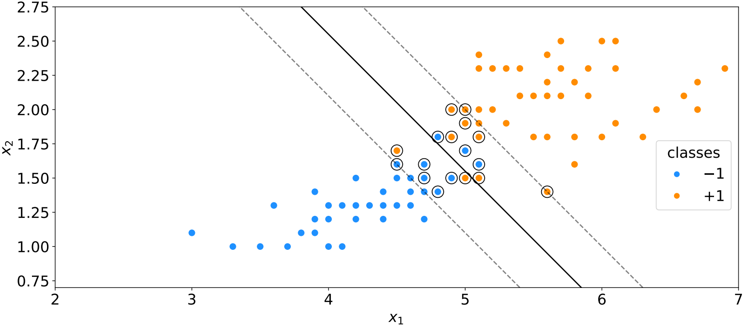
$$\alpha_i \left[ c_i(\langle \boldsymbol{w}, \boldsymbol{x_i} \rangle) - 1 + \xi_i \right] = 0, \quad i = 1, \dots, \ell,$$

$$\xi_i(\alpha_i - C) = 0, \quad i = 1, \dots, \ell,$$

Thus follows:

- $\xi_i \neq 0 \Rightarrow \alpha_i = C$ and $\boldsymbol{x_i}$ does not have the minimal distance $\frac{1}{\|w\|_2}$ in the correct half space
- $\alpha_i > 0 \Rightarrow$: $\boldsymbol{x_i}$ is a support vector
  - $0 < \alpha_i < C \Rightarrow \boldsymbol{x_i}$ is on the border of the margin
  - $\alpha_i = C \Rightarrow \boldsymbol{x_i}$ is either correctly classified but within the margin ($\xi_i < 1$), on the hyperplane ($\xi_i = 1$), or misclassified ($\xi_i > 1$)

# The Maximum Distance Classifier




In this example, we yield 17 support vectors, most of them within the margin.

Notebook 05_1_Iris_23, Cells 13–31

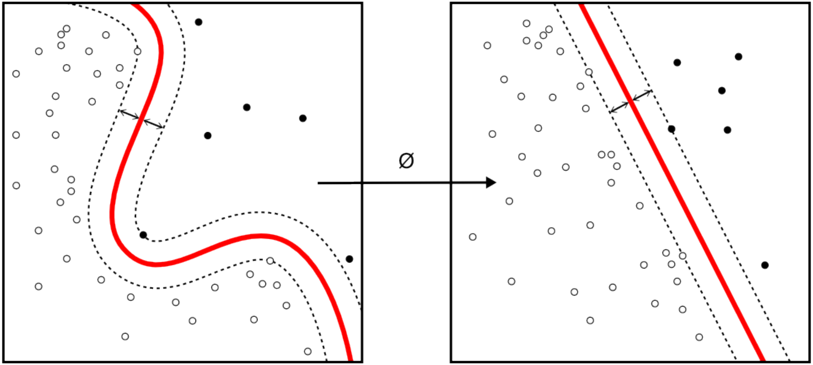
# Outline

# Embedding into higher-dimensional spaces

**Problem:** Datasets are rarely linear separable

**Idea:**

1. use a non-linear embedding into a higher-dimensional space
2. find the optimal separating hyperplane there
3. transform it back



# Embedding: Example

Given a non-separable dataset in $\mathbb{R}^3$.

1.) Choose embedding $\phi : \mathbb{R}^3 \to \mathbb{R}^6$ :

$$\begin{aligned} \phi_1(\boldsymbol{x}) &= x_1 & \phi_2(\boldsymbol{x}) &= x_2 & \phi_3(\boldsymbol{x}) &= x_3 \\ \phi_4(\boldsymbol{x}) &= x_1^2 & \phi_5(\boldsymbol{x}) &= x_1 x_2 & \phi_6(\boldsymbol{x}) &= x_1 x_3 \end{aligned}$$

2.) Find optimal hyperplane $\mathcal{H}(\boldsymbol{w}, b) = \langle \boldsymbol{w}, \boldsymbol{z} \rangle + b$ with $\boldsymbol{w}, \boldsymbol{b}, \boldsymbol{z} \in \mathbb{R}^6$!

