* * * **Artificial Intelligence and Machine Learning** * * *

Support Vector Machines (SVMs) and Kernel Methods

Daniel Wehner, M.Sc.

SAP SE / DHBW Mannheim

Summer term 2025

Find all slides on GitHub (DaWe1992/Applied_ML_Fundamentals)

Lecture Overview

Agenda for this Unit

Section:

Linear Support Vector Machines

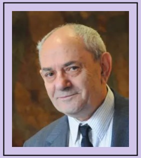
What is a Support Vector Machine (SVM)?

- A **Support Vector Machine (SVM)** is a **binary classifier** introduced by VAPNIK
- SVMs are one of the most studied models, being based on statistical learning frameworks of **VC theory** proposed by VAPNIK and CHERVONENKIS
- It is a **linear** classifier

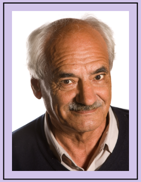
Yet another linear classifier? Unlike other linear classifiers (e. g. logistic regression), an SVM aims to maximize the margin, i. e. it maximizes the distance of the decision boundary to the closest data points from either of the classes **(maximum margin classifier)**. This results in a **unique solution** for hyperplanes, and in most cases allows for better **better generalization**.

Portrait: VAPNIK and CHERVONENKIS

VLADIMIR VAPNIK, born 6 December 1936, is a computer scientist, researcher, and academic *(top image)*. While at AT&T (USA), VAPNIK and his colleagues did work on the support vector machine, which he also worked on much earlier before moving to the USA. They demonstrated its performance on a number of problems of interest to the machine learning community, including handwriting recognition. Also he worked on support vector clustering algorithms.



ALEXEY CHERVONENKIS, 7 September 1938 – 22 September 2014, was a Soviet and Russian mathematician *(bottom image)*. He and VAPNIK were the main developers of the VAPNIK–CHERVONENKIS (VC) theory of statistical learning, an important part of computational learning theory.



(Wikipedia)

Recall: Norms in $\mathbb{R}^D$

- **Definition:** Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^D$, $\alpha \in \mathbb{R}$. The mapping $||\cdot|| : \mathbb{R}^D \to \mathbb{R}$ is called a **norm**, if the following properties hold:

$$\textit{Positive definiteness:} \qquad ||\boldsymbol{x}|| > 0 \quad \forall \boldsymbol{x} \in \mathbb{R}^D \backslash \{\boldsymbol{0}\} \tag{1}$$

$$\textit{Absolute homogeneity:} \qquad ||\alpha \boldsymbol{x}|| = |\alpha| ||\boldsymbol{x}|| \tag{2}$$

$$\textit{Triangle inequality:} \qquad ||\boldsymbol{x} + \boldsymbol{y}|| \leqslant ||\boldsymbol{x}|| + ||\boldsymbol{y}|| \tag{3}$$

- From (1) and (2) it follows $||\boldsymbol{x}|| = 0 \iff \boldsymbol{x} = \boldsymbol{0}$, as $||\boldsymbol{0}|| = ||0 \cdot \boldsymbol{0}|| = 0||\boldsymbol{0}|| = 0$
- We use $||\cdot||$ to denote the **EUCLIDean norm**: $||\boldsymbol{x}|| := \sqrt{x_1^2 + \ldots + x_D^2}$

Recall: Scalar Products

- **Definition:** Let $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{R}^D$ and $\alpha, \beta \in \mathbb{R}$. We call the mapping $\langle \cdot, \cdot \rangle : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ a **scalar product**, if the following properties hold:

$$\textit{Positive definiteness:} \quad \langle \boldsymbol{x}, \boldsymbol{x} \rangle > 0 \quad \forall \boldsymbol{x} \in \mathbb{R}^D \backslash \{\boldsymbol{0}\} \tag{4}$$

$$\textit{Symmetry:} \quad \langle \boldsymbol{x}, \boldsymbol{y} \rangle = \langle \boldsymbol{y}, \boldsymbol{x} \rangle \tag{5}$$

$$\textit{Linearity:} \quad \langle \alpha \boldsymbol{x} + \beta \boldsymbol{y}, \boldsymbol{z} \rangle = \alpha \langle \boldsymbol{x}, \boldsymbol{z} \rangle + \beta \langle \boldsymbol{y}, \boldsymbol{z} \rangle \tag{6}$$

- With this definition we can write $||\boldsymbol{x}|| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle} \Longleftrightarrow ||\boldsymbol{x}||^2 = \langle \boldsymbol{x}, \boldsymbol{x} \rangle$
- $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \equiv \boldsymbol{x}^\top \boldsymbol{y}$ is the **canonical scalar product**

Discriminant Functions

Assume we have an **affine-linear function** which defines the **decision boundary**:

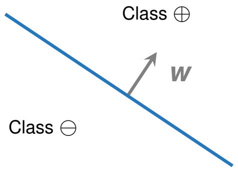
$$f(\boldsymbol{x}) := \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b \tag{7}$$

Remarks:

- $\boldsymbol{w} \in \mathbb{R}^M$ are the **weights**, and $b \in \mathbb{R}$ is called **bias**
- For $\boldsymbol{x} \in \mathbb{R}^M$ on the hyperplane, we have $f(\boldsymbol{x}) = 0$
- An input vector $\boldsymbol{x}$ is assigned...

  ...to the positive class $\oplus$, if $f(\boldsymbol{x}) \geqslant 0$

  ...to the negative class $\ominus$, if $f(\boldsymbol{x}) < 0$



Linear Separability

- Let a dataset $\mathcal{D} := \left\{ (\boldsymbol{x}^n, y_n) \right\}_{n=1}^{N}$ be given, where $\boldsymbol{x}^n \in \mathbb{R}^M$, $y_n \in \{-1, +1\}$
- A new data point $\boldsymbol{x}'$ is classified according to the sign of $f(\boldsymbol{x}')$:

$$\widehat{y} := h_{\boldsymbol{w},b}(\boldsymbol{x}') := \operatorname{sign}\big(f(\boldsymbol{x}')\big) \tag{8}$$

- The sign-function is defined as

$$\operatorname{sign}(z) := \begin{cases} -1 & \text{if } z < 0 \\ +1 & \text{if } z \geqslant 0 \end{cases} \tag{9}$$

Linear Separability (Ctd.)

- A dataset is called **linearly separable** in feature space, if there exist $\boldsymbol{w} \in \mathbb{R}^M$ and $b \in \mathbb{R}$, such that

$$f(\boldsymbol{x}^n) = \langle \boldsymbol{w}, \boldsymbol{x}^n \rangle + b \geqslant 0 \qquad \forall \boldsymbol{x}^n \text{ with } y_n = +1 \tag{10}$$

$$f(\boldsymbol{x}^n) = \langle \boldsymbol{w}, \boldsymbol{x}^n \rangle + b < 0 \qquad \forall \boldsymbol{x}^n \text{ with } y_n = -1 \tag{11}$$

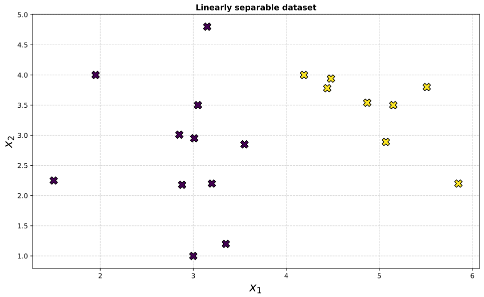
- We can write this in a more compact form:

Linear separability of a dataset:

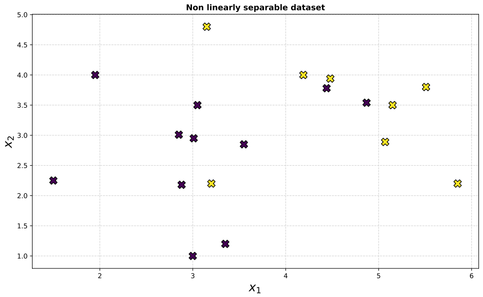
$$y_n f(\boldsymbol{x}^n) \geqslant 0 \qquad \forall n = 1, \dots, N \tag{12}$$

(Not) linearly separable Data

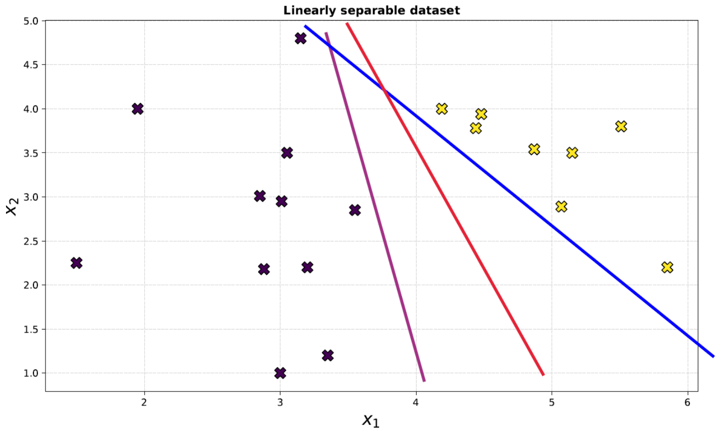
Linearly separable:



Not linearly separable:



Which Decision Boundary is the best?