3.) Retransformation into the original features space ($\mathbb{R}^3$) using $\boldsymbol{z} = \phi(\boldsymbol{x})$ :

$$\begin{aligned} & f_{\boldsymbol{w},b}(\boldsymbol{z}) = w_1 z_1 + w_2 z_2 + w_3 z_3 + w_4 z_4 + w_5 z_5 + w_6 z_6 + b \\ \Rightarrow\ & f(\boldsymbol{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_1^2 + w_5 x_1 x_2 + w_6 x_1 x_3 + b \end{aligned}$$

Note, that the classification function allows us to consider features combined, e.g. $x_1 x_2$.

# Discussion

**Potential:**

transforming the feature space might make the data better separable

**Issues:**

- What is a good transformation?
- The scalar product is very expensive in high-dimensional spaces (sometimes millions or even infinite dimensions)

**Observation:**

in the classifier and in the actual optimization, the training data occur only in scalar products, e.g.

$$f_{\boldsymbol{w},b}(\boldsymbol{x}) = \sum_{i:\alpha_i^* \neq 0} c_i \alpha_i^* \langle \boldsymbol{x_i}, \boldsymbol{x} \rangle + \boldsymbol{b}$$

- optimization and classification work with all kinds of scalar products

# Kernel Functions

**Idea:**

- Combine embedding into higher space and scalar product.
- Find functions $K$
  - that can be computed in the original space and
  - that work as the scalar product between the images of some embedding into some suitable higher-dimensional vector space
- $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \langle \phi(\boldsymbol{x_i}), \phi(\boldsymbol{x_j}) \rangle$.
- Such functions are called **kernel functions**.

**Win-Win:**

Better separability from the higher-dimensional space + cheap computability from the lower-dimensional (original) space

**SVMs:**

The full construction of the SVM classifier works, if regular scalar product is replaced by some kernel function.

# Classification Model

Regular classification function in input space $\mathcal{I}$

$$f_{\boldsymbol{w},b}(\boldsymbol{x}) = \sum_{i:\alpha_i^{\mathcal{I}*} \neq 0} c_i \alpha_i^{\mathcal{I}*} \langle \boldsymbol{x_i}, \boldsymbol{x} \rangle_{\mathcal{I}} + \boldsymbol{b}^{\mathcal{I}}$$

Transformation into the feature space $\mathcal{F}$

$$f_{\boldsymbol{w},b} \circ \phi(\boldsymbol{x}) = \sum_{i:\alpha_i^{\mathcal{F}*} \neq 0} c_i \alpha_i^{\mathcal{F}*} \langle \phi(\boldsymbol{x_i}), \phi(\boldsymbol{x}) \rangle_{\mathcal{F}} + \boldsymbol{b}^{\mathcal{F}}$$

Using a kernel function $K(x, y) \coloneqq \langle \phi(\boldsymbol{x_i}), \phi(\boldsymbol{x}) \rangle_{\mathcal{F}}$

$$f_{\boldsymbol{w},b} \circ \phi(\boldsymbol{x}) = \sum_{i:\alpha_i^{\mathcal{F}*} \neq 0} c_i \alpha_i^{\mathcal{F}*} K(\boldsymbol{x_i}, \boldsymbol{x}) + \boldsymbol{b}^{\mathcal{F}}$$