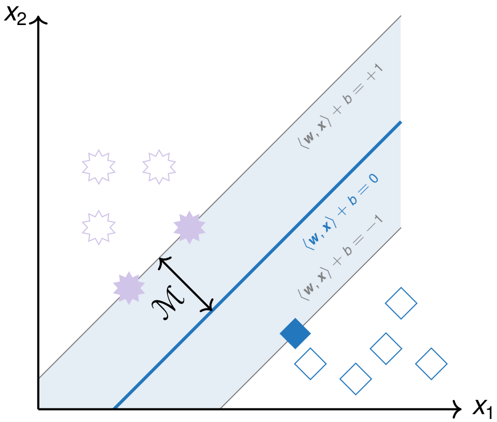


Maximum Margin Classifiers

- An SVM maximizes the **margin** $\mathcal{M}$

$$\mathcal{M}^\star := \max_{\boldsymbol{w}, b} \mathcal{M}$$

- The larger $\mathcal{M}^\star$, the less likely are false predictions $\Rightarrow$ **better generalization**
- The decision boundary is fully determined by the set of **support vectors** $\mathcal{S}$ *(filled data points)*

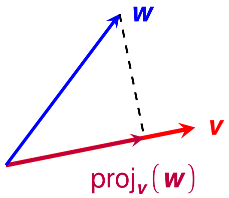

x_2
x_1
$\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = +1$
$\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0$
$\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = -1$
$\mathcal{M}$

Important

(Orthogonal) Projection of Vectors

- The length of the vector $\text{proj}_{\boldsymbol{v}}(\boldsymbol{w})$ is given by *(trigonometry!)*

$$\begin{aligned}\|\text{proj}_{\boldsymbol{v}}(\boldsymbol{w})\| &= \|\boldsymbol{w}\| \cos \measuredangle(\boldsymbol{v}, \boldsymbol{w}) \\ &= \|\boldsymbol{w}\| \frac{\langle \boldsymbol{v}, \boldsymbol{w} \rangle}{\|\boldsymbol{v}\| \cdot \|\boldsymbol{w}\|} = \frac{\langle \boldsymbol{v}, \boldsymbol{w} \rangle}{\|\boldsymbol{v}\|}\end{aligned}$$



- Multiply the length with a unit vector pointing in the direction of $\boldsymbol{v}$:

$$\text{proj}_{\boldsymbol{v}}(\boldsymbol{w}) = \|\text{proj}_{\boldsymbol{v}}(\boldsymbol{w})\| \cdot \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|} = \frac{\langle \boldsymbol{v}, \boldsymbol{w} \rangle}{\|\boldsymbol{v}\|^2} \boldsymbol{v} \tag{13}$$

Weight Vector is perpendicular to the Hyperplane

Lemma: The weight vector $\boldsymbol{w}$ is perpendicular to the hyperplane defined by the discriminant function f.

Proof: Let $\boldsymbol{x} \in \mathbb{R}^M$ and $\boldsymbol{y} \in \mathbb{R}^M$ be two distinct points on the decision surface. Then by definition we have

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0 \qquad \text{and} \qquad f(\boldsymbol{y}) = \langle \boldsymbol{w}, \boldsymbol{y} \rangle + b = 0. \tag{14}$$

This implies $\langle \boldsymbol{w}, \boldsymbol{x} \rangle = \langle \boldsymbol{w}, \boldsymbol{y} \rangle$ which is equivalent to $\langle \boldsymbol{w}, \boldsymbol{x} - \boldsymbol{y} \rangle = 0$ (we have used equation (6) here). The vector $\boldsymbol{x} - \boldsymbol{y}$ is entirely contained in the hyperplane and the dot product with $\boldsymbol{w}$ is equal to zero. Thus, $\boldsymbol{w}$ is orthogonal to the decision boundary defined by f. ■

Distance of a Point to the Hyperplane

Lemma (Distance of a point to the hyperplane): The (perpendicular) distance $d \in \mathbb{R}$ of a point $\boldsymbol{p} \in \mathbb{R}^M$ to the hyperplane is given by

$$d := \frac{|f(\boldsymbol{p})|}{||\boldsymbol{w}||}. \tag{15}$$

Furthermore, we obtain the **signed distance** $d_s \in \mathbb{R}$ of $\boldsymbol{p}$ to the hyperplane by replacing $|f(\boldsymbol{p})|$ with $f(\boldsymbol{p})$ in equation (15). The sign of d_s indicates whether the data point is located on the left side or on right side of the hyperplane.

Distance of a Point to the Hyperplane – Proof

Proof: Let $\boldsymbol{p} \in \mathbb{R}^M$ be an arbitrary point and $\boldsymbol{p}_\perp \in \mathbb{R}^M$ its orthogonal projection onto the decision hyperplane. We notice that $\boldsymbol{p} = \boldsymbol{p}_\perp + d_s \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$, where d_s is the (signed) perpendicular distance of $\boldsymbol{p}$ to the hyperplane whose normal vector is $\boldsymbol{w}$. Hence,

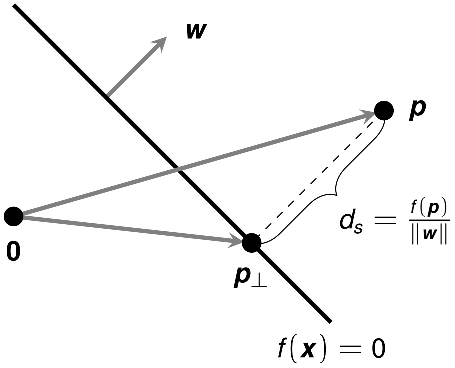
$$f(\boldsymbol{p}) = \langle \boldsymbol{w}, \boldsymbol{p} \rangle + b = \left\langle \boldsymbol{w}, \boldsymbol{p}_\perp + d_s \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \right\rangle + b \overset{(6)}{=} \langle \boldsymbol{w}, \boldsymbol{p}_\perp \rangle + b + d_s \frac{\langle \boldsymbol{w}, \boldsymbol{w} \rangle}{\|\boldsymbol{w}\|}. \tag{16}$$

We rearrange this equation for d_s and exploit $\langle \boldsymbol{w}, \boldsymbol{p}_\perp \rangle + b = f(\boldsymbol{p}_\perp) = 0$ (because $\boldsymbol{p}_\perp$ lies on the decision boundary) as well as $\langle \boldsymbol{w}, \boldsymbol{w} \rangle = \|\boldsymbol{w}\|^2$. We obtain

$$d_s = \frac{f(\boldsymbol{p})}{\|\boldsymbol{w}\|}.$$

Finally we have $d = |d_s| = |f(\boldsymbol{p})| / \|\boldsymbol{w}\|$ and the proof is complete. ■

Distance of a Point to the Hyperplane – Proof (Ctd.)



Distance of a Point to the Hyperplane – Another Proof

Proof: Let $\boldsymbol{p} \in \mathbb{R}^M$ be the point for which we wish to calculate the distance to the hyperplane. Let further $\boldsymbol{x} \in \mathbb{R}^M$, $\boldsymbol{q} \in \mathbb{R}^M$ be two points on the decision boundary. Specifically, we have $f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0$. Furthermore, $\boldsymbol{x} - \boldsymbol{q}$ is entirely contained in the hyperplane. Thus, we have

$$\langle \boldsymbol{w}, \boldsymbol{x} - \boldsymbol{q} \rangle = 0 \quad \overset{(6)}{\iff} \quad \langle \boldsymbol{w}, \boldsymbol{x} \rangle - \langle \boldsymbol{w}, \boldsymbol{q} \rangle = 0 \quad \iff \quad \langle \boldsymbol{w}, \boldsymbol{x} \rangle = \langle \boldsymbol{w}, \boldsymbol{q} \rangle. \tag{17}$$

Equation (17) can be rewritten to $w_1 x_1 + \ldots + w_M x_M = \langle \boldsymbol{w}, \boldsymbol{q} \rangle = -b$. We now define $\boldsymbol{z} := \boldsymbol{p} - \boldsymbol{q}$ and compute the orthogonal projection of $\boldsymbol{z}$ onto the normal vector $\boldsymbol{w}$:

$$\mathrm{proj}_{\boldsymbol{w}}(\boldsymbol{z}) \overset{(13)}{=} \frac{\langle \boldsymbol{w}, \boldsymbol{z} \rangle}{\|\boldsymbol{w}\|^2}\boldsymbol{w} = \frac{\langle \boldsymbol{w}, \boldsymbol{p} - \boldsymbol{q} \rangle}{\|\boldsymbol{w}\|^2}\boldsymbol{w} \overset{(6)}{=} \frac{\langle \boldsymbol{w}, \boldsymbol{p} \rangle - \langle \boldsymbol{w}, \boldsymbol{q} \rangle}{\|\boldsymbol{w}\|^2}\boldsymbol{w} = \frac{\langle \boldsymbol{w}, \boldsymbol{p} \rangle + b}{\|\boldsymbol{w}\|^2}\boldsymbol{w} \tag{18}$$

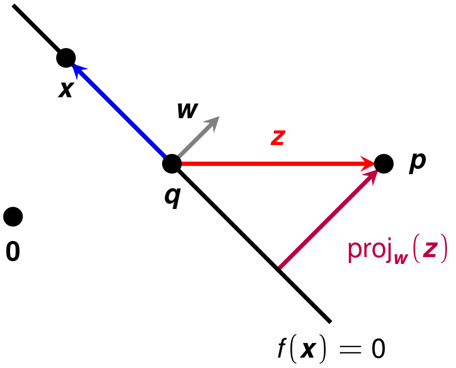
Distance of a Point to the Hyperplane – Another Proof (Ctd.)

The distance d is given by the norm of $\mathrm{proj}_{\boldsymbol{w}}(\boldsymbol{z})$, i. e.

$$\begin{aligned} d = \left\|\mathrm{proj}_{\boldsymbol{w}}(\boldsymbol{z})\right\| = \left\| \frac{\langle \boldsymbol{w}, \boldsymbol{p}\rangle + b}{\|\boldsymbol{w}\|^2} \boldsymbol{w} \right\| &\overset{(2)}{=} \frac{|\langle \boldsymbol{w}, \boldsymbol{p}\rangle + b|}{\|\boldsymbol{w}\|^2} \cdot \|\boldsymbol{w}\| \\ &= \frac{|\langle \boldsymbol{w}, \boldsymbol{p}\rangle + b|}{\|\boldsymbol{w}\|} \\ &= \frac{|f(\boldsymbol{p})|}{\|\boldsymbol{w}\|}, \end{aligned} \tag{19}$$

as claimed. ■

Distance of a Point to the Hyperplane – Another Proof (Ctd.)



Distance of the Hyperplane to the Origin

Lemma (Distance to the origin): The distance of the hyperplane to the origin is given by

$$d_0 := -\frac{b}{\|\boldsymbol{w}\|}. \tag{20}$$

Proof: Let $\boldsymbol{x} \in \mathbb{R}^M$ be a point on the hyperplane. The distance of $\boldsymbol{x}$ to the hyperplane is then

$$\frac{\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b}{\|\boldsymbol{w}\|} = 0 \quad \Longleftrightarrow \quad \frac{\langle \boldsymbol{w}, \boldsymbol{x} \rangle}{\|\boldsymbol{w}\|} = -\frac{b}{\|\boldsymbol{w}\|}.$$

The left-hand side is the scalar projection of $\boldsymbol{x}$ onto $\boldsymbol{w}$, which is the distance to the origin. ■

Derivation of the SVM Primal Optimization Problem

- Let a **linearly separable** dataset $\mathcal{D}$ be given
- From equation (15) we know that the perpendicular distance of a point $\boldsymbol{x}$ to the hyperplane defined by $f(\boldsymbol{x}) = 0$ is given by

$$d = \frac{|f(\boldsymbol{x})|}{\|\boldsymbol{w}\|} = \frac{|\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b|}{\|\boldsymbol{w}\|} \tag{21}$$

- We are only interested in solutions for which all data points are correctly classified, i. e. $y_n f(\boldsymbol{x}^n) \geqslant 0$ for all $n = 1, \dots, N$
- Such a solution exists, otherwise the dataset would **not** be linearly separable

Derivation of the SVM Primal Optimization Problem (Ctd.)

- Thus, the distance d is given by:

$$d = \frac{y_n f(\boldsymbol{x}^n)}{\|\boldsymbol{w}\|} = \frac{y_n\left(\langle \boldsymbol{w}, \boldsymbol{x}^n \rangle + b\right)}{\|\boldsymbol{w}\|} \tag{22}$$

- The **margin** $\mathcal{M}$ is given by the distance of the **closest data point** to the decision surface *(we call this data point* $\widetilde{\boldsymbol{x}}$*)*

Goal:

We wish to optimize the SVM parameters w and b so as to maximize this distance (maximum margin)

Derivation of the SVM Primal Optimization Problem (Ctd.)

Therefore we have to solve:

SVM optimization problem:

$$\left(\boldsymbol{w}^\star, b^\star\right) := \arg\max_{\boldsymbol{w},b} \left\{ \frac{1}{\|\boldsymbol{w}\|} \min_i \left\{ y_i\left(\langle \boldsymbol{w}, \boldsymbol{x}^i \rangle + b\right) \right\} \right\} \tag{23}$$

Problem: A direct solution to the optimization problem (23) **would be very hard to obtain!** We shall therefore rewrite this optimization problem to obtain a solution more easily

Derivation of the SVM Primal Optimization Problem (Ctd.)

- We notice that rescaling $\boldsymbol{w}$ and b by a factor κ **does not change the distance** to the decision boundary *(we can cancel* κ*)*
- Therefore, we can choose κ so that

$$\widetilde{y}\big(\langle \boldsymbol{w}, \widetilde{\boldsymbol{x}}\rangle + b\big) = 1 \tag{24}$$

for the data point $\widetilde{\boldsymbol{x}}$ (and corresponding label $\widetilde{y}$) closest to the decision surface

- All data points $\boldsymbol{x}^n$ $(n = 1, \dots, N)$ then satisfy the constraint:

$$y_n\big(\langle \boldsymbol{w}, \boldsymbol{x}^n\rangle + b\big) \geqslant 1 \tag{25}$$

Important

Calculation of the Margin

- Let $\widetilde{\boldsymbol{x}}$ be the data point closest to the decision boundary
- The margin is defined as the perpendicular distance of $\widetilde{\boldsymbol{x}}$ to the boundary
- We plug equation (24) into equation (22) to obtain the margin

Lemma (Margin): The margin $\mathcal{M}$ is the distance of the closest data point to the decision boundary. It is given by

$$\mathcal{M} := \frac{1}{\|\boldsymbol{w}\|}. \tag{26}$$

Primal Optimization Problem for SVMs

We substitute equation (25) into equation (23) to obtain the

Primal optimization problem for SVMs:

$$\textit{minimize} \qquad \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{27}$$

$$\textit{subject to} \qquad y_n\big(\langle \boldsymbol{w}, \boldsymbol{x}^n\rangle + b\big) - 1 \geqslant 0 \qquad (n = 1, \dots, N) \tag{28}$$

- Instead of maximizing $\frac{1}{\|\boldsymbol{w}\|}$, we choose to minimize $\|\boldsymbol{w}\|^2$
- The factor $^1/_2$ was added for later mathematical convenience

Quadratic Programming (QP)

- This is a **quadratic programming (QP)** problem
- The objective function is a quadratic function of the parameters $\boldsymbol{w}$:

$$q(\boldsymbol{w}) := \frac{1}{2}\|\boldsymbol{w}\|^2$$

- The constraints **(one for each data point!)** are linear:

$$g_n(\boldsymbol{w}, b) := -y_n\big(\langle \boldsymbol{w}, \boldsymbol{x}^n \rangle + b\big) - 1 \leqslant 0$$

- A global optimum is guaranteed to exist due to the **convexity** of q

Primal and dual Optimization Problems

- We will derive the dual optimization problem for SVMs which has some benefits
- Consider the **primal optimization problem**

$$\begin{aligned} &\textit{minimize} && q(\boldsymbol{w}) \\ &\textit{subject to} && g_n(\boldsymbol{w}, b) \leqslant 0 \qquad (n = 1, \dots, N). \end{aligned}$$

- The **dual optimization problem** is then given by

$$\begin{aligned} &\underline{\textit{maximize}}\ (!) && \mathfrak{D}(\boldsymbol{\alpha}) := \inf_{\boldsymbol{w} \in \mathbb{R}^M, b \in \mathbb{R}} \mathfrak{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) \\ &\textit{subject to} && \boldsymbol{\alpha} \geqslant \boldsymbol{0}. \end{aligned}$$

LAGRANGE Function

- The **LAGRANGE function** *(also called Lagrangian)* is defined as:

$$\mathfrak{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) := q(\boldsymbol{w}) + \sum_{n=1}^{N} \alpha_n g_n(\boldsymbol{w}, b)$$

- Since $\mathfrak{L}$ is convex, we can compute $\inf_{\boldsymbol{w}\in\mathbb{R}^M, b\in\mathbb{R}} \mathfrak{L}(\boldsymbol{w}, b, \boldsymbol{\alpha})$ by considering

$$\nabla_{\boldsymbol{w}}\mathfrak{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) \overset{!}{=} \mathbf{0} \qquad \text{and} \qquad \nabla_{b}\mathfrak{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) \overset{!}{=} 0$$