# Kernel Functions – Example

Kernel function $K : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R} : (\boldsymbol{x}, \boldsymbol{y}) \mapsto (\langle \boldsymbol{x}, \boldsymbol{y} \rangle_2 + 1)^2$

$$\begin{aligned}
K(\boldsymbol{x}, \boldsymbol{y}) &= (\langle \boldsymbol{x}, \boldsymbol{y} \rangle_2 + 1)^2 \\
&= (x_1 y_1 + x_2 y_2 + 1)^2 \\
&= x_1^2 y_1^2 + x_2^2 y_2^2 + 1 + 2 x_1 y_1 x_2 y_2 + 2 x_1 y_1 + 2 x_2 y_2 \\
&= \langle (x_1^2,\, x_2^2,\, 1,\, \sqrt{2} x_1 x_2,\, \sqrt{2} x_1,\, \sqrt{2} x_2)^T, \\
&\qquad (y_1^2,\, y_2^2,\, 1,\, \sqrt{2} y_1 y_2,\, \sqrt{2} y_1,\, \sqrt{2} y_2)^T \rangle_6
\end{aligned}$$

With $\phi : \mathbb{R}^2 \to \mathbb{R}^6 : x \mapsto (x_1^2,\, x_2^2,\, 1,\, \sqrt{2} x_1 x_2,\, \sqrt{2} x_1,\, \sqrt{2} x_2)^T$:

$$K(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle_6$$

Computation: scalar product in $\mathbb{R}^2$, sum, multiplication
Result: polynomial embedding, scalar product in $\mathbb{R}^6$

# Kernel Trick: discussion

**Achievement:**
Kernels make SVMs powerful and efficient

**Parameters:**
Kernels have their own parameters (see below) which become hyperparameters of the algorithm

**Construction:** 2 ways:

- Select $\phi$ by explicitly modeling the features in the feature space, derive $K$
- Select a function $K$ that is known to be a kernel function.
  - neither $\phi$ nor the feature space are considered explicitly
  - kernel property is verified mathematically (Mercer Theorem)
  - kernel functions can be found by combining known kernel functions

# Mercer Theorem – finite

**Theorem 4**

*Let $X$ be a finite subset of $\mathbb{R}^n$ and $K$ a symmetrical function on $X$. Then, $K$ is a kernel function if and only if matrix*

$$\left(K(\boldsymbol{x_i x_j})\right)_{i,j}^{n}$$

*is positive semidefinite (no negative eigenvalues).*

# Popular Kernels – The Polynomial Kernel

$$K(\boldsymbol{x}, \boldsymbol{y}) = (\gamma\langle \boldsymbol{x}, \boldsymbol{y}\rangle + r)^d$$

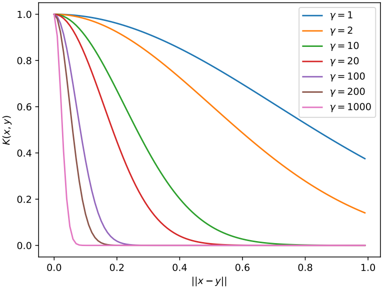
- here the feature space and its scalar product are explicit
- in the feature space, the original features occur combined (e.g. $(x_1 \cdot x_2)$)
- for logical values $(0, 1)$, feature combinations yield the logical "and" (e.g. co-occurrence in vector space models of texts)
- the feature space has many more dimensions than the original space ($n_{\mathcal{O}} << n_{\mathcal{F}}$)

| $n_{\mathcal{O}}$ | $d$ | $n_{\mathcal{F}}$ |
|---:|---:|---:|
| 2 | 2 | 6 |
| 2 | 3 | 10 |
| 10 | 2 | 66 |
| 1,000 | 2 | 501,501 |

# Popular Kernels – The Radial-Basis-Function Kernel

$$K(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\gamma \|\boldsymbol{x} - \boldsymbol{y}\|^2)$$

- a.k.a. Gaussian Kernel (for $\gamma = \frac{1}{2\sigma^2}$)
- models a similarity between $\boldsymbol{x}$ and $\boldsymbol{y}$
- $\gamma$ controls how strong similarity must be
- Danger: Overfitting for higher $\gamma$, related to kNN



Notebook 05_1_Iris_23, Cells 24–26

Exercises 4