- We then solve for the minimizing primal variables $\boldsymbol{w}^\star$ and $b^\star$ and plug them into $\mathfrak{L}$ *(or add a constraint to the dual problem if this is not possible)*
- The result is the dual function $\mathfrak{D}$ which only depends on the multipliers $\boldsymbol{\alpha}$

SVM LAGRANGE Function

We plug in the function definitions into $\mathfrak{L}$ and obtain the

LAGRANGE function for SVMs:

$$\mathfrak{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{n=1}^{N} \alpha_n \left[y_n \left(\langle \boldsymbol{w}, \boldsymbol{x}^n \rangle + b \right) - 1 \right] \tag{29}$$

Later we shall see that the LAGRANGE multipliers α_n will be non-zero for all support vectors, all other multipliers will turn out to be zero

Gradients of the LAGRANGE Function

We compute the gradients of $\mathfrak{L}$ w. r. t. $\boldsymbol{w}$ and b and set them to zero:

$$\nabla_{\boldsymbol{w}}\mathfrak{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \boldsymbol{w} - \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}^n \overset{!}{=} \boldsymbol{0} \quad \Longrightarrow \quad \boxed{\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}^n} \tag{30}$$

$$\nabla_{b}\mathfrak{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = - \sum_{n=1}^{N} \alpha_n y_n \overset{!}{=} 0 \quad \Longrightarrow \quad \boxed{\sum_{n=1}^{N} \alpha_n y_n = 0} \tag{31}$$

From equation (30) we see that $\boldsymbol{w}$ is a linear combination of the input!

Dual Optimization Problem for SVMs

Now we plug equations (30) and (31) into the LAGRANGE function (29) to obtain:

$$\begin{aligned}\mathfrak{D}(\boldsymbol{\alpha}) &= \frac{1}{2}\left\langle \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}^i, \sum_{j=1}^{N} \alpha_j y_j \boldsymbol{x}^j \right\rangle - \sum_{i=1}^{N} \alpha_i \left[y_i \left(\left\langle \sum_{j=1}^{N} \alpha_j y_j \boldsymbol{x}^j, \boldsymbol{x}^i \right\rangle + b \right) - 1 \right] \\ &\overset{(6)}{=} \frac{1}{2}\left\langle \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}^i, \sum_{j=1}^{N} \alpha_j y_j \boldsymbol{x}^j \right\rangle - \left\langle \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}^i, \sum_{j=1}^{N} \alpha_j y_j \boldsymbol{x}^j \right\rangle \\ &\quad - \underbrace{\sum_{i=1}^{N} \alpha_i y_i b}_{= 0 \Rightarrow (31)} + \sum_{i=1}^{N} \alpha_i \end{aligned} \tag{32}$$

Dual Optimization Problem for SVMs (Ctd.)

$$= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \left\langle \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}^i, \sum_{j=1}^{N} \alpha_j y_j \boldsymbol{x}^j \right\rangle$$

$$\overset{(6)}{=} \boxed{\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}^i, \boldsymbol{x}^j \rangle} \tag{33}$$

- We have found the dual objective function $\mathfrak{D}$
- The dual optimization problem has the constraints $\boldsymbol{\alpha} \geqslant \mathbf{0}$ as well as the constraint given by equation (31): $\sum_{i=1}^{N} \alpha_i y_i = 0$

Dual Optimization Problem for SVMs (Ctd.)

Dual optimization problem for SVMs (WOLFE dual):

$$\textit{maximize} \quad \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}^i, \boldsymbol{x}^j \rangle \tag{34}$$

$$\textit{subject to} \quad \alpha_i \geqslant 0 \text{ for all } i = 1, \ldots, N \tag{35}$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{36}$$

Some Remarks

- The solution to a quadratic programming problem in D variables has **computational complexity** of $\mathcal{O}(D)$ in general
- The primal SVM optimization problem had M variables (# features), whereas the dual representation has N variables (# data points)

Considering $M \ll N$, the dual representation seems disadvantageous

However, the dual representation of the optimization problem unlocks the concept of kernels *(see next section)*

Computation of the Offset b

- Once we know $\boldsymbol{\alpha}$, we can determine b by noting that any support vector satisfies *(where $\mathcal{S}$ is the set of indices of support vectors)*:

$$y_i f(\boldsymbol{x}^i) = y_i \left(\sum_{j \in \mathcal{S}} \alpha_j y_j \langle \boldsymbol{x}^i, \boldsymbol{x}^j \rangle + b \right) = 1 = y_i^2 \tag{37}$$

- Average over all support vectors to compute b:

$$b := \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \left(y_i - \sum_{j \in \mathcal{S}} \alpha_j y_j \langle \boldsymbol{x}^i, \boldsymbol{x}^j \rangle \right) \tag{38}$$

SVM Decision Rule

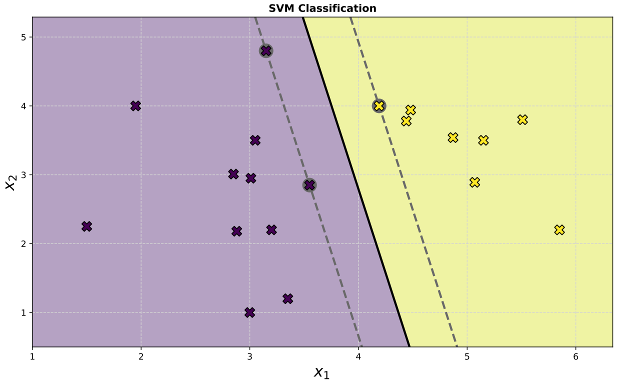
- Given our derivations, we can rewrite the SVM decision rule as follows:

$$h(\boldsymbol{x}') := \operatorname{sign}\big(\langle \boldsymbol{w}, \boldsymbol{x}' \rangle + b\big) \overset{(30)}{=} \operatorname{sign}\left(\sum_{i \in \mathcal{S}} \alpha_i y_i \langle \boldsymbol{x}^i, \boldsymbol{x}' \rangle + b\right) \tag{39}$$

- $\boldsymbol{x}'$ is an unknown instance for which the class label is not known

Since all α_i will be zero for non-support vectors, the decision for a class depends on the support vectors only! This makes predictions fast, even for large datasets. The number of support vectors can also be used as an evaluation criterion.

Example: Linear SVM

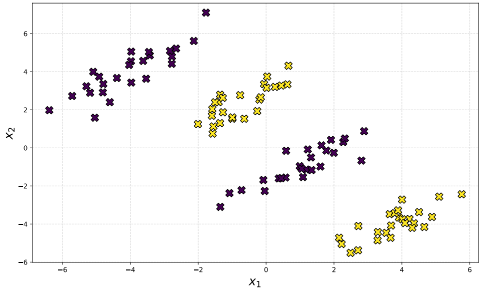


Section:

Sparse Kernel Machines

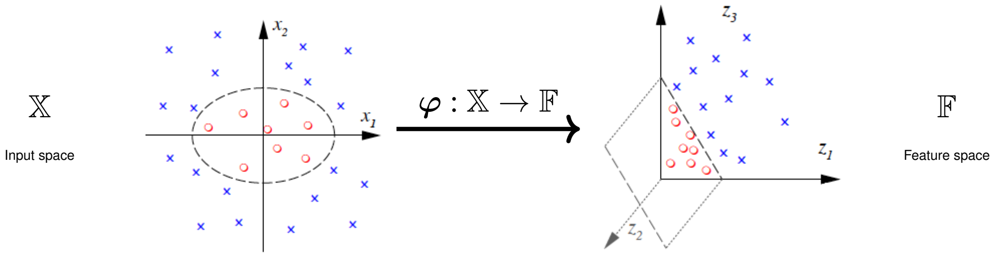
Non-Linear SVMs / Non-Linear Separability

- So far we have assumed **linear separability** of the data
- What if the data is **not linearly separable?**
- We cannot find a straight line to separate the data
- We have already learned about the **feature mapping** technique



Feature Mapping

The mapping function φ maps from input space $\mathbb{X}$ to feature space $\mathbb{F}$:



$$\varphi(x_1, x_2) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right) =: (z_1, z_2, z_3)$$

Feature Mapping (Ctd.)

Disadvantages of feature mapping:

- When using feature maps, we **EXPLICITLY** transform the data to a higher dimension where we hope classification becomes easier
- Computing the feature map can become very expensive
- And how do we know how many and which dimensions to add?

Alternative: In this section we will introduce the **kernel concept** which can be used as an alternative to feature mapping *(if certain prerequisites are met)*

Kernel Methods and the Kernel Trick

Kernel methods owe their name to the use of **kernel functions**, which enable them to operate in a high-dimensional *(potentially even infinite-dimensional)*, **IMPLICIT** feature space **without ever computing the coordinates of the data in that space**, but rather by simply **computing the inner products between the images of all pairs of data in the feature space.**

This operation is often computationally cheaper than the explicit computation of the coordinates *(also known as feature mapping)*. This approach is referred to as the **kernel trick**.

(Wikipedia)

Important

What is a Kernel Function?

- Let $\mathbb{X}$ be the input space and $\mathbb{F}$ the higher-dimensional feature space
- A kernel function k can be considered a **similarity function**, i. e. $k(\boldsymbol{x}, \widetilde{\boldsymbol{x}})$ is a measure of how similar $\boldsymbol{x} \in \mathbb{X}$ and $\widetilde{\boldsymbol{x}} \in \mathbb{X}$ are in feature space $\mathbb{F}$

A **kernel function** $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ maps a pair of input features to a real number. We can express k in terms of a feature map:

$$k(\boldsymbol{x}, \widetilde{\boldsymbol{x}}) := \left\langle \varphi(\boldsymbol{x}), \varphi(\widetilde{\boldsymbol{x}}) \right\rangle_{\mathbb{F}} \tag{40}$$

with $\varphi : \mathbb{X} \to \mathbb{F}$, i. e. k must be **symmetric** and **positive-semidefinite**.

Inner Products measure Distances!

Consider distances in the transformed feature space $\mathbb{F}$:

$$\begin{aligned}\|\varphi(\boldsymbol{x}) - \varphi(\widetilde{\boldsymbol{x}})\|^2 &= \langle \varphi(\boldsymbol{x}) - \varphi(\widetilde{\boldsymbol{x}}), \varphi(\boldsymbol{x}) - \varphi(\widetilde{\boldsymbol{x}}) \rangle \\ &= \langle \varphi(\boldsymbol{x}), \varphi(\boldsymbol{x}) \rangle - 2\langle \varphi(\boldsymbol{x}), \varphi(\widetilde{\boldsymbol{x}}) \rangle + \langle \varphi(\widetilde{\boldsymbol{x}}), \varphi(\widetilde{\boldsymbol{x}}) \rangle\end{aligned}$$

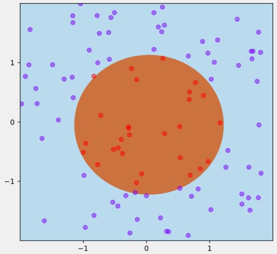
Conclusion: Distances can be computed by evaluating inner products!

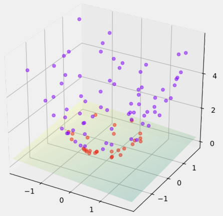
Example: Kernel Function

Let the following feature map be given:

$$\varphi : \mathbb{R}^2 \to \mathbb{R}^3, \left(x_1, x_2\right)^\top \mapsto \left(x_1, x_2, x_1^2 + x_2^2\right)^\top \tag{41}$$

Visualization:





Example: Kernel Function (Ctd.)

The kernel function k corresponding to the feature map given in (41) is

$$k(\boldsymbol{x}, \widetilde{\boldsymbol{x}}) = \langle \boldsymbol{x}, \widetilde{\boldsymbol{x}} \rangle + \|\boldsymbol{x}\|^2 \cdot \|\widetilde{\boldsymbol{x}}\|^2. \tag{42}$$

Proof: Let $\boldsymbol{x} = \left(x_1, x_2\right)^\top \in \mathbb{R}^2$ and $\widetilde{\boldsymbol{x}} = \left(\widetilde{x}_1, \widetilde{x}_2\right)^\top \in \mathbb{R}^2$. Then:

$$\begin{aligned} \left\langle \left(x_1, x_2, x_1^2 + x_2^2\right), \left(\widetilde{x}_1, \widetilde{x}_2, \widetilde{x}_1^2 + \widetilde{x}_2^2\right) \right\rangle &= x_1\widetilde{x}_1 + x_2\widetilde{x}_2 + \left(x_1^2 + x_2^2\right)\left(\widetilde{x}_1^2 + \widetilde{x}_2^2\right) \\ &= \langle \boldsymbol{x}, \widetilde{\boldsymbol{x}} \rangle + \|\boldsymbol{x}\|^2 \cdot \|\widetilde{\boldsymbol{x}}\|^2 \end{aligned}$$

■

Kernel Matrix / GRAM Matrix

We define:

Kernel matrix:

For a given dataset $\mathcal{D}$ comprising the N feature vectors $\boldsymbol{x}^1, \boldsymbol{x}^2, \dots, \boldsymbol{x}^N$, we define the **kernel matrix** (also known as **GRAM / GRAMian matrix**) as:

$$\boldsymbol{K} := \left[k_{ij}\right]_{i,j=1,2,\dots,N} \in \mathbb{R}^{N \times N} \tag{43}$$

where

$$k_{ij} := \left\langle \varphi\left(\boldsymbol{x}^i\right), \varphi\left(\boldsymbol{x}^j\right) \right\rangle_{\mathbb{F}} = k\left(\boldsymbol{x}^i, \boldsymbol{x}^j\right) \tag{44}$$

Kernel Matrix / GRAM Matrix (Ctd.)

Lemma: The kernel matrix $\boldsymbol{K}$ is positive semi-definite.

Proof: Let $\boldsymbol{z} \in \mathbb{R}^N$ be an arbitrary vector. Then we have

$$\boldsymbol{z}^\top \boldsymbol{K} \boldsymbol{z} = \sum_{i=1}^{N} \sum_{j=1}^{N} z_i z_j k_{ij} \overset{(44)}{=} \sum_{i=1}^{N} \sum_{j=1}^{N} z_i z_j \left\langle \varphi(\boldsymbol{x}^i), \varphi(\boldsymbol{x}^j) \right\rangle_{\mathbb{F}}$$

$$\overset{(6)}{=} \left\langle \sum_{i=1}^{N} z_i \varphi(\boldsymbol{x}^i), \sum_{j=1}^{N} z_j \varphi(\boldsymbol{x}^j) \right\rangle_{\mathbb{F}} = \left\| \sum_{i=1}^{N} z_i \varphi(\boldsymbol{x}^i) \right\|_{\mathbb{F}}^2 \geqslant 0.$$

■

MERCER's Condition

- In practice, we are usually not interested in the mapping function φ itself
- We only need to know **that it exists**
- The mapping function φ is guaranteed to exist, if the kernel function k fulfills MERCER's condition *(see next slide)*

If the kernel function k satisfies MERCER's condition, we call k a **MERCER kernel**

MERCER's Condition (Ctd.)

MERCER's theorem:

For any **symmetric** function $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ that is **square integrable** on its domain and which satisfies the condition

$$\int_{\mathbb{X}} \int_{\mathbb{X}} g(\boldsymbol{x}) g(\widetilde{\boldsymbol{x}}) k(\boldsymbol{x}, \widetilde{\boldsymbol{x}}) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\widetilde{\boldsymbol{x}} \geqslant 0 \tag{45}$$

for all square integrable functions g, there exist transforms $\varphi_j : \mathbb{X} \to \mathbb{R}$ and $\lambda_j \geqslant 0$ so that for all $\boldsymbol{x}, \widetilde{\boldsymbol{x}} \in \mathbb{X}$

$$k(\boldsymbol{x}, \widetilde{\boldsymbol{x}}) = \sum_j \lambda_j \varphi_j(\boldsymbol{x}) \varphi_j(\widetilde{\boldsymbol{x}}). \tag{46}$$

When can Kernel Methods be used?

- Many algorithms can be *'kernelized'*
- The most prominent example are
  - Support Vector Machines,
  - Kernel regression,
  - Kernel Principal Component Analysis (PCA), and the
  - Kernel Perceptron

IMPORTANT: The prerequisite for using kernels is that the original input vectors appear exclusively in inner products. These inner products can then be replaced by a kernel function.

Important

Well-known Kernels

- **Linear kernel**

$$k(\boldsymbol{x}, \widetilde{\boldsymbol{x}}) = \langle \boldsymbol{x}, \widetilde{\boldsymbol{x}} \rangle \tag{47}$$

- **Polynomial kernel** (with hyperparameter p)

$$k(\boldsymbol{x}, \widetilde{\boldsymbol{x}}) = \left(\langle \boldsymbol{x}, \widetilde{\boldsymbol{x}} \rangle + c \right)^p \tag{48}$$

- **GAUSSIAN (RBF) kernel** (with hyperparameter s, bandwidth parameter)

$$k(\boldsymbol{x}, \widetilde{\boldsymbol{x}}) = \exp\left\{ -\frac{\|\boldsymbol{x} - \widetilde{\boldsymbol{x}}\|^2}{2s^2} \right\} \tag{49}$$

Kernels in other Domains

- The concept of kernels can be used in other domains as well:
- **String kernels**
  - Operate on strings
  - A string kernel measures the similarity of strings
  - String kernels allow kernel algorithms to work with strings **without having to translate these to fixed-length, real-valued feature vectors**
  - **p-spectrum kernels:** Count the number of substrings in common between the two strings
- **Graph kernels**

(This list is not exhaustive!)

Construction of new Kernels

In general it is not straightforward to check if MERCER's condition is satisfied, but it is possible to construct new kernels out of known ones:

Lemma: If $k_1(\boldsymbol{x}, \widetilde{\boldsymbol{x}})$ and $k_2(\boldsymbol{x}, \widetilde{\boldsymbol{x}})$ are valid kernels, then so are:

- $c \cdot k_1(\boldsymbol{x}, \widetilde{\boldsymbol{x}})$ for any constant $c \in \mathbb{R}$
- $k_1(\boldsymbol{x}, \widetilde{\boldsymbol{x}}) + k_2(\boldsymbol{x}, \widetilde{\boldsymbol{x}})$
- $k_1(\boldsymbol{x}, \widetilde{\boldsymbol{x}}) \cdot k_2(\boldsymbol{x}, \widetilde{\boldsymbol{x}})$
- $f(\boldsymbol{x}) \cdot k_1(\boldsymbol{x}, \widetilde{\boldsymbol{x}}) \cdot f(\widetilde{\boldsymbol{x}})$ for any function f

This list is not exhaustive! See [Bishop.2006], chapter 6.2 for more details

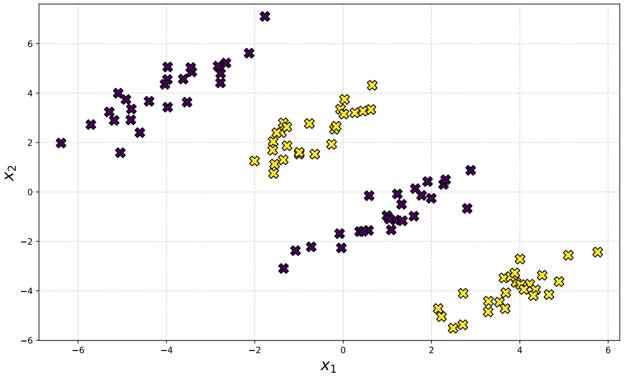
Incorporation of Kernel Functions for SVMs

- The kernel function k replaces any occurrence of a scalar product $\langle \boldsymbol{x}, \widetilde{\boldsymbol{x}} \rangle$ between feature vectors
- **Example:**

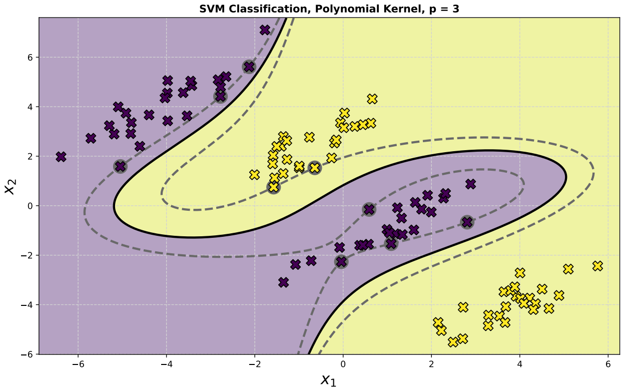
$$\mathfrak{D}(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k\left(\boldsymbol{x}^i, \boldsymbol{x}^j\right) \tag{50}$$

$$h(\boldsymbol{x}') = \operatorname{sign}\left(\sum_{i \in \mathcal{S}} \alpha_i y_i k\left(\boldsymbol{x}^i, \boldsymbol{x}'\right) + b \right) \tag{51}$$

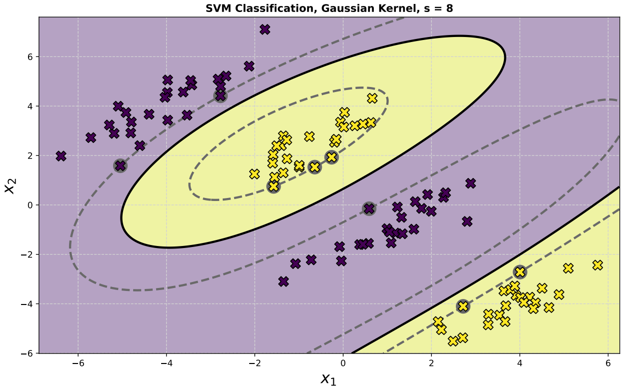
Example: Non-linear Data Set



Example: Polynomial Kernel ($p = 3$)



Example: GAUSSian (RBF) Kernel ($s = 8$)



Advantages and Disadvantages of the Kernel Method

Advantage: High-dimensional feature representations don't have to be computed explicitly! In theory we can work in **infinite-dimensional feature spaces!**

Disadvantages:

- The price is that we now have a **non-parametric method**, i. e. we need the training dataset to make predictions *(only the support vectors are needed!)*
- We no longer have explicit representations of the parameters $\boldsymbol{w}$ and b
- The method **becomes slow** for large amounts of data (> 50 k records)

Section:

Soft-Margin Support Vector Machines

Overlapping Data
Slack Variables
Incorporating Slack for SVMs

Overlapping Distributions

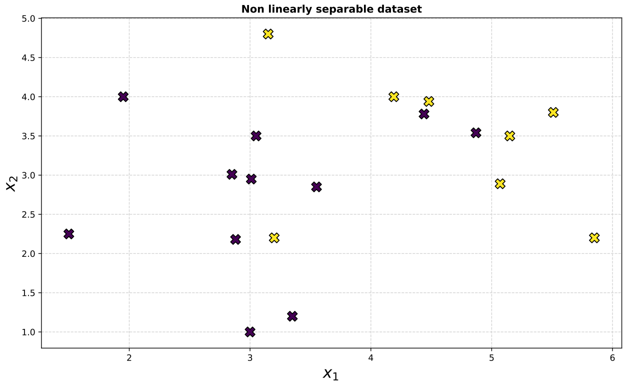
- Until now we have assumed the data to be linearly separable

  $\Longrightarrow$ ***Hard-margin SVM finds the best separating hyperplane***
- **But:** In general, the classes may have overlapping distributions

  $\Longrightarrow$ ***Hard margin leads to overfitting and poor generalization***
- We will modify the algorithm to deal with overlapping distributions

Soft-margin SVMs allow for misclassifications of some data points while introducing a penalty. The penalty increases linearly with the distance from the decision boundary. This is done using **slack variables** ξ_n (German: *Schlupfvariable*).

Example Overlapping Class Distributions



Slack Variables

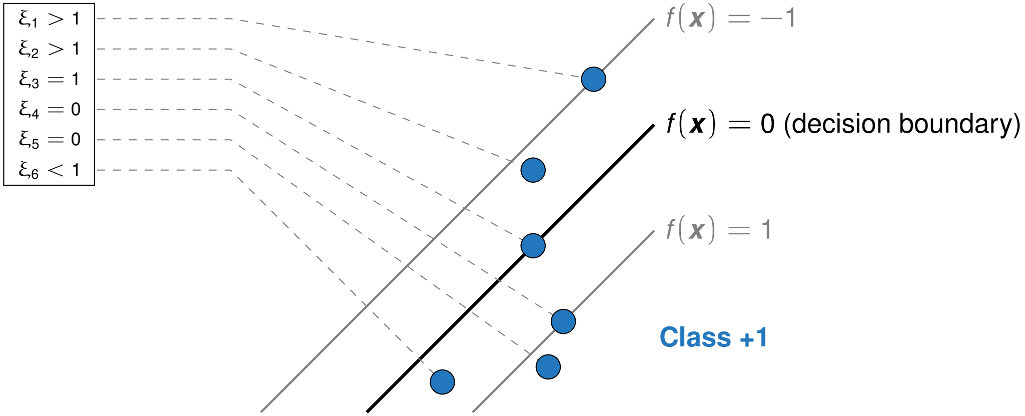
- We introduce one slack variable $\xi_n \geqslant 0 \; (n = 1, \dots, N)$ per data point
- **Different cases**:

$\xi_n = 0$	if $\boldsymbol{x}^n$ is on the correct side of the boundary *(outside or on margin boundary)*
$0 < \xi_n < 1$	if $\boldsymbol{x}^n$ is inside the margin, but on the correct side of the boundary
$\xi_n = 1$	if $\boldsymbol{x}^n$ is on the decision boundary
$\xi_n > 1$	if $\boldsymbol{x}^n$ lies on the wrong side of the decision boundary *(misclassification)*

- The optimization constraints (28) are replaced with:

$$y_n f(\boldsymbol{x}^n) \geqslant 1 - \xi_n \qquad (n = 1, \dots, N) \tag{52}$$

Slack Variables (Ctd.)



Primal Optimization Problem for Soft-Margin SVMs

Maximize the margin, but penalize points on the wrong side of the margin boundary

Primal optimization problem for soft-margin SVMs:

$$\text{minimize} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N}\xi_n \qquad (C > 0) \tag{53}$$

$$\text{subject to} \quad y_n\big(\langle\boldsymbol{w}, \boldsymbol{x}^n\rangle + b\big) \geqslant 1 - \xi_n \tag{54}$$

$$\xi_n \geqslant 0 \qquad (n = 1, \ldots, N) \tag{55}$$

Corresponding LAGRANGE Function

- $C > 0$ is a hyperparameter of the model controlling the **degree of softness**
- The larger C the more we penalize
- $C \to \infty$ recovers the hard-margin SVM introduced in the first section
- The LAGRANGE function is: ($\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^N$ are vectors of LAGRANGE multipliers)

$$\begin{aligned} \mathfrak{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{n=1}^{N} \xi_n \\ - \sum_{n=1}^{N} \alpha_n \left[y_n f(\boldsymbol{x}^n) - 1 + \xi_n\right] - \sum_{n=1}^{N} \beta_n \xi_n \end{aligned} \tag{56}$$

Dual Optimization Problem for Soft-Margin SVMs

Dual optimization problem for soft-margin SVMs:

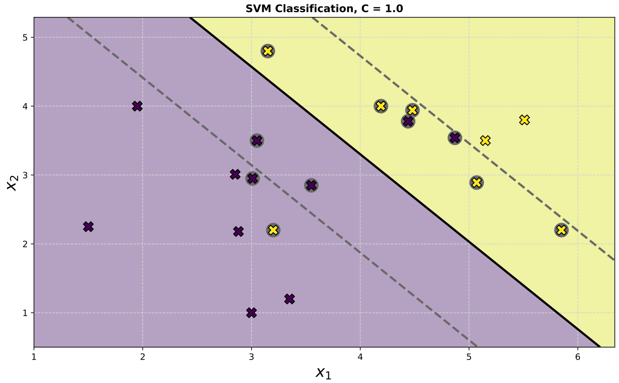
$$\textit{maximize} \qquad \mathfrak{D}(\boldsymbol{\alpha}) := \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}^i, \boldsymbol{x}^j \rangle \tag{57}$$

$$\textit{subject to} \qquad 0 \leqslant \alpha_i \leqslant C \qquad (i = 1, \ldots, N) \tag{58}$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{59}$$

The constraints in (58) are called **box constraints** *(for obvious reasons)*

Example: Soft Margin SVM



Section:

Sequential Minimal Optimization (SMO)

What is SMO?

- **Sequential Minimal Optimization (SMO)** is an optimization algorithm for solving the quadratic programming problem that arises during the training of SVMs
- It uses a modified version of **coordinate ascent**
- SMO was introduced 1998 by John Platt at Microsoft Research
- Please find the original paper ⇒ here

Why the name? Rather than solving the problem numerically as a whole, SMO breaks it into a series of smallest possible sub-problems which are solved analytically in a sequential manner

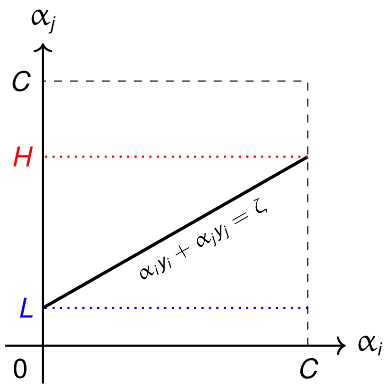
Main Idea of the Algorithm

- At each iteration, SMO ensures that the constraints (58) and (59) are satisfied
- SMO updates **two multipliers at a time** *(the others are fixed)*
- Rearranging (59) for α_i reveals a **functional dependence** of α_i on the other α's

$$\alpha_i = -y_i \sum_{\substack{k=1 \\ k \neq i}}^{N} \alpha_k y_k \tag{60}$$

- This means we cannot change α_i without changing at least one other multiplier α_j, otherwise constraint (59) is violated

Main Idea of the Algorithm (Ctd.)



- We choose to update α_i and α_j for $i \neq j$
- From (59) we obtain:

$$\alpha_i y_i + \alpha_j y_j = -\sum_{\substack{k=1 \\ k \neq i,j}}^{N} \alpha_k y_k =: \zeta \qquad (61)$$

- The solution has to lie on the line so as to fulfill the constraint
- We have to choose $\alpha_j \in [L, H]$

Computation of L and H

- We begin by expressing α_i in terms of α_j: ($\Rightarrow 1/y_i = y_i$, as $y_i \in \{-1, 1\}$)

$$\alpha_i y_i + \alpha_j y_j = \zeta \quad \Longleftrightarrow \quad \alpha_i = y_i\big(\zeta - \alpha_j y_j\big) \tag{62}$$

- We know $0 \leqslant \alpha_i \leqslant C$, therefore $0 \leqslant y_i(\zeta - \alpha_j y_j) \leqslant C$
- Rearranging for α_j we obtain:

$$y_i\zeta - C \leqslant \alpha_j \leqslant y_i\zeta \qquad \text{if} \quad y_i = y_j \tag{63}$$

$$-y_i\zeta \leqslant \alpha_j \leqslant C - y_i\zeta \qquad \text{if} \quad y_i \neq y_j \tag{64}$$

Computation of L and H (Ctd.)

- We plug in the definition of ζ (61) to simplify these bounds:

$$\alpha_i^{\text{old}} + \alpha_j^{\text{old}} - C \leqslant \alpha_j \leqslant \alpha_i^{\text{old}} + \alpha_j^{\text{old}} \qquad \text{if} \quad y_i = y_j \tag{65}$$

$$\alpha_j^{\text{old}} - \alpha_i^{\text{old}} \leqslant \alpha_j \leqslant C + \alpha_j^{\text{old}} - \alpha_i^{\text{old}} \qquad \text{if} \quad y_i \neq y_j \tag{66}$$

- Also we know that $0 \leqslant \alpha_j \leqslant C$ has to be fulfilled
- We therefore have two lower bounds and two upper bounds for α_j which have to be satisfied **simultaneously**

Computation of *L* and *H* (Ctd.)

We combine the conditions for α_j to obtain:

Case 1: $y_i = y_j$

$$L := \max\left(0, \alpha_i^{\text{old}} + \alpha_j^{\text{old}} - C\right)$$
$$H := \min\left(C, \alpha_i^{\text{old}} + \alpha_j^{\text{old}}\right)$$

Case 2: $y_i \neq y_j$

$$L := \max\left(0, \alpha_j^{\text{old}} - \alpha_i^{\text{old}}\right)$$
$$H := \min\left(C, C + \alpha_j^{\text{old}} - \alpha_i^{\text{old}}\right)$$

We have: We know the range α_j has to be chosen from.

We need: As a next step we have to find the maximum on the line!

Intuition: Maximization of the Objective

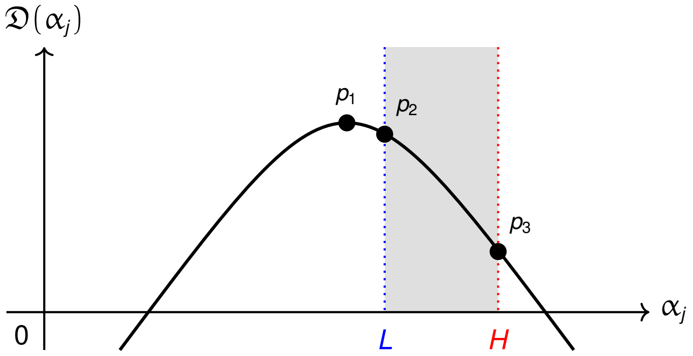
- We only update the parameters α_i and α_j *(all other α's are fixed)* and use equation (62) to express α_i in terms of α_j
- The objective function (57) becomes a **1-dimensional quadratic function** in α_j:

$$\begin{aligned}\mathfrak{D}(\alpha_i, \alpha_j) &= \mathfrak{D}\big(y_i(\zeta - \alpha_j y_j), \alpha_j\big) \\ &= a\alpha_j^2 + b\alpha_j + c \qquad (67)\end{aligned}$$

for some constants $a, b, c \in \mathbb{R}$ *(where the constant c can be ignored)*

This problem is easy to solve with standard optimization techniques. But consider that the optimal value for α_j has to lie within the bounds L and H!

Intuition: Maximization of the Objective (Ctd.)



- **We only have to check the three points** p_1, p_2, p_3
- Left: The global optimum is not in $[L, H]$
- We have to **clip** the new value for α_j
- We set $\alpha_j^{\text{new}} := L$

Maximization of the Objective: Formulas

- The value for α_j at the global optimum p_1 is given by *(check the paper for details!)*:

$$\alpha_j^\star := \alpha_j^{\text{old}} - \frac{y_j(E_i - E_j)}{\eta} \tag{68}$$

$$E_k := \sum_{n=1}^{N} \alpha_n y^n \langle \boldsymbol{x}^n, \boldsymbol{x}^k \rangle + b - y_k \tag{69}$$

$$\eta := 2\langle \boldsymbol{x}^i, \boldsymbol{x}^j \rangle - \langle \boldsymbol{x}^i, \boldsymbol{x}^i \rangle - \langle \boldsymbol{x}^j, \boldsymbol{x}^j \rangle \tag{70}$$

- E_k is the error between the SVM output on the k-th example and the true label

Maximization of the Objective: Formulas (Ctd.)

- η is the second-order derivative of $\mathfrak{D}$ along the line
- When calculating η you may want to **replace the inner products with kernel functions**
- Next, we clip α_j to lie within the range $[L, H]$:

$$\alpha_j^{\text{new}} := \begin{cases} H & \text{if } \alpha_j^\star > H \\ \alpha_j^\star & \text{if } L \leqslant \alpha_j^\star \leqslant H \\ L & \text{if } \alpha_j^\star < L \end{cases} \tag{71}$$

Computation of α_i^{new}

- We have to compute α_i^{new} based on the value α_j^{new} to satisfy the constraint (59)
- We had: $\alpha_i = y_i\big(\zeta - \alpha_j y_j\big)$, therefore we get the update rule:

$$\begin{aligned}\alpha_i^{\text{new}} &= y_i(\zeta - \alpha_j^{\text{new}} y_j) \\ &= y_i\big(\alpha_i^{\text{old}} y_i + \alpha_2^{\text{old}} y_i - \alpha_j^{\text{new}} y_j\big) \\ &= \alpha_i^{\text{old}} + y_i y_j\big(\alpha_j^{\text{old}} - \alpha_j^{\text{new}}\big) \qquad (72)\end{aligned}$$

KARUSH-KUHN-TUCKER Conditions for the SVM

- The **KARUSH-KUHN-TUCKER (KKT) conditions** provide a necessary condition for the optimal solution
- For SVMs, these conditions are also sufficient:

$$\alpha_n = 0 \quad \Longrightarrow \quad y_n\big(\langle \boldsymbol{w}, \boldsymbol{x}^n\rangle + b\big) \geqslant 1 \tag{73}$$

$$\alpha_n = C \quad \Longrightarrow \quad y_n\big(\langle \boldsymbol{w}, \boldsymbol{x}^n\rangle + b\big) \leqslant 1 \tag{74}$$

$$0 < \alpha_i < C \quad \Longrightarrow \quad y_n\big(\langle \boldsymbol{w}, \boldsymbol{x}^n\rangle + b\big) = 1 \tag{75}$$

- Iterate until these conditions are met *(to within some numerical tolerance)*

What is `cvxopt`?

- `cvxopt` is a Python package for **convex optimization**
- Please find the documentation of the package $\Rightarrow$ here
- Let's see how it works for linear SVMs

Rewriting the Optimization Problem

Recall our original optimization problem:

maximize

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}^i, \boldsymbol{x}^j \rangle$$

subject to

$$\alpha_i \geqslant 0 \qquad (i = 1, \ldots, N)$$

$$\sum_{i=1}^{N} \alpha_i y^{(i)} = 0$$

`cvxopt` requires a different format:

$$\textit{minimize} \quad \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^\top \boldsymbol{x} \qquad (76)$$

$$\textit{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \qquad (77)$$

$$\boldsymbol{G}\boldsymbol{x} \preceq \boldsymbol{h} \qquad (78)$$

Rewriting the Optimization Problem (Ctd.)

We multiply the objective by -1 to obtain a **minimization problem** and define:

- Let $\boldsymbol{P} \in \mathbb{R}^{N \times N}$ be given by $\left[p_{ij}\right]_{i,j=1,2,\dots,N}$ with

$$p_{ij} := y_i y_j \left\langle \boldsymbol{x}^i, \boldsymbol{x}^j \right\rangle$$

- $\boldsymbol{\alpha} \in \mathbb{R}^N$ (vector of LAGRANGE multipliers)
- $\boldsymbol{q} \in \mathbb{R}^N$ is a constant vector whose components are equal to 1 (i. e. $\boldsymbol{q} := \mathbf{1}$)
- $\boldsymbol{A} \in \mathbb{R}^N$ is the vector containing the labels ($\boldsymbol{A} := \boldsymbol{y}$); $\boldsymbol{b} \in \mathbb{R}$, $\boldsymbol{b} := 0$
- $\boldsymbol{G} := -\boldsymbol{I}_N$ is the negative $N \times N$ identity matrix
- $\boldsymbol{h} \in \mathbb{R}^N$ is a constant vector whose components are equal to zero (i. e. $\boldsymbol{h} := \mathbf{0}$)

Rewriting the Optimization Problem (Ctd.)

- Therefore, we get the following optimization problem:

$$\text{minimize} \quad \frac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{P}\boldsymbol{\alpha} + \mathbf{1}^\top \boldsymbol{\alpha} \tag{79}$$

$$\text{subject to} \quad \boldsymbol{y}^\top \boldsymbol{x} = 0 \tag{80}$$

$$-\boldsymbol{\alpha} \preceq \mathbf{0} \tag{81}$$

- Call of the library function:

`sol = solvers.qp(P, q, G, h, A, b)`

`alphas = np.array(sol["x"])` $\longleftarrow$ LAGRANGE multipliers

Section:

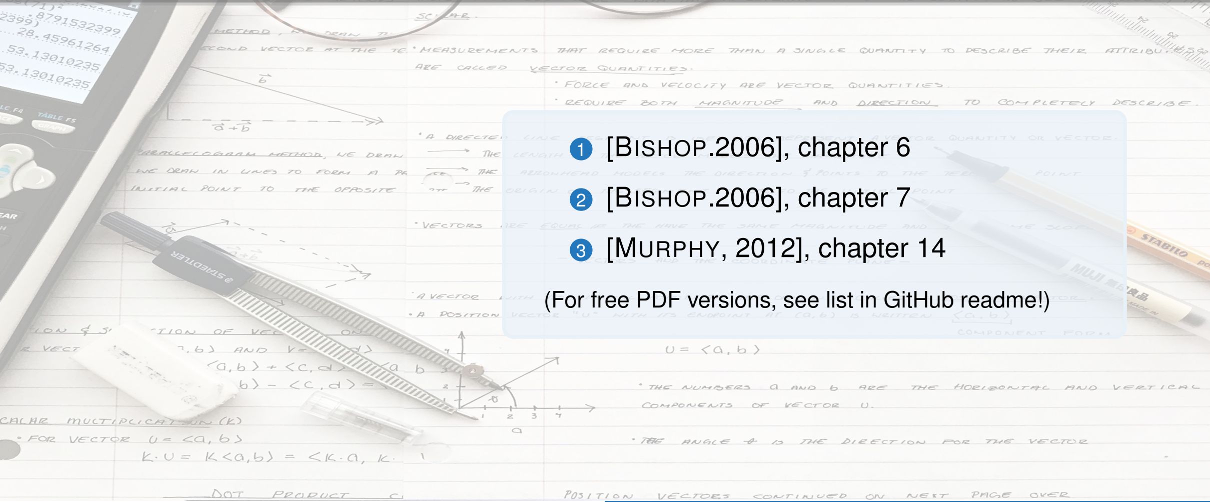
Wrap-Up

- Summary
- Recommended Literature
- Self-Test Questions
- Lecture Outlook

Summary

- Standard SVMs assume the data to be **linearly separable**
- **Generalization guarantee:** SVMs are **maximum margin** classifiers
- The set of **support vectors** defines the decision boundary
- We have to solve a quadratic optimization problem to obtain the support vectors which are needed for prediction
- **Kernels** enable SVMs to classify non-linear data **without having to compute explicit representations** of the data in the high-dimensional feature space
- Slack variables allow for **soft-margin classification**
- **SMO** is an efficient algorithm to solve the quadratic programming problem

Recommended Literature



1. [BISHOP.2006], chapter 6
2. [BISHOP.2006], chapter 7
3. [MURPHY, 2012], chapter 14

(For free PDF versions, see list in GitHub readme!)

Self-Test Questions

1. What is a maximum-margin classifier? What advantages does it have compared to other classifiers?
2. Which data points are needed for prediction? How do we get them?
3. What is a kernel? Can every function serve as a kernel?
4. What prerequisite must be fulfilled so that kernels can be used?
5. Name well-known kernels and write down the equation to compute them!
6. What is a slack variable? What can we do with it?
7. Outline the SMO algorithm!

What's next...?

	I	Machine Learning Introduction
	II	Optimization Techniques
	III	Bayesian Decision Theory
	IV	Non-parametric Density Estimation
	V	Probabilistic Graphical Models
	VI	Linear Regression
	VII	Logistic Regression
	VIII	Deep Learning
	IX	Evaluation
	X	Decision Trees
	XI	Support Vector Machines
●	**XII**	Clustering
	XIII	Principal Component Analysis
	XIV	Reinforcement Learning
	XV	Advanced Regression

Thank you very much for the attention!

*** * * Artificial Intelligence and Machine Learning * * ***

Topic:	Support Vector Machines (SVMs) and Kernel Methods
Term:	Summer term 2025

Contact:
Daniel Wehner, M.Sc.
SAP SE / DHBW Mannheim
daniel.wehner@sap.com

Do you have any questions?